UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

GENERALIZAÇÃO DE FATOS NA COMPREENSÃO DE TEXTOS EM LINGUAGEM NATURAL

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO

JOSÉ NEVES DE LACERDA

FLORIANÓPOLIS, NOVEMBRO DE 1996

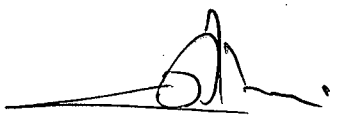
GENERALIZAÇÃO DE FATOS NA COMPREENSÃO DE TEXTOS EM LINGUAGEM NATURAL

JOSÉ NEVES DE LACERDA

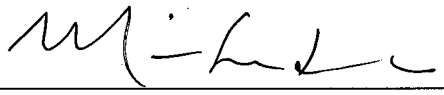
ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA A OBTENÇÃO DO TÍTULO DE

MESTRE EM CIÊNCIA DA COMPUTAÇÃO

ESPECIALIDADE SISTEMAS DE CONHECIMENTO E APROVADA EM SUA FORMA FINAL PELO PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

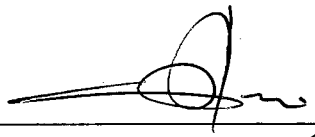


Cezar Augusto Mortari, Dr.Phil. - ORIENTADOR

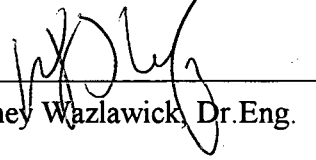


Murilo Silva de Camargo, Dr.Eng. - COORDENADOR DO CURSO

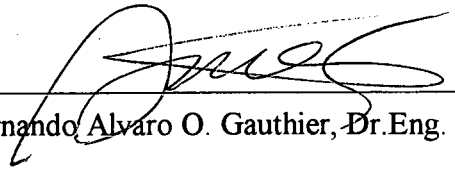
BANCA EXAMINADORA



Cezar Augusto Mortari, Dr.Phil. - PRESIDENTE



Raul Sidney Wazlawick, Dr.Eng.



Fernando Alvaro O. Gauthier, Dr.Eng.

À Lígia e às crianças (Elisa, Conrado e Felipe),

pelo seu amor e paciência.

Agradecimentos

Em primeiro lugar, gostaria de agradecer a meus pais, Roberto e Sylvia Lacerda, que sempre me estimularam a seguir em frente, em todos os momentos de minha vida. Tanto nos momentos felizes, como naqueles não tão felizes, eles sempre estiveram, à sua maneira, ao meu lado.

Em segundo lugar, agradeço ao meu orientador, Professor Cesar Augusto Mortari, Ph.D., pelo estímulo dado para enfrentar esta empreitada, bem como pelo suporte em material, conhecimento e sugestões, sem os quais este trabalho não teria se realizado.

Agradeço ao corpo gerencial do Departamento de Informática da Eletrosul, que viabilizou minha participação neste curso. Agradeço também aos demais colegas da Eletrosul, por sua paciência e dedicação nos momentos em que eu estava "absorto" nos trabalhos acadêmicos.

Um agradecimento ao pessoal da pós-graduação da computação, em especial à Verinha, por sua dedicação aos mestrandos.

Por último, agradeço à minha esposa, Lígia, o amor, o carinho e a paciência que me dedicou durante o período em que estive envolvido com este trabalho.

Resumo

As comunicações humanas são, de forma geral, incompletas em relação ao seu conteúdo. Muito conhecimento é transmitido em poucas palavras. Isso é possível porque existe, em nossa memória, um conhecimento pré-existente sobre o mundo, suas coisas, e interrelações entre estas coisas, que nós utilizamos para complementar as lacunas existentes nas comunicações recebidas. Sistemas que compreendam a linguagem humana precisam ser capazes de fazer esse tipo de atividade, para nós tão natural. O uso adequado desse recurso dá ao sistema uma capacidade de interpretação maior do que a oferecida pela pura análise sintático-semântica das sentenças. A maioria dos sistemas já desenvolvidos recebe este conhecimento através de múltiplas estruturas de dados e regras, codificadas pelos seus autores, para que possam utilizá-las nos processos de interpretação de linguagem.

O objetivo dessa dissertação é descrever um processo automático de criação destas estruturas de dados, de forma a poderem, posteriormente, ser utilizadas na interpretação de textos em linguagem natural. Tratadas aqui como "fatos genéricos", seu papel é o de representar aquilo que conhecemos como senso comum. O princípio que embasa esse trabalho é o seguinte: o senso comum é fruto de inúmeros e sucessivos processos de captação, interpretação, generalização e armazenagem de conhecimento.

Essa dissertação descreve um sistema que "compreende" textos escritos em linguagem natural e, a partir dos fatos identificados nessa compreensão, constrói estruturas semânticas, manipuláveis, que representam genericamente estes fatos. Aqui, apresentamos a base teórica, as definições das estruturas semânticas e das bases de dados utilizadas, os algoritmos de análise, interpretação e generalização de fatos, e demonstramos o funcionamento do sistema implementado. Problemas relacionados às alternativas adotadas são discutidos.

Abstract

Human communications, in a general way, is not complete with regard to its content . Few words are able to transmit a lot of information, because our memory contains prior knowledge about the the world, the things in it and their interrelations, that we use to fill in the blanks in received messages. In order to understand human language, any system must be able to perform this kind of activity, wich is so natural for us. The appropriate use of this ability gives the system a more powerful capability of interpretation than that offered by the pure sintatic-semantic analysis. Most developed systems receive this knowledge in the form of multiple data and rule structures coded by their programmers, and they can use these structures to understand human language.

The purpose of this dissertation is to describe an automatic process for generating these data structures, so that they can be further used in the understanding of natural language texts. Called here "generic facts", their role is to represent the kind of knowledge we know as common sense. This work is based on the following principle: common sense is the result of countless and successive processes of knowledge pickup, interpretation, generalization and storage.

This dissertation describes a system that "understands" natural language texts and, from the facts identified in the understanding process, builds usable semantic structures that generically represent these facts. We present here the theoretical base, the semantic structures and data base definitions used, the parsing, interpretation and generalization algorithms, and demonstrate the functioning of the implemented system. Problems related to the adopted alternatives are also discussed.

Sumário

Índice das Figuras

1 Introdução

> *"Há uma caricatura que mostra dois homens pré-históricos querendo saber, agora que já aprenderam a falar, exatamente sobre o que deveriam conversar. Por mais estranho que pareça, os psicólogos e pré-historiadores modernos não estão melhor informados do que o caricaturista: eles simplesmente não conseguem chegar a um acordo sobre a razão do aparecimento da linguagem."*
>
> (Leakey, 1981)

Com esse parágrafo, Richard Leakey, o conhecido paleontólogo queniano, inicia o oitavo capítulo de seu livro *A Evolução da Humanidade*, intitulado *O nascimento da linguagem.* Entre as diversas teorias apresentadas, a que associa o desenvolvimento da linguagem humana ao da destreza manual é das mais aceitas atualmente. Existe uma íntima relação entre a língua e as mãos. Uma pessoa, quando executa tarefas manuais e traiçoeiras (colocar a linha numa agulha, por exemplo), normalmente movimenta a língua, mordendo-a muitas vezes, como se mãos e língua fossem sócias na mesma atividade. Isso acontece principalmente em crianças. As regiões do cérebro responsáveis por movimentos sutis das mãos e por movimentos da língua estão muito próximas uma da outra. Muitos cientistas da área acreditam que o desenvolvimento da capacidade manual humana, em função do fato de o homem passar a andar ereto, levou a uma conseqüente evolução na sua capacidade de movimentar a língua, dando origem à linguagem falada que conhecemos.

A comunicação humana é uma atividade cognitiva extremamente complexa, a qual fazemos naturalmente durante o nosso dia-a-dia. Ela envolve desde sentidos apurados (visão, audição, tato), até capacidade motora afinada, que permitirá um perfeito uso da língua e das mãos. O responsável pelo perfeito funcionamento disso tudo e, principalmente, pelo armazenamento das informações e regras linguísticas a serem utilizadas é o cérebro. Um defeito em um órgão sensorial ou motor, entre os mencionados acima, os quais podemos aqui considerar como dispositivos de entrada e saída, dificulta mas não impede totalmente a comunicação humana. Entretanto, uma falha no cérebro pode comprometer totalmente a capacidade de comunicação de um indivíduo, podendo torná-lo parcial ou totalmente isolado do mundo que o cerca. Exemplos disso são a dislalia (dificuldade de articular palavras), dislexia (dificuldade de ler palavras escritas) e dislogia (perturbação da expressão verbal, gerando interrupções

súbitas em meio a frases). Outras falhas podem provocar o total esquecimento de fatos acontecidos, mesmo os mais recentes, prejudicando totalmente a comunicação.

A compreensão da linguagem humana pelos computadores

Para adaptarmos essa atividade cognitiva aos computadores, duas grandes classes de problemas se abrem. Em primeiro lugar existe a tarefa de fazer os computadores compreenderem a linguagem propriamente dita, utilizando todo o conhecimento léxico, sintático e semântico característico da linguagem, bem como a informação exigida do mundo real. Em segundo lugar temos que fazer o computador compreender a linguagem falada, utilizando todas as informações mencionadas anteriormente, mais o conhecimento adicional a respeito da fonologia e, outras informações úteis, relativas à ambigüidade da fala.

A pesquisa em lingüística computacional - o uso de computadores no estudo da linguagem - começou logo após os computadores terem ficado disponíveis, nos anos 40. A habilidade da máquina para manipular símbolos foi prontamente aproveitada para compilar listas com ocorrências de palavras e concordâncias. Esse tipo de processamento teve algum valor na pesquisa lingüística, mas logo tornou-se claro que o computador poderia desempenhar funções lingüísticas muito mais poderosas do que estas.

Em 1949, Warren Weaver propôs que os computadores poderiam ser úteis como tradutores. A pesquisa resultante, chamada *Tradução por Máquina* tentou simular com um computador as funções normalmente executadas por um tradutor humano. A tradução era feita através da procura das palavras num dicionário bilíngüe, que mapeava na língua de saída as palavras expressas na língua de entrada.

Apesar da simplicidade da idéia, muitos problemas apareceram, relativos à seleção das palavras apropriadas e ao trabalho de arranjá-las para produzir uma sentença na língua de saída. Este conceito de tradução foi posteriormente abandonado.

Em seu lugar, a *compreensão* tornou-se o foco principal das pesquisas de Inteligência Artificial (IA) em linguagem natural - se a máquina pudesse realmente *entender* o sentido de uma sentença, ela poderia parafraseá-la, responder a perguntas sobre ela ou traduzi-la para outra língua. Mas a natureza da compreensão é, por si só, um problema difícil.

A partir dos anos 60, os pesquisadores de IA desenvolveram um novo grupo de programas de computação para lidar com a linguagem natural. Esses primeiros programas marcaram o início do trabalho, em IA, sobre o entendimento da linguagem.

Foram desenvolvidos programas com os enfoques mais diversos, desde casamento de palavras chaves (ELIZA (Weizenbaum, 1966) e PARRY (Colby, 1975)) até processamento de estruturas episódicas e temáticas (BORIS (Dyer, 1983)).

Esses resultados, aliados aos novos conhecimentos sobre linguagem descobertos por psicólogos e lingüistas, fizeram com que a linguagem humana começasse a ser vista como uma habilidade cognitiva complexa, envolvendo diferentes tipos de conhecimento, relacionados:

- à estrutura das sentenças;
- ao sentido das palavras;
- ao modelo do ouvinte;
- às regras de conversação;
- a um extenso e compartilhado corpo de informações gerais sobre o mundo.

Toda pesquisa em compreensão de linguagem natural tem sido norteada por objetivos que podem ser resumidos em:

- desenvolvimento de sistemas práticos e úteis que compreendam a linguagem natural, para atender principalmente às seguintes finalidades:
  a) tradução por máquina;
  b) compreensão de textos;
  c) interface natural com bancos de dados;
  d) diálogo com máquina.
- melhor compreensão da linguagem humana e da natureza da inteligência humana.

Objetivos do trabalho

O objetivo desta dissertação é definir e demonstrar o funcionamento de um sistema que compreende textos escritos em linguagem natural e, a partir dessa compreensão, constrói estruturas que representam genericamente os fatos mencionados nos textos. Tal sistema, como concebido originalmente, deveria possuir as seguintes características básicas:

a) aquisição supervisionada de conhecimentos léxico, gramatical e conceitual;
b) utilização de estruturas conceituais para representação do conhecimento;
c) geração de estruturas conceituais episódicas a partir da interpretação de textos em linguagem natural, a fim de representar os fatos mencionados;
d) capacidade de generalização, a partir das estruturas conceituais episódicas obtidas no processo de interpretação, gerando estruturas conceituais genéricas.

A razão da escolha pela generalização de fatos reside numa característica do aprendizado humano que é a generalização de conceitos. Nós, seres humanos, generalizamos quase tudo que observamos, de forma a construir padrões que serão posteriormente utilizados na compreensão de futuras observações. Esses padrões nos permitem inferir muita coisa a respeito do que acontece à nossa volta, facilitando bastante nosso entendimento a respeito do mundo e do que nos é comunicado pelas outras pessoas. Estruturas genéricas para facilitar a compreensão de textos por computadores têm sido usadas por diversos pesquisadores. Os *scripts* (Schank, 1986) são um exemplo. Eles são, na realidade, padrões genéricos de comportamento previamente fornecidos a programas de compreensão de textos, e que os auxiliam na identificação dos elementos (personagens, cenários, ações, causas, conseqüências, etc.) do texto.

Nosso objetivo não é utilizar tais padrões genéricos, mas sim gerá-los a partir de fatos apresentados. Acreditamos que bases de informações genéricas para suporte à compreensão da linguagem natural serão de grande importância no futuro. O processo de obtenção dessas bases terá que ser automatizado ao máximo, face o volume das informações envolvidas. Neste trabalho apresentamos um sistema que, a partir de fatos informados (doravante chamados de fatos episódicos) sob forma de textos, interpreta estes fatos e generaliza-os, de modo a construir uma base de fatos genéricos que poderiam, por sua vez, auxiliar em novos processos de interpretação de textos, consultas, etc.

Estrutura do trabalho

A primeira parte dessa dissertação trata da fundamentação teórica necessária à compreensão do assunto. Inicialmente discorremos sobre a problemática da compreensão de textos com uma ou mais frases, abordando as análises sintática, semântica e pragmática (capítulo 2). Depois, vemos algumas representações declarativas de conhecimento (capítulo 3), normalmente utilizadas por programas de IA para representar os conhecimentos necessários ao desempenho de suas funções. Abordamos em mais detalhes as redes semânticas (que foi nossa opção de representação neste trabalho), além

de quadros, dependência conceitual, roteiros (*scripts*) e pacotes de organização de memória (MOP). Também discorremos um pouco sobre "Memória", de acordo com Schank (1986). No capítulo 4 vemos alguma coisa sobre aprendizado de conceitos e de linguagem.

Na segunda parte, apresentamos a definição do sistema e demonstramos seu funcionamento. Inicialmente, apresentamos a simbologia adotada para a construção das redes semânticas que representarão os conhecimentos gerados pelo sistema (capítulo 5). Em seguida, apresentamos as estruturas da base de dados que armazenarão o léxico, a gramática e todos os elementos das redes semânticas geradas (capítulo 6). O capítulo 7 descreve o funcionamento do analisador de textos, também chamado de *parser*. O capítulo 8 descreve o processo de generalização de fatos, bem como problemas relacionados à solução adotada. O capítulo 9 demonstra, através de imagens de telas, o funcionamento do sistema. No capítulo 10 apresentamos nossas conclusões, bem como futuros trabalhos que podem ser feitos em continuidade a este.

Parte I - Fundamentação Teórica

2 Compreensão e Geração de Linguagem Natural

2.1 Compreensão

Quando uma pessoa fala algo para outra, existe a comunicação de uma mensagem de um emissor para um receptor e, este, para compreender o que foi comunicado, precisa:

- captar a mensagem (ouvir sons; visualizar gestos, escritos);
- identificar os componentes da mensagem (palavras) e verificar se são aceitáveis (regras léxicas);
- verificar se estas palavras estão interrelacionadas de acordo com um padrão aceitável pela linguagem utilizada (regras sintáticas);
- verificar se o significado das palavras combinam entre si (regras semânticas);
- procurar identificar o real significado da mensagem, que expressa a intenção do emissor (pragmática).

Com base no real significado da mensagem, o receptor terá condições de reagir respondendo com outra mensagem, ou de outra forma qualquer.

Em termos teóricos, pode-se afirmar que uma entidade qualquer compreende uma mensagem, se conseguir transformá-la da representação de origem (língua fonte) para uma representação alvo, que possa ser utilizada para tomar decisões, fazer inferências, responder perguntas, parafraseá-la, etc. Se pegarmos como exemplo a frase *"preciso ir a São Paulo tão logo seja possível"* (representação origem), seu significado (representação alvo) vai ser diferente para diferentes tipos de interlocutores e provocar decisões e ações diferentes, se houver compreensão:

Tipo de Interlocutor	**Interpretação**	**Ação Desejada**
Funcionário de Linha Aérea	Preciso do primeiro avião disponível.	Selecionar alternativas de vôos para escolha.
Funcionário de Banco	Preciso de dinheiro urgente para poder ir a São Paulo.	Verificar créditos para analisar possibilidade de empréstimo.
Médico	Preciso estar em condições de saúde para poder ir a São Paulo.	Verificar situação e elaborar prognósticos e roteiros de ação.

No exemplo acima cada interlocutor transformou a representação de origem em uma representação alvo específica. Isso demonstra que, para uma tarefa de compreensão, o conhecimento sobre o mundo relacionado à mensagem real que está sendo transmitida é vital. Isso é conhecido como "modelo de ouvinte".

A implementação da compreensão de linguagem natural por computadores exige que se pense na linguagem como sendo a composição de língua (representação) fonte, representação alvo, mapeamento das ligações entre língua fonte e representação alvo, e modelo de ouvinte.

Segundo Rich (1988), existem três fatores que contribuem para dificultar a compreensão de linguagem natural por computadores:

- a complexidade da representação alvo, a qual vai determinar o grau de compreensão do sistema em questão. Conforme o nível de compreensão desejada aumenta, a complexidade da representação alvo deve ser maior. Segundo Schank (1984), existem três níveis diferenciados de compreensão, baseados nas respostas formuladas por uma entidade qualquer, seja ela humana ou uma máquina:

  a) *fazer sentido* - ao receber uma notícia de jornal como entrada, por exemplo, a entidade fornece um resumo sobre a notícia;

  b) *compreensão cognitiva* - ao receber um conjunto de estórias sobre acidentes de aviões, completadas com estruturas de conhecimento sobre aviões, vôos e

circunstâncias sobre viagens de aviões, a entidade fornece, por exemplo, conclusões altamente fundadas sobre o que pode ter provocado um acidente ocorrido recentemente;

c) *empatia completa* - ao receber observações sobre um problema pessoal, a entidade exprime pensamentos que dão a entender que compreende a situação e pode ajudar, falando de suas próprias experiências similares àquelas.

- o tipo de mapeamento gerado, que pode ser:

  a) *Um-para-um* - são normalmente os existentes em linguagens de programação, uma vez que estas não admitem ambigüidades.

  b) *Muitos-para-um* - são mais comuns. Ocorrem com freqüência quando se mapeia de uma linguagem natural para uma representação alvo mais simples. Como exemplo, podemos ter as seguintes frases - *"Que horas são?, Pode me dizer as horas?, O senhor sabe que horas são?, Tem horas?"* - todas para representar a mesma idéia.

  c) *Um-para-muitos* - são os casos de ambigüidade, i.é., quando uma expressão, ou sentença, pode levar a diversas representações alvo. Como exemplo, podemos ter: *"Eu vi o rapaz na praça com um telescópio"*. As possíveis interpretações para esta frase podem ser:
     - Através de um telescópio eu vi o rapaz na praça;
     - Eu vi o rapaz na praça com um telescópio debaixo do braço;
     - Eu vi o rapaz na praça que possui um telescópio.

  d) *Muitos-para-muitos* - são os casos em que ocorrem simultaneamente os mapeamentos de tipo Muitos-para-Um e Um-para-Muitos. Acontece com praticamente todas as línguas humanas.

- o nível de interação existente entre os componentes da linguagem, que nas linguagens humanas é bastante alto. Isso significa que uma simples mudança em uma palavra ou símbolo de pontuação pode mudar completamente a estrutura e o sentido de uma declaração. Como exemplo, temos as frases *"quem canta, seus males espanta"* e *"quem canta seus males, espanta"*. A simples mudança na posição da vírgula fez com que as frases ficassem sintática e semanticamente diferentes.

Para que um programa de computador consiga compreender textos em linguagem natural, é preciso que ele desempenhe tarefas para atender a dois objetivos básicos:

a) compreensão de frases individuais;
b) compreensão de frases múltiplas (parágrafos e textos mais extensos).

2.2 Compreensão de frases individuais - sintaxe, semântica e pragmática

Para que tenhamos sucesso no processo de compreensão de frases individuais, podemos dividi-lo em três componentes:

- Análise Sintática - Transformação da seqüência de palavras em estruturas que mostram como as palavras se relacionam entre si. As rejeições acontecem quando palavras violarem as regras da linguagem que regem a combinação de palavras;

- Análise Semântica - Designação de significados às estruturas criadas pelo analisador sintático. Faz-se um mapeamento entre as estruturas sintáticas e os significados no domínio de atividade. As rejeições acontecem quando esse mapeamento não é possível;

- Análise Pragmática - Reinterpretação da estrutura que representa o que foi dito, para se determinar o que realmente se quis dizer. Por exemplo, a frase - *"Você sabe que horas são?"* - se interpretada somente sob o ponto de vista sintático-semântico, levará a uma resposta do tipo *"Sim, eu sei"* ou *"Não, eu não sei"*. Sob o enfoque pragmático, deve ser interpretada como uma solicitação para serem informadas as horas.

Como não existem, muitas vezes, fronteiras claras entre essas três fases, alguns analisadores executam-nas ora em seqüência, ora simultaneamente.

2.2.1 Análise Sintática

Método que, dada uma gramática, permite determinar se uma sentença qualquer pertence à lingua especificada pela gramática e, se pertencer, identificar as estruturas da gramática que podem ser associadas a esta sentença (Petrick, 1990).

Formalmente, uma gramática **G** é definida pela quádrupla **G=(VN,VT,S,P)**, onde

VN = vocabulário não terminal
VT = vocabulário terminal
S = símbolo inicial
P = um conjunto finito de produções

O símbolo V representa a união dos conjuntos VN e VT e é chamado de vocabulário. VN e VT não possuem elementos em comum.

Cada produção P é da forma $X \rightarrow Y$, onde X e Y pertencem a V e X não é nulo.

Segundo Lyons (1979), Chomsky delineou quatro tipos de gramática, numerados de 0 a 3, da mais geral para a mais particular:

- **Tipo 0** - Uma gramática de tipo 0 é definida como acima. É um conjunto de produções sobre um vocabulário dado, sem restrições quanto à forma das produções.

- **Tipo 1** - Uma gramática é de tipo 1 se a forma de suas produções é restrita, tal que para cada produção $X \rightarrow Y$ da gramática, Y possui pelo menos o mesmo número de símbolos de X. Também chamadas de gramáticas sensíveis ao contexto.

- **Tipo 2** - São as gramáticas livres de contexto, onde em suas produções ($X \rightarrow Y$) X tem que ser um único símbolo não terminal. As linguagens humanas se enquadram, em sua maioria, neste tipo.

- **Tipo 3** - São as gramáticas regulares. Suas produções podem ser da forma $X \rightarrow aY$ ou $X \rightarrow a$, onde X e Y são símbolos não terminais isolados e *'a'* é um símbolo terminal único.

A seguir temos um exemplo de gramática livre de contexto (tipo 2):

VN = (S, SN, SV, ARTDEF, SUBSTANTIVO, VERBO),
VT =(o, a, menino, bola, chutou),
P = (**S → SN SV,**
SN[1] **→ ARTDEF SUBSTANTIVO,**
SN → SUBSTANTIVO,
SV[2] **→ VERBO SN,**
ARTDEF → o | a,
SUBSTANTIVO → menino | bola,
VERBO → chutou)

Aplicando as produções definidas acima ao vocabulário especificado, pode-se gerar as seguintes sentenças (algumas delas, e somente com elementos do vocabulário terminal):

a menino chutou a menino
a menino chutou a bola
a menino chutou o menino
...
a bola chutou a bola
a bola chutou o menino
a bola chutou o bola
...
o menino chutou a bola
o menino chutou o menino
...
menino chutou a bola
menino chutou o menino
menino chutou o bola
...
bola chutou menino
bola chutou bola

Uma das principais características dessas gramáticas é que, adequadamente implantadas, permitem fazer o processo inverso, isto é, determinar se uma sentença é válida ou não para a gramática especificada. Esse processo é conhecido como *parsing*. Se aplicarmos as regras acima para a sentença *"o menino chutou a bola"*, a resposta da gramática será

[1]SN - Sintagma Nominal

[2]SV - Sintagma Verbal

positiva, enquanto que para a sentença *"o menino chutou a lata"* a resposta será negativa (*"lata"* não está definido no vocabulário). Pode-se notar também que a gramática acima é bastante tolerante em comparação com as regras gramaticais da nossa língua, aceitando construções consideradas inválidas. Para que possam evitar essas situações, as gramáticas precisam ser mais restritivas.

O processo de parsing permite gerar uma estrutura gráfica, conhecida como árvore sintática, que representa a estrutura sintática da sentença analisada. A sentença *"o menino chutou a bola"*, geraria a seguinte árvore sintática:

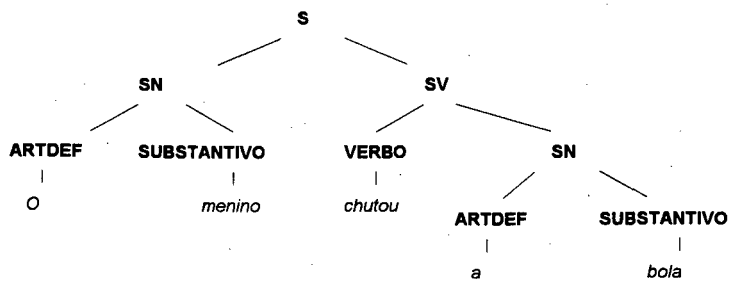


Fig. 2.1 - Exemplo de árvore sintática

Existem diversos formalismos que podem ser adotados na implementação de parsers, tais como *cima-para-baixo* e *esquerda-para-direita*. Dentro da pesquisa computacional, os dois formalismos mais adotados, segundo a bibliografia pesquisada, têm sido as ATNs (Redes de Transição Aumentada) (Winograd, 1983) e as DCGs (Gramáticas de Cláusulas Definidas) (Pereira, 1980). Ambas têm sido utilizadas com sucesso para implementar modelos eficientes de analisadores sintáticos, sendo que particularmente as DCGs possuem a propriedade de poderem ser convertidas em programas da linguagem Prolog.

2.2.2 Análise Semântica

"O termo 'semântica' significa significado ou o estudo do significado. O significado de uma palavra ou sentença em linguagem natural é a entidade ou ação que ela denota. Na compreensão de linguagem por computadores, interpretação semântica é o processo de determinar o significado dos dados de entrada. Isto pressupõe que se tem, em primeiro lugar, uma representação computacional para o significado, e, em segundo lugar, um método para derivar esta representação a partir de uma entrada" (Hirst, 1990).

A análise sintática determina a estrutura de uma sentença e a análise semântica determina seu significado. As duas abordagens são igualmente importantes e se complementam.

Enquanto que o produto da análise sintática é importante para o analisador semântico determinar o significado, a análise semântica fornece elementos que facilitam a desambiguação léxica de uma sentença.

Como exemplo da importância da análise sintática na determinação do significado de uma sentença temos: *"Bola chutou o menino"*. Se o analisador semântico não souber que *"Bola"* é o sujeito da oração (o agente), provavelmente vai associá-lo àquele brinquedo esférico. No entanto, se souber que é o agente, e sabendo que um agente da ação *"chutar"* tem que ser humano (ou pelo menos possuir pernas), vai procurar por uma ocorrência de *"Bola"* mais apropriada para o caso (por exemplo, o apelido de um outro menino).

Da mesma forma, a análise semântica ajuda a análise sintática. Por exemplo, na sentença *"Pedro dirige o carro com pneus carecas"*, o analisador sintático se depara com duas interpretações possíveis para *"pneus carecas"*: a primeira, como uma característica do carro; a segunda, como o instrumento da ação. Nesse caso, o analisador semântico fornece informações que permitem ao analisador sintático a escolha da estrutura mais correta.

Muitas controvérsias existem com relação ao fato de se estipular qual delas deve ser executada em primeiro lugar. Muitos defendem a tese de que ambas devem ser executadas simultaneamente. Hirst (1990) menciona um trabalho de Marcus, em que ele demonstra que "num caso mais genérico, uma interpretação semântica completa necessita de uma análise sintática completa, embora esta necessidade não seja sempre verdadeira para as aplicações mais simples; e, desde que os conhecimentos sintáticos e semânticos são relativamente independentes, uma metodologia de programação apropriada sugere que os dois sejam mantidos em módulos separados".

Pode-se classificar as abordagens desenvolvidas para solucionar este problema, da seguinte maneira (Rich, 1988):

a) Gramáticas Semânticas;
b) Gramáticas de Caso;
c) Filtragem semântica das estruturas sintáticas geradas, através dos seguintes métodos:
   - fazer a filtragem à medida em que as ambigüidades forem sendo encontradas. Ex.: ATN (Winograd, 1983);
   - Aguardar até que o processo de análise sintática esteja completado e avaliar as estruturas resultantes quanto à aceitabilidade semântica;

d) Desenfatizar a análise sintática e acionar o processo de compreensão por conhecimento semântico e não sintático, usando, por exemplo, dependência conceitual e *demon-parsers*[3] (Dyer, 1983) e (Schank, 1986).

2.2.2.1 Gramáticas Semânticas

Gramáticas semânticas são gramáticas livres de contexto, onde a escolha de símbolos não terminais e regras de produção é governada por funções semânticas, além das funções sintáticas (Rich, 1988). Rich apresenta um fragmento simplificado de uma gramática semântica utilizada pelo sistema LADDER, que provê acesso em linguagem natural a uma grande base de dados distribuida, para uso por oficiais da Marinha:

S → qual é **PROPRIEDADE-NAVIO** de **NAVIO** ?
PROPRIEDADE-NAVIO → a **PROP-NAVIO** | o **PROP-NAVIO** | **PROP-NAVIO**
PROP-NAVIO → velocidade | comprimento | contingente | largura | tipo
NAVIO → **NOME-NAVIO** | o mais rápido **NAVIO2** | o maior **NAVIO2** | **NAVIO2**
NOME-NAVIO → Kennedy | Kitty Hawk | Constellation | ...
NAVIO2 → **TIPO-NAVIO LOC** | **TIPO-NAVIO**
TIPO-NAVIO → **PAÍSES TIPO2** | **TIPO2**
TIPO2 → porta-aviões | submarino | barco a remo | ...
PAÍSES → americano | francês | inglês | russo | ...
LOC → no Mediterrâneo | no Pacífico | ...

Esta gramática possui categorias semânticas não terminais (**NAVIO, LOC, PAÍSES**), em vez das tradicionais categorias sintáticas **NP** e **VP**. Assim, a frase *"Qual é a velocidade do [de+o] mais rápido porta-aviões americano no Pacífico?"* está sintática e semanticamente correta. Entretanto, a frase *"Qual é a altura do mais rápido porta-aviões americano no Pacífico?"* seria rejeitada, pois apesar de sintaticamente correta, *'altura'* não corresponde semanticamente ao universo da linguagem que foi definida pelas regras de produção.

---

[3]Demon-parser - Analisador de textos baseado em demônios. Veja Quadros e Dependência Conceitual.

2.2.2.2 Gramáticas de Caso

"A noção de 'caso' tem sido usada para fazer referência a diversos conceitos relacionados. Tradicionalmente, significa a classificação das palavras de acordo com o seu papel sintático em uma sentença, sinalizado por várias formas de flexão. No Inglês, apenas os pronomes possuem estes casos de flexão. Por exemplo, o pronome singular da primeira pessoa é 'I' (caso nominativo), 'me' (caso acusativo/objetivo), ou 'my' (caso genitivo/possessivo), de acordo com o seu uso como sujeito, objeto ou artigo possessivo. Em línguas como o Grego, todas as palavras possuem sufixos que indicam seus casos" (Bruce, Moser, 1990).

Sob o ponto de vista da compreensão de linguagem natural, Bruce e Moser categorizam os tipos de caso em superficiais e profundos. Casos superficiais são aqueles em que as palavras são categorizadas de acordo com seus sufixos, flexões, ou mesmo são consideradas marcadores sintáticos, os quais permitem atribuir um papel especial a qualquer termo da sentença. Também são conhecidos como casos de nível sintático. Casos profundos são aqueles em que existem categorizações dos sintagmas nominais (SNs) de acordo com papéis conceituais que eles desempenham na sentença, independente do verbo ou predicado. Como exemplo, um caso 'agente' pode ser a generalização de diversos conceitos, tais como: leitor, atirador, trabalhador. Ou seja, aquele que executa uma ação. Estes casos também são conhecidos como casos 'semânticos'.

Dessa forma, as gramáticas semânticas permitem representar de uma única forma relações semânticas idênticas, independentemente das sentenças possuirem estruturas sintáticas diferentes. Como exemplo, temos estas duas sentenças com estruturas sintáticas distintas:

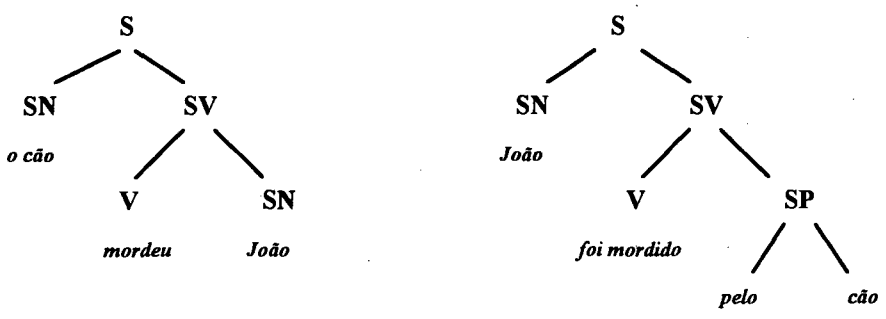


Fig. 2.2 - Estruturas sintáticas diferentes para mesma estrutura semântica

O fato descrito nas duas sentenças é o mesmo, mas, sintaticamente, existem diferenças muito grandes. Enquanto *'cão'* é sujeito em uma frase, ele é agente da passiva na outra. Por sua vez, *'João'* que é objeto na primeira, é sujeito da segunda. Numa gramática de caso estas diferenças deixariam de acontecer porque a representação enfatiza o aspecto semântico do fato descrito. As duas representações sintáticas acima poderiam ser semanticamente representadas da seguinte forma:

Mordeu (Agente (*cão*), Paciente (*João*))

2.2.2.3 Dependência Conceitual

Tal como no uso das gramáticas de caso, a representação através de dependência conceitual(CD) é fortemente caracterizada pelo enfoque semântico, especificamente pelo uso de primitivos semânticos. Este assunto será discutido em maiores detalhes no item 3.4.

2.2.3 Análise Pragmática

"Pragmática é o estudo da comunicação como ela está situada em relação a um conjunto particular de necessidades de comunicação, emissores, receptores, tempos, lugares, ambiente, convenções ligüísticas e práticas culturais" (Scha, Bruce, Polanyi, 1990).

Segundo este autores, a pesquisa da compreensão de linguagem natural, sob a ótica da pragmática, possui três implicações importantes:

- o significado de uma mensagem expressa lingüisticamente está representado apenas parcialmente pelo conteúdo da mensagem; para o receptor é importante a percepção das intenções do emissor ao produzir a mensagem;
- a atribuição de intenções ao emissor deve ser componente integral do processo de compreensão do receptor;
- a teoria da compreensão da linguagem deveria determinar o âmbito, para o qual, as mesmas estratégias que as pessoas usam para chegar a explicações razoáveis sobre o comportamento físico das outras pudessem ser empregados na compreensão dos atos da fala.

Um bom exemplo em que a interpretação da real intenção do emissor depende de uma série de fatores, entre eles o próprio receptor, é o citado no item 2.1 (*"preciso ir a São*

Paulo tão logo seja possível"). Como citado acima, neste caso o significado está apenas parcialmente representado pelo contéudo da mensagem e cabe ao receptor (ouvinte) identificar a real intenção do emissor durante o seu processo de compreensão.

2.3 Compreensão de Frases Múltiplas

Compreender um conjunto de frases, quer num pedaço de texto, quer numa porção de diálogo, além da compreensão de cada frase, requer que as relações entre as frases sejam conhecidas.

As relações que podemos encontrar em um texto são inúmeras. Apresentaremos, a seguir, alguns exemplos destas relações (Rich, 1988):

1) **Objetos Idênticos** - *"Carlos tem um carrinho a pilha. José o quer"*. O pronome *'o"* se refere ao carrinho. Estas referências são conhecidas como referências anafóricas, ou anáforas. Chama-se anáfora o fenômeno de se fazer referência a elementos que já apareceram. Essas referências são feitas por pronomes ou expressões resumidas.

2)**Partes de Objetos** - *"Mário tirou a televisão da caixa. O botão do volume estava quebrado"*. Nesse caso, *"o botão do volume"* faz parte do objeto *'televisão'* que acabara de ser tirada da caixa.

3)**Partes de Ação** - *"João foi em viagem de negócios a São Paulo. Ele partiu em vôo cedo pela manhã"*. Da ação *"foi em viagem"* entende-se que o *'vôo'* é parte dessa ação.

4) **Objetos Envolvidos em Ações** - *"Bill decidiu ir até a loja, mas o carro não funcionou"*. O objeto *'carro'* está envolvido na ação de *"Bill ir à loja"*.

5) **Cadeias Causais** - *"Ontem rompeu uma adutora da CASAN. Hoje, os bairros estavam sem água"*. Fatos que aconteceram no passado afetam o presente e/ou futuro.

6) **Seqüências de Planejamento** - *"Maria queria um carro novo. Ela decidiu arranjar um trabalho"*. O plano de *"arranjar um trabalho"* foi traçado para obter meios de 'comprar' *"um carro novo"*.

O conhecimento sobre o mundo real é necessário para que se possa identificar as possíveis relações existentes entre duas ou mais sentenças. Veremos a seguir algumas maneiras de se explorar as representações do conhecimento na compreensão de frases múltiplas, de maneira a resolver as relações mencionadas acima:

- **Utilização de Espaços de Foco** - A utilização de espaços de foco facilita e agiliza o processo de compreensão de frases, envolvendo os seguintes passos:

  a) focalizar a parte relevante na base de conhecimentos disponível, a partir de palavras chaves encontradas no texto;
  b) utilizar estes conhecimentos focalizados para resolver ambigüidades e formular relações entre as coisas que foram ditas.

- **Utilização de Estruturas de Meta** - A compreensão de relações entre objetos e eventos físicos é insuficiente quando temos que analisar uma estória que apresente pessoas e suas ações. Neste caso, é interessante utilizar a estrutura de meta, a qual fornece as ações (planos) que normalmente são executadas para alguém atingir uma meta. Para tal devemos identificar as metas dos personagens e os planos que devem ser explorados para atingi-las. Cada meta, por sua vez, pode ser subdividida em sub-metas específicas, as quais corresponderão a planos mais detalhados, e assim sucessivamente.

- **Utilização de Roteiros (Scripts)** - Roteiros são utilizados basicamente com três objetivos:
  1) orientar o processo de compreensão do texto, dando-lhe flexibilidade e eficiência;
  2) prever eventos que não foram mencionados explicitamente;
  3) relacionar os eventos citados entre si.

2.4 Geração de Linguagem

O processo de geração de linguagem pode ser visto como o oposto da tarefa de compreensão de linguagem. Todo processo de geração de linguagem pode ser dividido em:

- construir uma estrutura que represente a informação a ser comunicada;
- aplicar regras de estrutura de texto e de diálogo;
- aplicar informação léxica e regras sintáticas para gerar frases.

A primeira fase é mais complicada em função da utilização do sistema. Podendo o mesmo ser:

a) Restrito : o processo de geração de informação é respondedor de questões;

b) Não restrito : o processo gera texto onde a questão importante é a construção de uma estrutura que represente uma seqüência interessante de eventos.

A segunda e a terceira fase do processo de geração supõem que o conteúdo a ser comunicado está fixo, sobrando apenas o problema de comunicá-lo eficientemente.

Embora compreensão e geração de textos sejam atividades opostas, os problemas enfrentados por uma normalmente também são enfrentados pela outra. Para gerar textos bons, todos os problemas que surgem na compreensão de linguagem devem ser resolvidos. Como exemplo, o problema das referências anafóricas:

João viu uma bicicleta na vitrine da loja. Ele queria a bicicleta.

Utilizando um pronome resolve-se o problema:

João viu uma bicicleta na vitrine de uma loja. Ele a queria.

Os pronomes só podem ser utilizados quando não produzem ambigüidades:

João viu um chapéu num carro vermelho. João queria o chapéu.

João viu um chapéu num carro vermelho. João o queria. (o quê?)

A terceira fase de geração de frases é a construção de frases gramaticais a partir de representação do conhecimento subjacente. Como exemplo, podemos utilizar as ATN (Redes de Transição Aumentada) para a compreensão de frases. Para gerar frases a partir destas estruturas, basta colocarmos nos arcos as condições e ações de regras que compõem as frases.

A geração de textos em linguagem natural enfrenta, além desses, outros problemas, como a necessidade de decisão entre o uso de voz ativa ou voz passiva numa determinada frase. McKeown (1985,1986) discute isso e propõe soluções.

3 Representações Declarativas do Conhecimento

3.1 Redes Semânticas

3.1.1 O que são redes semânticas?

As redes semânticas são um meio de representar o conhecimento humano, proposto originalmente por M.R. Quillian em 1966, em sua tese de doutorado. Quillian queria representar as conceitos das palavras inglesas, de forma que esse conhecimento pudesse ser manipulado.

Uma rede semântica é composta basicamente de nodos, os quais representam objetos quaisquer (pessoas, objetos, eventos, etc.), e de ligações, ou arcos rotulados, fazendo com que um nodo referencie outro. Esse conjunto forma uma definição. A idéia é similar à de um dicionário, onde se encontra a definição de uma determinada palavra expressa em função de outras palavras, cujas definições também se encontram espalhadas pelo dicionário.

A figura 3.1 exemplifica algumas redes semânticas, usando simbologia proposta por Schubert, Goebel e Cercone (1979).

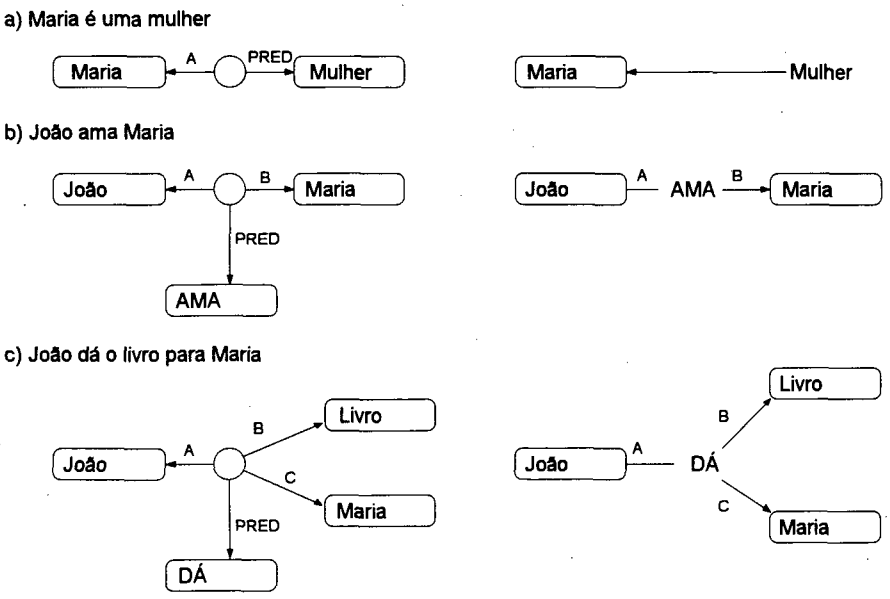


Fig. 3.1 - Exemplos de redes semânticas

No exemplo anterior, usou-se notação completa para representar as redes da esquerda e notação abreviada para as redes da direita. A simbologia adotada é a seguinte:

Símbolo	**Significado**
Rótulo	Nodo de conceito. Normalmente associado às palavras. Substantivos, verbos e adjetivos são representados dessa forma.
○	Nodo da proposição. É o elo de ligação entre os demais nodos. Representa, normalmente, a instância do que se quer representar.
Rótulo →	Ligação. Representa as ligações existentes entre os nodos. O rótulo da ligação, normalmente, possui significado semântico. No exemplo anterior, PRED indica o predicado da proposição. As letras indicam, na ordem, os demais parâmetros. Poderíamos, no lugar das letras, usar rótulos com significado, tais como *agente, paciente,* etc.

Note que a supressão do nodo da proposição na forma abreviada é apenas aparente. A substituição deste pelo nodo conceitual do predicado faz com que os dois nodos sejam representados por apenas um deles. Ainda assim, deve-se interpretá-los como sendo uma instância do nodo conceitual.

3.1.2 A importância das redes semânticas

Em redes semânticas, as ligações existentes entre os nodos são bi-direcionais, isto é, um aponta para o outro, e vice-versa, apesar do arco possuir apenas um sentido. Essa característica permite que se estabeleça uma "vizinhança semântica" para cada conceito.

Define-se como vizinhança semântica de um nodo qualquer, todos os conceitos e relacionamentos alcançáveis através de um número arbitrário de ligações. Esse conceito embasa a suposição fundamental do uso de redes semânticas, que é:

> *"O conhecimento necessário para o desempenho de qualquer tarefa intelectual geralmente reside na vizinhança semântica dos conceitos envolvidos na tarefa."*
>
> (Schubert, Goebel, Cercone, 1979).

Tarefas descritivas, tais como *"descreva o quarto de Maria"*, são exemplos dessa utilização. Neste caso, muito provavelmente, os objetos e conceitos relacionados ao *"quarto de Maria"* estão a uma, duas ou, no máximo, três ligações de distância. Conceitos mais remotos dificilmente terão algo a ver com o objeto da questão. A figura 3.2 ilustra essa situação.

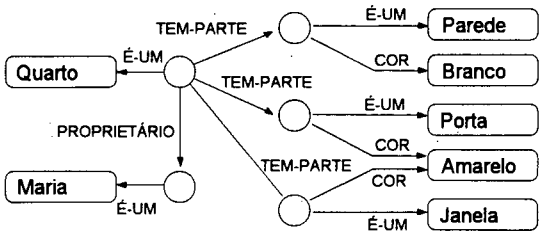


Fig. 3.2 - Parte da descrição do quarto de Maria

Essa organização do conhecimento também oferece a possibilidade de se fazer inferências simples. Como exemplo, se tivermos as seguintes afirmações:

a) Todo homem que ama uma mulher gosta de estar junto dela;
b) Pedro ama Maria;
c) Pedro está em Curitiba;
d) Maria está em Florianópolis.

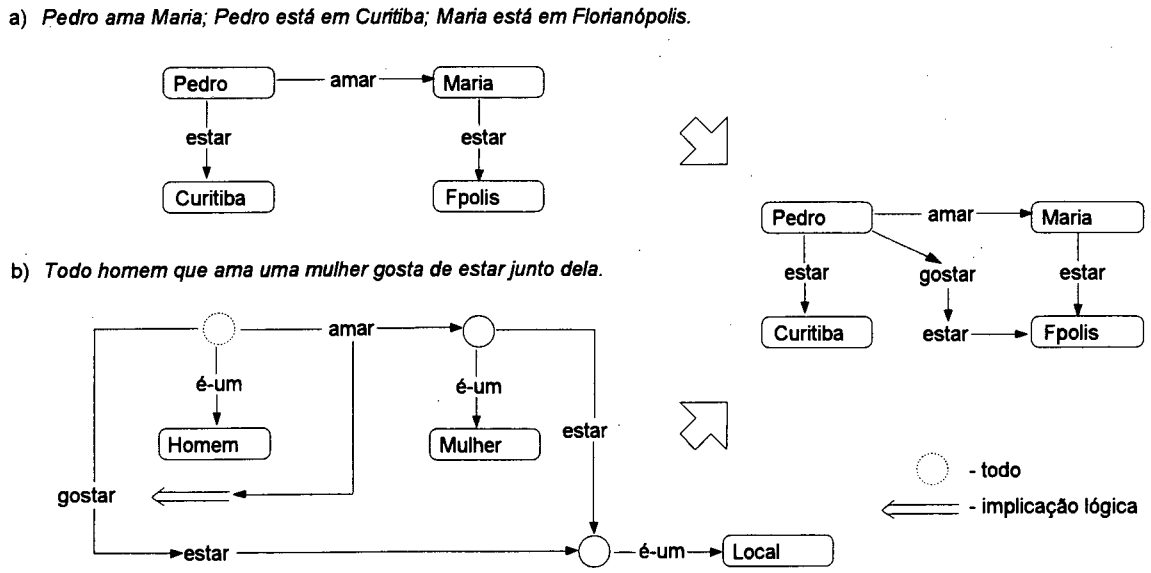


Fig. 3.3 - Exemplos de redes semânticas.

Se fizermos a pergunta "*onde Pedro gostaria de estar agora?*", poderemos inferir a resposta através das redes semânticas que representam as idéias acima, como demonstrado na figura 3.3.

A sobreposição das redes *a* e *b* permite inferir que *'João'* gostaria de estar em *'Florianópolis'*, independentemente de onde esteja no presente momento.

Essa característica das redes semânticas estimulou inúmeras pesquisas, principalmente com relação a simbologias de representação e a algoritmos de busca. Estes permitiriam maior poder de inferência aos sistemas e, aquelas, maior veracidade na representação dos fatos.

3.1.3 Formalismo das redes semânticas

Praticamente cada artigo que se escreve sobre redes semânticas apresenta uma simbologia própria, normalmente de acordo com o grau de formalismo que o autor quer dar ao assunto.

Com relação às *ações* propriamente ditas, alguns as representam sob a forma de combinações de primitivas (Schank, 1986), o que é computacionalmente desejável, mas sacrifica muito a veracidade da representação; já Schubert (1979) as representa nas suas formas verbais complexas.

Sob outro ponto de vista, a organização proposta por Schubert apresenta um padrão de referência para suficiência lógica da rede. Essa organização apresenta clara correspondência com a lógica dos predicados, permitindo que se represente praticamente qualquer proposição expressável em linguagem humana.

Brachman discute esses e outros pontos acerca das diversas representações das redes semânticas, e conclui que grande parte dos problemas e deficiências encontrados em tal diversidade de representações se deve à mistura inconsciente de primitivos de diversos níveis conceituais em uma representação. Dele citamos: *"Um conceito em particular poderia ser estruturado com casos semânticos. Esses casos podem ser entendidos como sendo conjuntos de predicados lógicos, os quais são implementados como conjuntos de átomos e ponteiros. Contudo, cada esquema de rede propõe um conjunto explícito de primitivos como parte da linguagem, e este é o conjunto que é suportado pelo interpretador da linguagem. ..."* (Brachman, 1979, p.27-28).

Para que se possa representar esses diversos níveis, ou pontos de vista, Brachman propõe cinco níveis de representação. Em sua opinião, qualquer notação de rede poderia ser analisada em termos de qualquer um dos cinco níveis, uma vez que nenhum deles possui uma *existência* independente. São eles:

Nível	**Explicação**
Implementação	Possui como primitivos átomos e ponteiros, e permite representar as estruturas dos dados;
Lógico	Possui como primitivos proposições, predicados e operadores lógicos, e permite representar proposições lógicas;
Epistemológico	Possui como primitivos tipos de conceitos, subtipos conceituais, heranças e relações de estruturação, e permite representar relacionamentos entre conceitos, heranças de atributos e valores, múltiplas descrições, etc.
Conceitual	Possui como primitivos relações conceituais ou semânticas (casos), objetos e ações primitivas, e permitem representar as relações semânticas entre os objetos. As representações de Schank são exemplos desse nível.
Lingüístico	Possui como primitivos conceitos arbitrários, palavras, expressões. Permite representar relacionamentos que existem no mundo em foco, em uma forma dependente da linguagem, e espera-se que os significados de seus primitivos mudem à medida que a rede cresce.

3.1.4 Analisando em maior detalhe as heranças

Conforme visto anteriormente, a herança entre conceitos se situa no nível epistemológico de representação de uma rede semântica. O conceito de herança também é amplamente discutido nas áreas de bancos de dados e orientação a objetos, notadamente nos níveis de projeto e implementação (Booch, 1991) (Rumbaugh, 1994).

A herança é um mecanismo pelo qual as informações atribuídas a conceitos gerais podem ser *herdadas* por suas especializações e instanciações. A figura 3.4 exemplifica esse conceito.

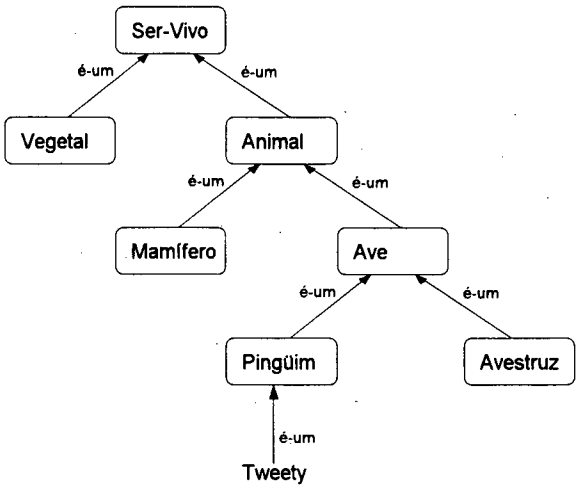


Fig. 3.4 - Heranças Simples

Ao afirmarmos que *'Tweety'* é um *'pingüim'*, automaticamente estamos afirmando que todos os atributos válidos para *'pingüim'*, *'ave'*, *'animal'* e *'ser-vivo'* também são válidos para *'Tweety'*. Assim, se atribuirmos *'data-nascimento'* a *'ser-vivo'*, estamos também atribuindo-a a *'Tweety'*. Os casos representados na figura acima são conhecidos como casos de herança simples, que permitem representar os muitos filhos que um conceito pai pode ter.

Examinando a figura 3.5,

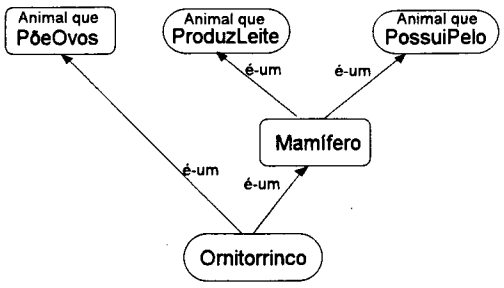


Fig. 3.5 - Herança Múltipla

encontramos exemplos de herança múltipla, onde um conceito *herda* atributos e valores de mais de um outro conceito, ou seja, um filho com muitos pais. Esses dois casos de herança (simples e múltipla) podem ocorrer simultaneamente em uma rede semântica.

Entretanto, um problema adicional pode aparecer quando se fala em heranças, e isso está demonstrado na figura 3.6.

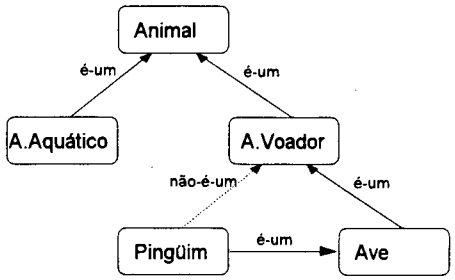


Fig. 3.6 - Heranças com Exceções

Ao afirmarmos que um *'pingüim'* é uma *'ave'*, indiretamente estamos afirmando que também é um *'animal-voador'*. Entretanto, sabemos que um *'pingüim'* não voa. Isso é expressado na rede através da ligação *não-é-um*. Essa dupla relação, de pertinência e não pertinência, gera um paradoxo e é bastante difícil de ser tratada, principalmente se levarmos em conta as múltiplas heranças que podem estar envolvidas.

Para resolver computacionalmente esse problema, Sandewal (1986) propõe regras de inferência não monotônicas para heranças múltiplas com exceções. Resumidamente, ele propõe uma forma de encontrar o caminho mais curto entre dois conceitos quaisquer, através de ligações *é-um* e *não-é-um*. O processo de inferência gera novas ligações *é-um* e *não-é-um*, as quais estabelecem a relação direta real entre qualquer par de conceitos (x,y). Assim, no final, através de uma busca simples, pode-se saber se o conceito x é um, ou não é um, conceito y. A figura anterior, após esse processo, ficaria assim:

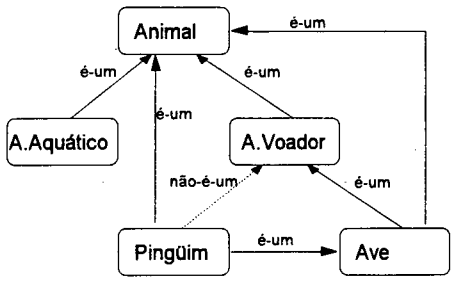


Fig. 3.7 - Heranças após tratamento de exceções

Neste caso, *'pingüim'* herda características de *'ave'* (excluidas as de *'animal voador'*) e de *'animal'*.

3.2 Memória

Um dos recursos básicos que qualquer entidade, seja ela máquina ou homem, deve possuir para lograr êxito na tarefa de compreensão de linguagem é a memória. Sem ela, nós não conseguimos associar os fatos apresentados a outros já ocorridos, de forma a podermos executar as inferências necessárias à tarefa de compreensão.

Schank (1986) classifica a memória em diversos tipos, ou níveis, em função do que as pessoas são capazes de lembrar:

- **Memória de Evento (EM)**

Responsável pelo armazenamento dos fatos ocorridos ou assimilados. Permite lembranças específicas sobre situações particulares. Também referenciada como memória episódica. Após certo tempo, tende a perder os aspectos menos salientes sobre os fatos, guardando apenas os mais importantes ou pouco usuais.

- **Memória de Evento Generalizado (GEM)**

Fatos similares que ocorrem diversas vezes tendem a gerar lembranças genéricas, mais permanentes do que as anteriores. Assim, tendemos a lembrar genericamente das consultas a médicos, dentistas, idas a supermercados, etc. Com relação a estes fatos, somente lembramos detalhes daqueles mais recentes e detalhes marcantes dos mais antigos. Todo o resto são lembranças superficiais sobre o padrão dos fatos em si. Essa é conhecida como memória de evento generalizado. Sempre que novos fatos similares aos anteriores ocorrerem, serão automaticamente vinculados à GEM correspondente, e seus aspectos menos relevantes serão rapidamente esquecidos.

- **Memória Situacional (SM)**

Nível mais genérico que o anterior, armazena informações que são comuns a mais de uma GEM. Assim, se considerarmos *consulta-a-médico* e *consulta-a-dentista* como GEMs distintas, ambas possuem uma SM única do tipo *consulta-a-profissional-de-saúde*, na qual temos compartilhadas informações sobre salas de espera, secretárias, planos de saúde, etc.

- **Memória Intencional (IM)**

A memória intencional contém memórias preponderantemente voltadas para metas. Assim, contextos gerais, como viagens, eleições e namoros, cujas metas imediatas são conhecidas, são estruturas IM. A organização das informações é feita em função da razão da sua existência. As confusões nesse nível acontecem quando situações diferentes envolvem intenções semelhantes.

Um programa de computador, para compreender linguagem natural deve, pelo menos, dispor do primeiro nível de memória. Um sistema que disponha do segundo já é capaz de identificar situações baseado em eventos genéricos armazenados previamente. Um que disponha do terceiro nível já possui muito mais flexibilidade e economia de armazenamento das informações necessárias à compreensão. A presença do quarto nível fornece maior capacidade de fazer inferências.

3.3 Quadros (Frames)

Quadro é uma estrutura de dados utilizada para representar uma situação estereotipada. Minsky (1985), em seu artigo, tenta formular uma teoria, onde o conhecimento a respeito do mundo, mais conhecido como senso-comum, pode ser representado de forma estruturada, de maneira que possa ser *lembrado* e *alterado* conforme a situação se apresenta.

O objetivo básico da teoria dos quadros é permitir a implementação, em computadores, do raciocínio conhecido como senso-comum. Uma implementação desse tipo consiste de um módulo de conhecimento que se torna ativo em uma situação apropriada, e serve para oferecer interpretações e novas previsões (expectativas) daquela situação.

Para demonstrar sua teoria, Minsky propõe que nos imaginemos em frente a uma porta fechada, em uma casa que não conhecemos. À nossa mente vêm diversos quadros mentais acerca do que talvez exista atrás da porta. Imagens de quartos, salas, cozinhas, jardins, etc., vêm-nos à mente, mas não se fixam porque nenhum indício existe para reforçar um ou outro. Entretanto, ao visualizarmos, por uma fresta dessa porta, qualquer coisa do outro lado, imediatamente um dos quadros tende a se fixar, enquanto abandonamos os demais através do processo de esquecimento. Imaginemos que visualizamos um pedaço de uma cama com um travesseiro. O quadro que se fixa em nossa mente é o de um quarto. Novas dúvidas se apresentam. Se é um quarto (porque

tem uma cama), então, provavelmente, possui um armário, uma mesinha de cabeceira, um abajur, etc. Essas dúvidas se traduzem em novas informações que têm que ser buscadas, para que se possa montar (e reforçar) o quadro completo sobre o quarto. Para se trabalhar com este tipo de situação é que se concebeu a teoria de quadros, ou frames.

É fácil concluir que essa idéia é bastante similar ao conceito de *vizinhança semântica* de Quillian. Entretanto, os quadros possuem características importantes, não vistas em redes semânticas.

3.3.1 O que é quadro ?

Podemos imaginar um quadro como sendo uma rede de nodos e relacionamentos, onde os nodos de nível mais alto são fixos, e representam coisas que são sempre verdadeiras sobre a situação suposta. Os de nível mais baixo são conhecidos como *slots*, ou nodos terminais, que devem ser preenchidos com dados ou instâncias de outros quadros. Esquematicamente, podemos ter um quadro da seguinte maneira:

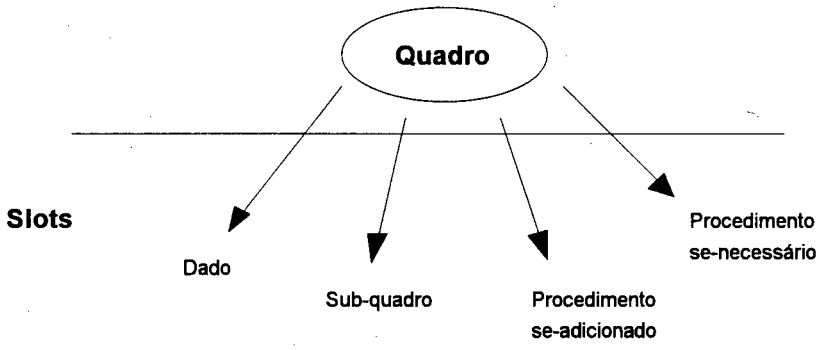


Fig. 3.8 - Exemplo de Quadro (Frame)

Um determinado quadro pode possuir muitas instâncias, ou ocorrências. Por exemplo, *"o quarto de Maria"* e *" o quarto de Pedro"* são instâncias do quadro *'quarto'*.

Em um determinado quadro, seus slots podem ser:

a) Dados — *Quaisquer valores associados a atributos do quadro.*

b) Procedimentos se-adicionado — *São invocados cada vez que uma instância do quadro é adicionada, ou seja, criada.*

c) Procedimentos se-necessário — *São invocados cada vez que se necessita de uma determinada informação associada ao quadro, a qual não está explicitada ainda, gerando dúvida.*

d) Sub-quadros — *Aponta para outros quadros, menores ou não, permitindo a elaboração de um sistema de quadros, o qual representa todo o conhecimento sobre uma determinada situação.*

A operacionalização dos procedimentos *se-adicionado* e *se-necessário* é feita através de demônios (*demons*). Estes são pequenos procedimentos, com filosofia de processamento paralelo, que são responsáveis:

- pelo preenchimento de certos *slots*, quando uma instância de um quadro é adicionada;
- pelo estabelecimento de ligações entre a instância em questão e outras instâncias de outros quadros, se for o caso;
- pela criação automática de instâncias de outros quadros, se necessário;
- pela eliminação de sua respectiva instância (suicídio), caso ela não esteja de acordo com o contexto que está sendo formado.

3.3.2 Atribuição de Valores Padrões (Default)

Toda vez que um quadro é criado, diversos *slots* são automaticamente preenchidos com valores padrões para aquela situação. Assim, quando ouvimos alguém falar *"o programador está perto do terminal"*, imediatamente imaginamos um profissional de informática próximo a um terminal de vídeo, trabalhando com o computador. Isso ocorre porque o nosso quadro para *'programador'* possui estes valores padrões para esta situação. Entretanto, se posteriormente nos é mostrada a imagem do programador próximo a um terminal rodoviário, ou de aviões, temos que refazer alguns *slots* de nosso quadro, a fim de torná-lo mais de acordo com a realidade.

Assim, muitos *slots* podem possuir valores padrões, os quais normalmente estão *fracamente* associados ao quadro. Isso permite que se modifique a estrutura do conhecimento mais tarde, conforme a evolução da situação que se apresenta.

3.3.3 Aplicações para Quadros

As estruturas baseadas em quadros são utilizadas para construção de bases de dados de conhecimento (Knowledge Data Bases - KDB), principalmente em:

- Sistemas Especialistas;
- Compreensão de Linguagem Natural;
- Compreensão de Imagens (Visão por Computador).

3.3.4 O Uso de Quadros na Compreensão de Linguagem Natural

Uma vez que a estrutura de um quadro permite interpretar uma determinada situação, bem como formular expectativas sobre elas, eles se tornam bastante úteis no processo de reconhecimento. A seguir, alguns exemplos comentados.

1 - a) *João foi a um restaurante.*
b) *Pediu um hambúrguer à garçonete.*
c) *Pagou a conta e saiu.*

Este exemplo nos demonstra uma situação coerente, em que o quadro inicialmente invocado (*'restaurante'*) tem seus slots posteriormente confirmados pelas frases subseqüentes.

2 - a) *João foi ao parque.*
b) *Ele pediu um rato ao anão.*
c) *Pegou a caixa e saiu.*

Este exemplo nos dá a idéia de *desorientação*. As situações apresentadas pelas diversas frases não possuem um quadro em comum, nem ligações entre os diversos quadros, que permitam dar algum tipo de coerência à situação como um todo.

3 - a) *Ele entregou $5,00 pelo guichê.*
b) *Ela tentou entregar-lhe $2,50, mas ele não aceitou.*
c) *Então, quando eles entraram, ela comprou-lhe um grande saco de pipoca.*

Este exemplo nos dá a idéia de *confusão*. Após a primeira frase, diversos quadros se candidatam a representar a situação descrita (*cinema, jóquei clube, teatro,* etc). A segunda frase não ajuda muito, pois aumenta ainda mais a confusão (*quem tentou entregar-lhe os $2,50? A moça do guichê ?*). Somente com a terceira frase é que a situação se esclarece. Do conjunto de quadros invocados anteriormente, somente o referente a *'cinema'* se ajusta, pois é lá que normalmente (como padrão) se come

'pipoca'. Outro fato que fica esclarecido é o de que *foi a moça com quem o rapaz se encontrou que tentou entregar-lhe os $2,50, para pagar sua parte da entrada.* Essa situação nos dá a idéia, então, de um *'encontro'*, entre um *'rapaz'* e uma *'moça'* em um *'cinema'*. Pode se afirmar, conforme (Schank, Childers, 1984), que um programa que interpretasse corretamente este texto, estaria fazendo-o no nível de compreensão cognitiva.

4 - *João contou a Pedro que havia ficado 1 hora na fila do posto de gasolina para colocar $10 de combustível. Pedro lembrou-se que na semana anterior havia ficado 40 minutos na fila do correio para comprar 1 selo.*

Este exemplo nos dá a idéia de *analogia.* Quando João contou a Pedro sua história, Pedro, automaticamente, montou em sua mente *quadros* para representar a situação. Assim, criou-se um quadro para *'posto'*, associado ao qual havia um *slot* com significado *perda-de-tempo-por-pouca-coisa.* Como Pedro possuia em suas experiências pessoais um quadro com o mesmo tipo de *slot*, automaticamente fez a ligação e lembrou-se do fato. Um programa que soubesse da estória de *'João'* e se lembrasse de fatos similares, assim como o fez Pedro, teria um nível de compreensão equivalente a empatia completa (Schank, Childers, 1984).

3.3.5 Herança em quadros

Os quadros, assim como os objetos e os nodos das redes semânticas, podem herdar características de outros quadros. Essa herança pode ser feita na forma de árvore (herança simples) ou de reticulado (herança múltipla). O uso de hierarquias IS-A (é-um) permite a herança de *slots* de quadros superiores para outros em posição mais especializada. Não só suas definições são herdadas, mas também seus conteúdos, suas informações. Um exemplo dessa herança pode ser visto a seguir:

```
[quadro : EVENTO
        is-a : ... (qualquer coisa)
        slots : (TEMPO (um INSTANTE-TEMPO)(valor default século vinte))
                (LOCAL (uma LOCALIZAÇÃO) (valor default Brasil))]
[quadro : AÇÃO
        is-a : EVENTO
        slots : (AGENTE (uma PESSOA))]
[quadro : CAMINHADA
        is-a : AÇÃO
        slots : (ORIGEM (uma LOCALIZAÇÃO))
                (DESTINO (uma LOCALIZAÇÃO))]
```

Essa característica dos quadros é particularmente útil porque elimina a necessidade de redundância no armazenamento de informações já tratadas anteriormente. O exemplo a seguir ilustra bem essa característica:

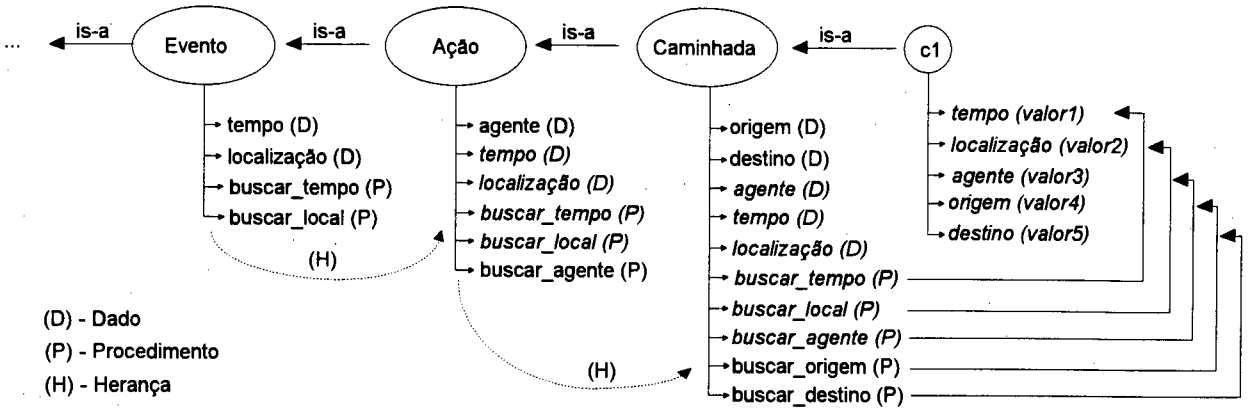


Fig. 3.9 - Heranças entre quadros

Na figura 3.9, os *slots* que aparecem em itálico são herdados do quadro *pai*. Os demais são atributos próprios do quadro *filho*.

3.4 Dependência Conceitual - CD

Desde o início das suas pesquisas, Schank (1986) tem perseguido o objetivo de orientar os trabalhos de compreensão de linguagem natural com um enfoque mais semântico do que sintático. É dele e de sua equipe a idéia de utilizar dependências conceituais para permitir que fatos sejam representados de acordo com o seu real significado, eliminando possíveis ambigüidades naturais da linguagem humana.

3.4.1 O que é dependência conceitual ?

Dependência Conceitual (CD) (Dyer,1983) é um sistema projetado para representar os sentidos das sentenças pela decomposição delas em um número pequeno de *ações* (acts) primitivas. De acordo com esta teoria, sentenças com mesmo significado terão a mesma representação, independentemente de diferenças gramaticais ou de línguas utilizadas. Segundo a teoria de CD, cada sentença tende a apresentar apenas uma pequena porção do significado através de suas próprias palavras. O resto é obtido através de pesquisa a

memória, planejamento e inferências. Como exemplo, temos a seguinte frase : *"João comprou uma televisão"*. A partir desta sentença, entende-se que:

1) Existe outra pessoa envolvida;
2) João deu dinheiro a esta pessoa;
3) Esta pessoa deu a televisão a João em troca do dinheiro;
4) João comprou a televisão para assistir programas nela, provavelmente para sua própria diversão;
5) João provavelmente comprou a televisão em uma loja;
6) A loja não possui mais a televisão; etc.

A maioria dessas informações não aparecem na sentença acima. As pessoas chegarão às mesmas conclusões se forem apresentadas à sentença a seguir: *"Uma televisão foi vendida para João"*. Apesar de sintaticamente distintas, o conteúdo informacional das duas sentenças é o mesmo. Isso motivou a criação das CDs.

3.4.2 Os componentes da dependência conceitual

Basicamente, para se construir uma CD, são necessárias as seguintes categorias de conceitualizações primitivas(Rich, 1988):

ACTs	Ações;
PPs	Objetos (produtores de imagens);
AAs	Modificadores de Ações (auxiliadores de ação);
PAs	Modificadores de PPs (auxiliadores de imagem).

Além disso, existe um conjunto de regras que permitem a representação das relações semânticas existentes entre os conceitos referenciados.

As ações primitivas (ACTs) dessa teoria são onze, e para cada uma delas existem associados uma série de quadros para tratar as expectativas geradas durante o processo de compreensão. São elas :

1. **ATRANS** Transferência da posse de um objeto físico;
2. **PTRANS** Transferência de localização física de um objeto;
3. **MTRANS** Transferência de informação mental;
4. **INGEST** Ingestão de um objeto ou substância;
5. **PROPEL** Aplicação de uma força física a um objeto;

6. **ATTEND** Enfocar um órgão sensorial (olhos, ouvidos,...) em um estímulo;
7. **SPEAK** Produzir sons verbalmente;
8. **MBUILD** Criar novo conceito mental baseado em conhecimento prévio;
9. **MOVE** Mover uma parte do corpo;
10. **GRASP** Fechar a mão sobre um objeto, segurar o objeto;
11. **EXPEL** Expulsão de objeto ou substância do corpo de pessoa ou animal.

Os modificadores de ações (AAs) servem para suprir informações que normalmente estão indicadas na linguagem pelo tempo, modo ou aspecto da forma do verbo. Como exemplos, temos:

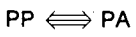
p	Passado
f	Futuro
t	Transição
ts	Início de Transição
tf	Fim de Transição
k	Continuidade
?	Interrogativo
/	Negativo
nil	Presente
delta	Intemporalidade
c	Condicional

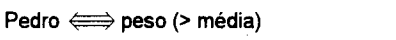
Os modificadores de PPs (PAs) servem para melhor caracterizar um objeto, descrevendo seu estado atual bem como as mutações que sofre. As regras *b* e *d* (abaixo) contêm exemplos de PAs.

A seguir vemos algumas regras que permitem a montagem de estruturas de dependência. Note que cada estrutura de dependência pode ser considerada *primitiva* na construção de estruturas mais complexas.

Regra	**Representação**	**Exemplo**
a) Relação entre um ator e o evento que ele provoca.	$PP \overset{T}{\Longleftrightarrow} ACT$ T = tempo	$Pedro \overset{P}{\Longleftrightarrow} PTRANS$ *Pedro andou.*

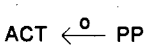
b) Relação entre um objeto e uma característica sua.





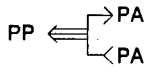
Pedro é gordo.

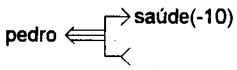
c) Relação entre uma ação e o objeto dessa ação.



Pedro empurrou o carro.

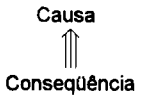
d) Relação entre um objeto e sua mudança de estado.

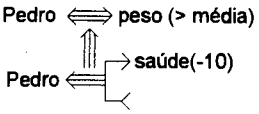




Pedro morreu.

e) Relação entre uma conceitualização e outra que é sua causa.





Pedro morreu porque era gordo.

Outra característica importante dos sistemas baseados em CDs é que o analisador (parser) é dirigido pelos demônios (demons) definidos nos quadros. Eles são responsáveis pela busca de informações no texto, ou fora dele (na memória), que vão satisfazer a algum slot vago. Um tipo de léxico que pode ser montado de acordo com esta teoria é o seguinte:

Palavra	**Representação**			**Observações**
disse	(MTRANS[4]	ACTOR	X	⇐ demônio : espera que já tenha sido dito quem está executando a ação.
		OBJECT	*	⇐ demônio : espera saber de alguma ação, declaração ou meta.
		FROM	X	
		TO	*	⇐ demônio : espera que a pessoa alvo da ação seja o objeto das preposições "para" ou "a".)

[4] MTRANS => Transferência de informação mental

3.5 Roteiros (Scripts)

Roteiros são estruturas de informação concebidas para auxiliar a compreensão de situações onde o comportamento está tão estilizado que raramente é necessário o uso de análises complexas voltadas para planos e metas. Organizados como uma seqüência de estruturas *frame-like*, eles permitem, como já mencionado anteriormente, atender aos seguintes objetivos:

1) orientar o processo de compreensão do texto, através do fornecimento de expectativas, dando-lhe flexibilidade e eficiência;
2) prever eventos que não foram mencionados explicitamente;
3) relacionar os eventos citados entre si.

Um exemplo típico de roteiro é o do restaurante, o qual pode, em linhas gerais, ser definido da seguinte forma:

RESTAURANTE	ENTRAR-RESTAURANTE +
	SENTAR +
	GARÇOM-ATENDE +
	FAZER-PEDIDO +
	TRAZER-COMIDA +
	COMER-COMIDA +
	PEDIR-CONTA +
	RECEBER-CONTA +
	PAGAR-CONTA +
	DEIXAR-GORJETA +
	SAIR-RESTAURANTE

Cada evento do roteiro (tarefa) possui os demônios (*demons*) necessários à busca e satisfação, ou frustração, das expectativas associadas. Por sua vez, cada uma das tarefas mencionadas pode ser mais detalhada através de um roteiro próprio. Como exemplo temos:

SENTAR	PROCURAR-MESA-VAGA +
	ANDAR-ATÉ-MESA +
	SENTAR-NA-CADEIRA

O uso dessas expectativas ajudam em tarefas como :

1) Resolução de pronomes ⇒ além das tradicionais referências anafóricas, também existem aqueles problemas do tipo - *"ele pagou-lhe a gorjeta"*. Com o uso das expectativas, sabe-se que é o cliente quem está pagando ao garçom a gorjeta.

2) Desambiguação léxica ⇒ na frase - *"o programador está perto do terminal"* - o roteiro associado a programador ajuda a não confundir terminal de computador com terminal de cargas, por exemplo.

3) Auxílio na elaboração de inferências ⇒ na frase - *"quando foi ao restaurante, descobriu que estava sem dinheiro na hora de pagar a conta"* - o fato de pagar a conta sugere que pediu-se comida, e que esta foi ingerida.

Outra característica dos roteiros é que eles podem possuir caminhos alternativos. No exemplo SENTAR acima, pode-se ter, além de PROCURAR-MESA-VAGA, algo do tipo SOLICITAR-MESA-AO-MAITRE.

Os roteiros, como foram concebidos originalmente (Schank, Childers, 1984) (Schank, 1986) (Dyer, 1983), apresentam problemas, os quais podem ser resumidos como segue:

- roteiros foram concebidos como "nacos" de conhecimento isolados. Não é possível compartilhar fatos similares em roteiros diferentes, nem relacioná-los a outros fatos. A redundância torna-se necessária;
- o mesmo fato impede a generalização das informações contidas em um roteiro;
- roteiros não permitem a representação de intencionalidade. Assim, os fatos se sucedem porque foram definidos dessa forma. O sistema não pode responder perguntas do tipo "por que?".

3.6 Pacotes de Organização de Memória (MOP)

A idéia dos MOPs foi concebida baseada na de Memória Situacional (SM), e visa sanar deficiências apresentadas pelos roteiros. Enquanto estes são "nacos" monolíticos e isolados, os MOPs possuem ligações mostrando como eles foram montados a partir de outras fontes de conhecimento. Cada ligação conecta um evento de um MOP a algum evento em outro MOP. Estas ligações são realizadas dinamicamente por meio de demônios (veja Quadros).

Além da eliminação de redundâncias, a utilização de MOPs permite a percepção das informações sob múltiplos pontos de vista, traduzindo as expectativas dos diversos participantes de uma determinada situação. O exemplo a seguir, procura demonstrar estas características:

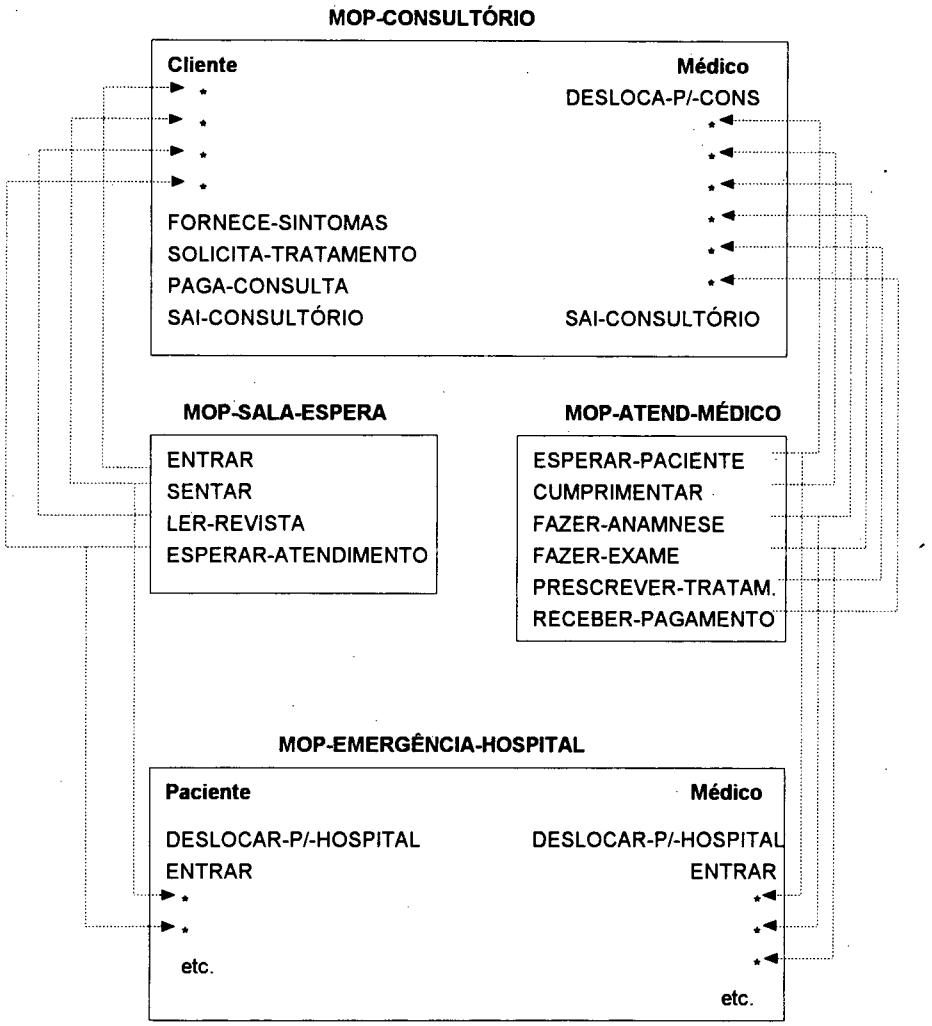


Fig. 3.10 - Exemplo de MOP.

Como se pode notar, existe uma série de padrões de informações que são compartilhadas por mais de um MOP (SENTAR, FAZER-EXAME, etc.). Além disso, os MOPs permitem que se avalie as situações por mais de um ponto de vista. Assim, se tivermos o seguinte trecho - *"João foi levado às pressas para a emergência do hospital. O diagnóstico foi grave"* - a primeira frase traz à tona o MOP-EMERGÊNCIA-HOSPITAL, o qual traz embutidas expectativas tanto sob o ponto de vista do paciente como do médico. Pode-se inferir, pela segunda frase, que pelo menos um FAZER-EXAME ocorreu por parte do médico.

Essas estruturas são genéricas. Os fatos, os episódios, são armazenados aproveitando-se apenas o que de concreto foi informado ou inferido. Ligações entre as estruturas genéricas e as episódicas são feitas para permitir *lembranças* futuras.

Essas estruturas foram implementadas com sucesso em BORIS (Dyer, 1983), o qual interpreta estórias sobre alguns tipos de relações sociais (casamento, divórcio, etc.), e responde perguntas sobre os textos apresentados, notadamente sobre causas dos fatos e motivações e intenções das personagens.

4 Aprendizado

Nesta dissertação, o assunto "aprendizado" se refere aos estudos dos métodos computacionais para aquisição de novos conhecimentos. Especificamente para este trabalho, interessa o aprendizado de conceitos e a aquisição de linguagem.

4.1 Aprendizado de Conceitos

O aprendizado de conceitos é uma das características dos seres dotados de inteligência. Significa o agrupamento de objetos, seres, comportamentos, eventos, em classes de entidade de acordo com princípios ou características em comum. Isso ajuda bastante a atividade intelectual, bem como a própria sobrevivência. Mesmo os animais, por exemplo os carnívoros, costumam classificar quais outros animais são "alimentos" e quais não são.

Especificamente no caso computacional, o aprendizado de conceitos pode ser feito de acordo com as seguintes técnicas (Michalski, 1990) :

a) Implantação Direta do Conhecimento → é o caso em que o conhecimento é alimentado inteiramente pelo criador do sistema, sem nenhuma interferência ou inferência por parte do mesmo;

b) Aprendizado por Instrução (com professor) → é o caso dos sistemas que adquirem seu conhecimento através de um processo dirigido, ou supervisionado. Nestes casos, o sistema recebe o auxílio para classificar as informações (mais/menos importantes), representá-las de uma maneira mais apropriada, etc.;

c) Aprendizado por Dedução → é o caso em que o sistema adquire novos conhecimentos a partir de conhecimentos já disponíveis, através de transformações que preservam o valor verdade desses conhecimentos. Como exemplo, um sistema pode aprender que o fatorial de 5 é 120 através da execução de um procedimento já conhecido (cálculo de fatorial) e do armazenamento dessa informação em sua base de conhecimento;

d) Aprendizado por Analogia → é o caso em que o sistema aprende um novo conceito pela modificação de um conceito anterior já conhecido. É o caso de se aprender o conceito de tangerina após já se saber o que é laranja. Apenas algumas modificações e se tem o conceito de tangerina.

e) Aprendizado por Indução → é o caso em que o sistema constrói conceitos através de processos de inferência a partir de fatos fornecidos ou observados. Dependendo das informações que são fornecidas ao aprendiz (o sistema) e do grau de interferência do instrutor, o aprendizado por indução pode ser subclassificado em aprendizado a partir de exemplos e aprendizado a partir de observações e descoberta. Dependendo da forma como os conceitos são gerados e pesquisados, os métodos de aprendizado por indução podem ser: orientado por dados, orientado por modelo e, misto.

4.1.1 Aprendizado a partir de exemplos

Neste processo, o sistema induz uma descrição conceitual pela generalização de exemplos, e contra-exemplos também, fornecidos pelo professor (humano) ou pelo ambiente. Um exemplo típico desse tipo de aprendizado é o algoritmo AQ (Reinke, Michalski, 1988):

Dado:

Um conjunto de eventos positivos E+ pertencentes à classe para a qual uma descrição está sendo feita, e um conjunto de eventos negativos E-, que pertencem a outras classes.

Produzir:

Uma descrição H que seja satisfeita por todos os eventos em E+ e nenhum dos eventos em E-.

Neste tipo de aprendizado o conceito já é conhecido anteriormente por parte do professor, ou existe maneira de verificar se um determinado exemplo pertence ou não ao conjunto que descreve o conceito. É considerado um método supervisionado. A tarefa do sistema é determinar a descrição geral do conceito baseado nestes exemplos e contra-exemplos.

4.1.2 Aprendizado por observação e descoberta

Neste caso, o sistema analisa entidades fornecidas ou observadas e determina se alguns subconjuntos destas entidades podem ser agrupadas em classes, i.é., conceitos. Como não existe interferência de um professor, é considerado um método não supervisionado. Os conceitos, uma vez aprendidos, podem ser nomeados e serão posteriormente aproveitados para o aprendizado de outros conceitos.

4.1.3 Método orientado por dados

Um método orientado por dados seleciona um ou mais exemplos, formula hipótese que enquadre-os, e então generaliza-a (ou especializa-a, ocasionalmente) para poder enquadrar outros exemplos.

4.1.4 Método orientado por modelo

Um método orientado por modelo inicia por uma hipótese geral qualquer, e então especializa-a (ou generaliza-a, ocasionalmente) para poder enquadrar todos os exemplos.

4.1.5 Método misto

Um método misto possui elementos dos dois métodos anteriores. Utiliza exemplo(s) para construir uma hipótese válida, testá-la e, então, modificá-la para poder enquadrar outros exemplos.

4.2 - Aquisição de Linguagem

O tema de aquisição de linguagem (Hill, 1990) tem sido historicamente abordado sob o ponto de vista do aprendizado de uma criança. Isto é, tem-se tentado simular, em computadores, o aprendizado de linguagem por parte de uma criança.

Os inatistas, uma corrente de pesquisa do aprendizado de linguagem, defendem a posição de que a linguagem é muito complexa para ser aprendida nos mesmos moldes em que o são a matemática e xadrez, por exemplo. Mesmo assim, as crianças tendem a ter adquirido sua língua nativa por volta dos cinco anos de idade, bem como conseguem ser bastante criativas no uso dessa língua, a ponto de criar estruturas novas, diferentes das já aprendidas. Chomsky (1965) defende a idéia de uma gramática universal, que seria um sistema de princípios que caracterizam a classe das gramáticas biologicamente possíveis. O uso da seqüência sujeito-verbo-objeto é um exemplo de regra que pode fazer parte dessa gramática universal.

Uma segunda corrente de pesquisa, os empiristas, acreditam que a variedade de linguagens existentes é muito grande para que estas sejam comportadas em uma

gramática universal. Para eles, a linguagem é apenas um aspecto do desenvolvimento cognitivo do indivíduo, fazendo com que o aprendizado da linguagem seja enquadrado em um quadro cognitivo muito maior do que defendem os inatistas. Os empiristas, de acordo com Piaget (Hill, 1990), "enxergam a criança, motivada por um desejo inato de comunicação, como um construtor de linguagem, auxiliada por esquemas cognitivos inatos e pelos aparatos perceptivos através dos quais todos os humanos percebem o mundo".

Outra corrente de pesquisa, a dos modelos conexionistas, é inspirada no paralelismo do funcionamento do cérebro. Estes modelos funcionam pela conexão de um grande número de pequenas unidades de processamento. Nestes modelos, também conhecidos como "redes neurais" ou "redes neuronais", cada pequena unidade de processamento, ou "neurônio", está conectada aos outros através de ligações. Parte dessas ligações serve como entrada para um "neurônio", que a utiliza para modificar suas saídas, que afetarão outros "neurônios". Esse processo tem um efeito excitatório em alguns neurônios e inibitório em outros, levando a rede a convergir para um estado final de equilíbrio. Num dos tipos de modelos, a fase de "aprendizado" da rede é supervisionada. Nela, ao ser apresentada a um exemplo, a rede ajusta os valores das conexões entre os "neurônios" para que o estado final de equilíbrio seja um estado desejado. Posteriormente, sempre que lhe são fornecidas entradas semelhantes a um dos exemplos aprendidos, a rede tende a convergir para um daqueles estados desejados. Noutro tipo de modelo, a fase de "aprendizado" não é supervisionada. Não existem estados finais desejados. A rede tende a "classificar" as entradas, de forma a convergir para um estado final que as agrupe de acordo com suas características mais marcantes (que aparecem mais vezes). Hill (1990) cita um exemplo de aprendizado do tempo passado (past tense) do verbo *ir* do Inglês (*go*) por um sistema conexionista. Nele, o sistema apresenta características semelhantes ao aprendizado de uma criança, cometendo inclusive erros semelhantes, tipo confundir *"goed"* e *"went"*.

Também têm sido feitos experimentos de aquisição de conhecimentos sintáticos através de exemplos de sentenças, ou de pares sentença-significado (Carbonell, Langley, 1990). Estas experiências mostraram a possibilidade de aquisição de regras gramaticais simples através de exemplos. Alguns outros exemplos, mesclando diversas técnicas, lograram bons resultados. Um exemplo destes é TEAM (Grosz, et al., 1987), um sistema "transportável" de interface em linguagem natural para acesso a banco de dados. Através de um interface amigável, ele permite a aquisição do vocabulário necessário ao acesso a novas bases de dados que forem anexadas ao sistema. As regras gramaticais são previamente definidas. O sistema não faz nenhuma inferência, nem gera regras novas.

Praticamente todo o aprendizado é semântico, no sentido em que relaciona as palavras ao significado correspondente (conteúdo do banco de dados) e às regras gramaticais que podem usá-las.

Parte II - O sistema

Introdução

O objetivo desse trabalho foi o de construir um sistema que possuísse as seguintes características básicas:

a) Aquisição supervisionada de conhecimento léxico, gramatical e conceitual;
b) Utilização de estruturas conceituais para representação do conhecimento;
c) Geração de estruturas conceituais episódicas a partir de textos em linguagem natural;
d) Capacidade de generalização, a partir de estruturas conceituais episódicas, gerando estruturas conceituais genéricas.

O seguinte diagrama exemplifica o funcionamento e os principais componentes do sistema.

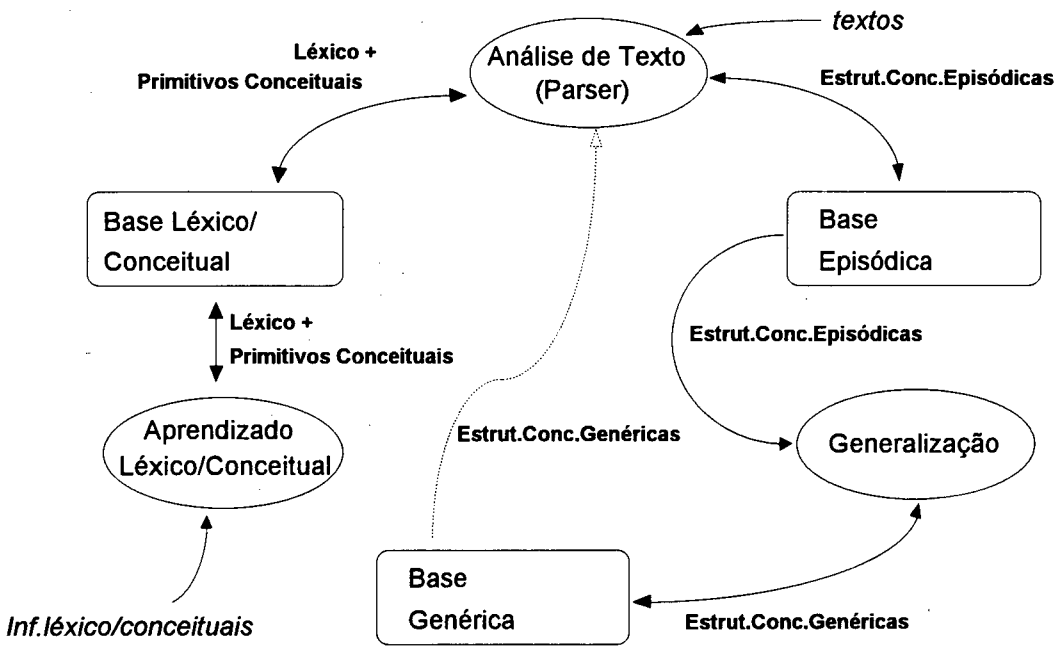


Fig. II.01 - Diagrama do Sistema

Informações léxicas (palavras) e primitivas conceituais (ex.: uso de verbos) alimentam uma base léxico-conceitual que é fonte de informações para que o analisador (parser)

possa interpretar textos e gerar estruturas episódicas. Estas, por sua vez, são utilizadas num processo de generalização progressivo, o qual gera estruturas conceituais genéricas.

A linha pontilhada demonstra uma possível utilização para estas estruturas genéricas. Nesse caso, estas informações iriam subsidiar o analisador no processo de interpretação de textos, através de inferências simples previamente armazenadas. Programas que respondem a perguntas também podem utilizar estruturas desse tipo para desempenharem seu papel.

O sistema que será discutido a partir de agora foi desenvolvido com a finalidade de aprendizado (meu aprendizado). Meu objetivo mais geral era o de pesquisar, analisar e, se possível, testar e implementar as mais diversas técnicas relacionadas à tarefa de compreensão da linguagem escrita humana. Também não era meu objetivo, conforme pode ser apurado ao longo do trabalho, esgotar o assunto, tampouco utilizar de representações semânticas com rigorismo lógico que inviabilizasse a implementação.

Sob o ponto de vista prático, pode-se utilizar estas técnicas das mais variadas formas. Especificamente, na empresa Eletrosul, existem algumas aplicações práticas possíveis para soluções desse tipo, entre as quais cito as seguintes:

- interfaces de linguagem natural para consultas a bancos de dados;
- módulo de compreensão de linguagem natural para alimentação de bases de dados de manutenções a partir de relatórios de campo.

As palavras *geralmente* e *usualmente* caracterizam bem as estruturas genéricas geradas pelo sistema. Por exemplo, após processar um ou mais textos em que um homem pague a conta de um restaurante para uma mulher, o sistema pode inferir que *geralmente* isso acontece quando um homem e uma mulher vão ao restaurante. Qualquer exceção a esta regra é aceita e, apenas gera uma estrutura diferente da primeira, apontando para mais uma possível alternativa de desfecho para o fato.

Como poderá ser visto adiante, o analisador de textos, doravante denominado *parser*, possui um funcionamento misto (sintático e semântico, simultaneamente), e grande parte dele é procedural. Sua implementação foi feita em Prolog, mas sua característica procedural foi baseada na possibilidade de uma implantação futura em uma linguagem orientada a objetos (C++, por exemplo). Entretanto, muitas partes do sistema foram desenvolvidas utilizando-se técnicas não-procedurais, principalmente por ser a alternativa mais viável em algumas situações.

Procurarei, a partir de agora, demonstrar a estrutura dos dados necessária à representação dos fatos, bem como a simbologia utilizada. O parser será discutido em suas características e o processo de generalização será detalhado. Posteriormente, será demonstrado o funcionamento do sistema através de uma seqüência de telas comentadas. Por último, farei considerações sobre possíveis futuros campos de pesquisa que dêem continuidade a esse tipo de trabalho e discutirei minhas conclusões sobre o trabalho.

5 Simbologia e Estrutura dos Dados Adotadas

5.1 Primitivos

A estrutura de dados utilizada pelo sistema é bastante simples. Consiste de uma rede semântica com poucos primitivos, cujo objetivo é dar maior liberdade de representação e menor complexidade ao programa gerador. Doravante, os diagramas serão representados pela combinação dos seguintes primitivos:

- Classe
- Ação
- Instância
- Pertinência
- Atributo
- Seqüência

5.1.1 Classe

Identifica qualquer conceito, concreto ou abstrato, que se queira representar, exceção feita às ações (veja a seguir). É representado graficamente por um retângulo:

Fig. 5.1 - Representação gráfica de uma classe.

Nela, *rótulo* identifica o nome da classe, e sempre está no singular.

5.1.2 Ação

Representa, normalmente, a parte principal dos fatos. Identificado sempre pelos verbos, é representado graficamente por um retângulo arredondado:

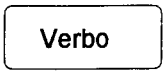


Fig. 5.2 - Representação gráfica de uma ação

Nesse caso, *verbo* está sempre no infinitivo.

5.1.3 Instância

Representa a existência concreta de uma ocorrência de uma classe, ou de uma ação. Sua representação gráfica é a seguinte:

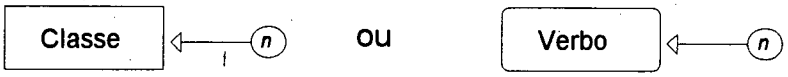


Fig. 5.3 - Representação gráfica de uma instância.

Nesse caso, a instância é representada por um círculo, com um número dentro (*n*), ligado a uma classe, ou ação, através de uma relação de pertinência (é-um).

5.1.4 Pertinência (é-um)

Representada por um arco com uma seta vazia, indica as relações entre as classes (herança) e entre instâncias e classes, ou ações. A figura a seguir exemplifica os usos dessa relação:

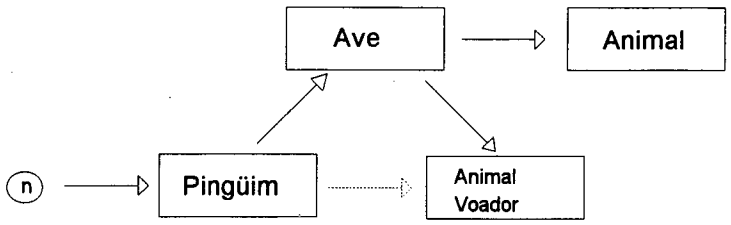


Fig. 5.4 - Representação gráfica da relação de pertinência (é-um).

Na figura, representamos uma instância da classe *'pingüim'*, que, por sua vez, pertence ao conjunto das *'aves'*, que, por sua vez, pertencem aos conjuntos de *'animais'* e *'animais voadores'*. A seta pontilhada indica que *'pingüim'* não é um *'animal voador'*. A seguir, temos os tipos de relação de pertinência, em função do grau de certeza da relação:

Símbolo	Relação
	é-um (afirmação +)
	não-é-um (afirmação -)
	talvez-seja-um (dúvida)[5]

5.1.5 Atributo

Todas as classes, ações e instâncias possuem atributos. Estes são complementos que melhor descrevem e identificam cada uma delas. Todos os atributos são representados nas formas a seguir:

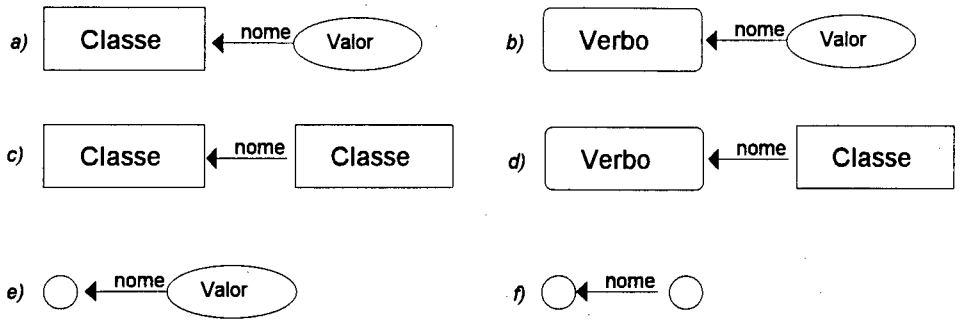


Fig. 5.5 - Representação gráfica de atributos.

A representação do atributo é feita usando-se uma seta cheia e nomeada. A seguir, uma descrição dos casos:

a) Na representação dos atributos de uma classe, os valores são opcionais. Devem ser preenchidos apenas nos casos em que são considerados como padrão. O exemplo a seguir demonstra essa distinção, pois pode-se considerar 'H' como valor padrão para 'sexo' de um homem, enquanto que 'nome' não pode possuir valor padrão:

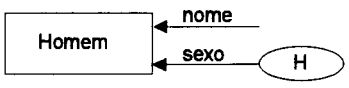


Fig. 5.6 - Atributos de classes.

---

[5]O sistema trata esta relação como uma negação até o momento em que obtém uma confirmação para a pertinência. A partir desse momento ela se transforma em uma relação *é-um*.

b) Uma ação pode possuir um atributo com valor associado. Isso acontece, principalmente, nos casos em que necessitamos representar os modificadores das ações, como a seguir:

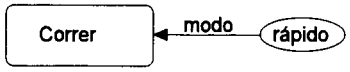


Fig. 5.7 - Atributos de ações.

c) Existem casos em que determinadas classes são atributos de outras. Isso não é comum, pois a maioria das relações entre classes se faz através de suas instâncias. Entretanto, quando formos tratar da generalização dos fatos (ver 9), esse recurso será importante. Por exemplo:

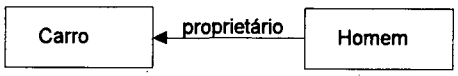


Fig. 5.8 - Classes que são atributos de classes.

d) As ações também possuem classes como atributos e, estas, normalmente, apontam para as classes que poderão desempenhar o papel explicitado pelo atributo. Por exemplo:

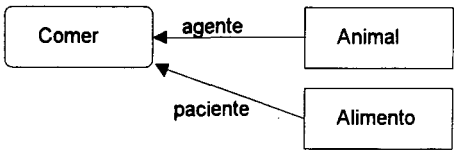


Fig. 5.9 - Classes que são atributos de ações.

Essas relações indicam as regras gerais de uso dos verbos. Excepcionalmente, podemos encontrar casos como a seguir:

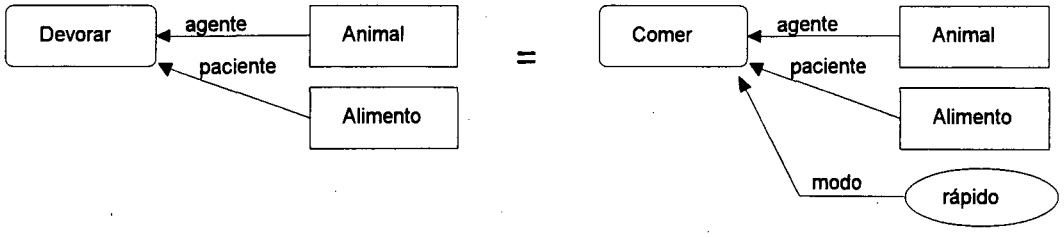


Fig. 5.10 - Regras gramaticais

e) Normalmente, as instâncias possuem valores associados a seus atributos. O exemplo a seguir demonstra isso:

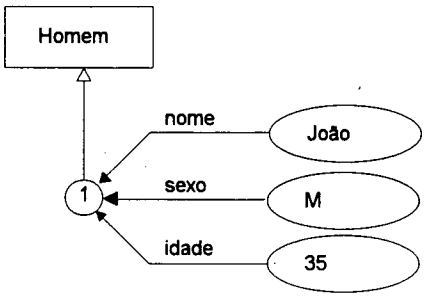


Fig. 5.11 - Atributos de instâncias

f) As interrelações concretas, existentes entre instâncias de classes e ações, também são representadas através de atributos. A seguir, alguns exemplos:

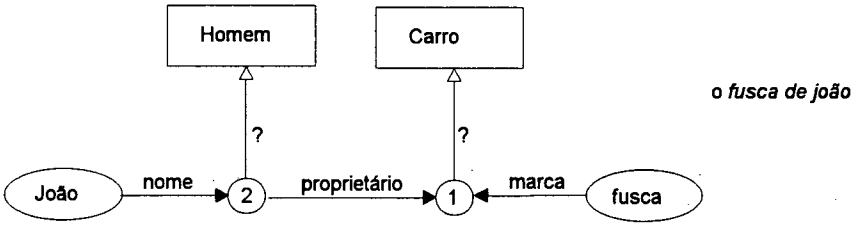


Fig. 5.12 - Interrelação entre instâncias - ex. 1.

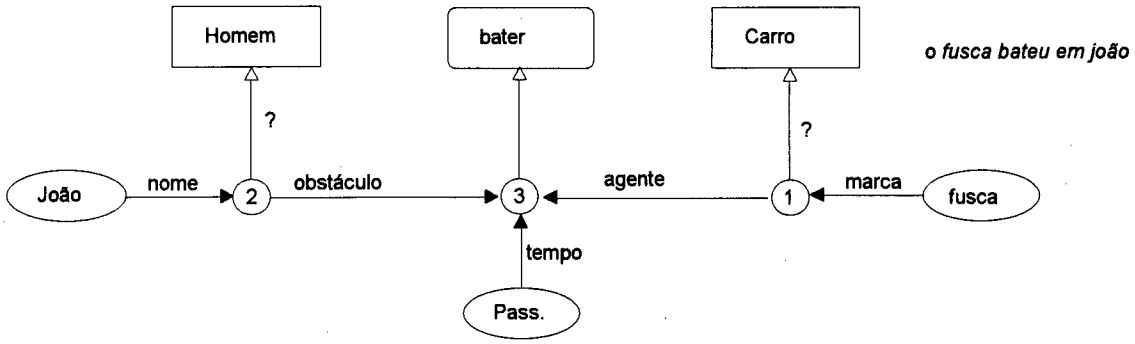


Fig. 5.13 - Interrelação entre instâncias - ex. 2.

Note que nos exemplos das figuras 5.12 e 5.13, o ponto de interrogação indica uma provável relação de pertinência, uma vez que é assumido que *'João'* é um *'homem'* e *'fusca'* é um *'carro'*. Essas relações só passam a ser positivas após confirmação do fato.

Da mesma forma que as relações de pertinência, os atributos também podem ter três graus de certeza:

Símbolo	Certeza
→	SIM
⇢	NÃO
?→	TALVEZ (sujeito a confirmação positiva/negativa)

5.1.6 Seqüência

A maioria das interrelações existentes entre ações será representada usando-se atributos. Entretanto, ao analisarmos textos mais longos, notamos que existe um tipo de relação que é somente temporal, dando a seqüência das ações. Prevendo a necessidade de uma representação especial, para a notação simplificada (veja em Generalização), definimos uma seta tracejada para indicar esta ordenação, ou seqüência de ações.

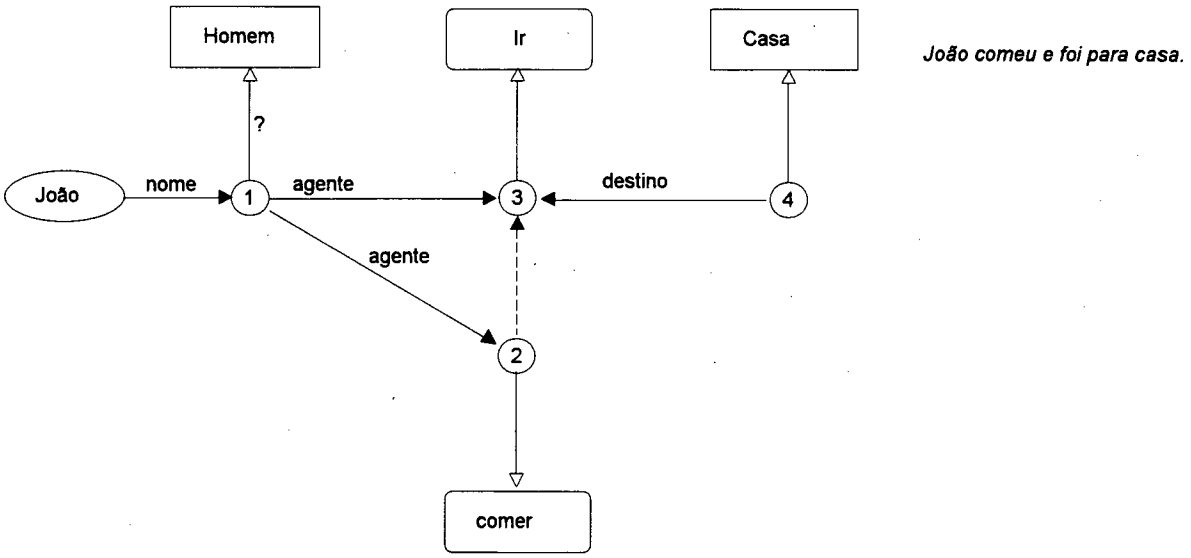


Fig. 5.14 - Seqüência de ações.

5.2 Heranças e Exceções

O sistema comporta heranças de atributos e valores, bem como situações de exceção nessas heranças. As heranças sempre se fazem no sentido oposto ao das setas de pertinência (é-um) explicitadas no diagrama. Valores atribuídos a classes são considerados como padrões (default) nos processos de herança.

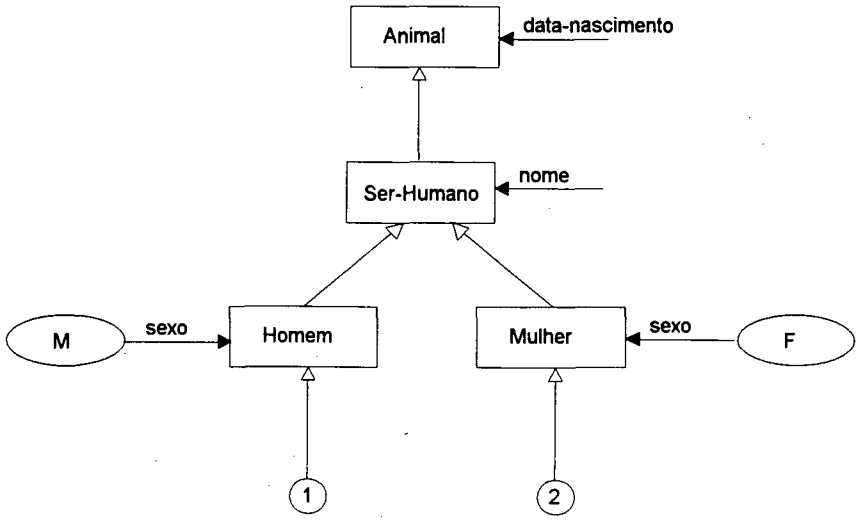


Fig. 5.15 - Herança de atributos e valores.

Assim, as instâncias herdam os atributos e valores de suas mães, avós, bisavós, etc. As instâncias citadas na figura 5.15 possuem os seguintes atributos e valores:

Instância	**Atributo**	**Valor**
1	data-nascimento	?
	nome	?
	sexo	M
2	data-nascimento	?
	nome	?
	sexo	F

Reproduzindo aqui o exemplo da figura 3.6, agora com atributos, temos:

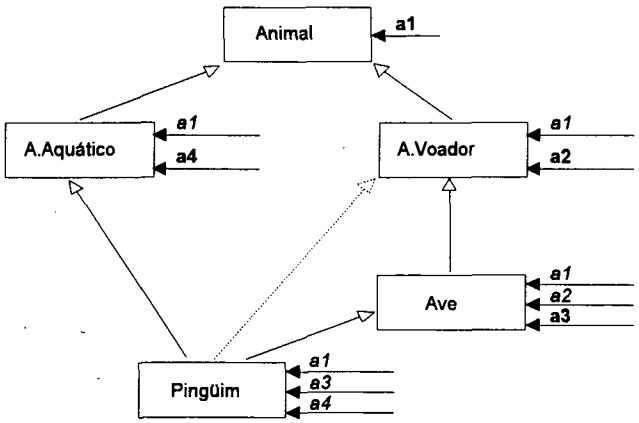


Fig. 5.16 - Herança de atributos com exceção.

Conforme mencionado anteriormente, após a identificação das relações é-um e não-é-um pertinentes ao caso, faz-se a herança apenas das relações positivas que não possuam relação negativa, equivalente, de caminho mais curto. Assim, os atributos de *'pingüim'* são *'a1'*, *'a3'* e *'a4'* (*'a2'*, que é atributo próprio de *'animal-voador'*, não é herdado por *'pingüim'*).

5.3 Fatos

A combinação dos elementos descritos anteriormente permite a representação de fatos interpretados nos textos, doravante denominados *fatos episódicos*, em contraste com os fatos oriundos do processo de generalização destes fatos episódicos, doravante denominados *fatos genéricos*. Veremos, a seguir, alguns exemplos de fatos representados através da simbologia descrita:

5.3.1 Exemplo 1

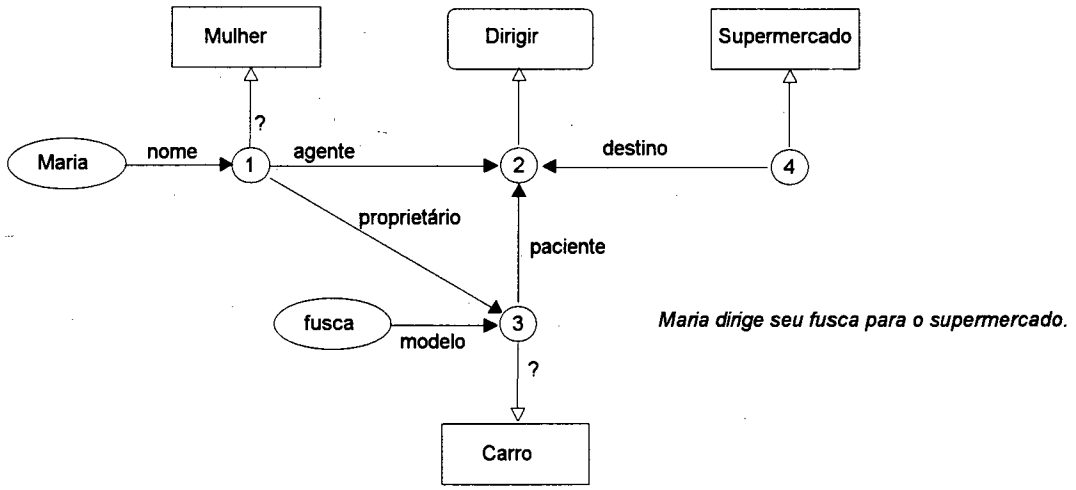


Fig. 5.17 - Exemplo no. 1 de representação de fato.

5.3.2 Exemplo 2

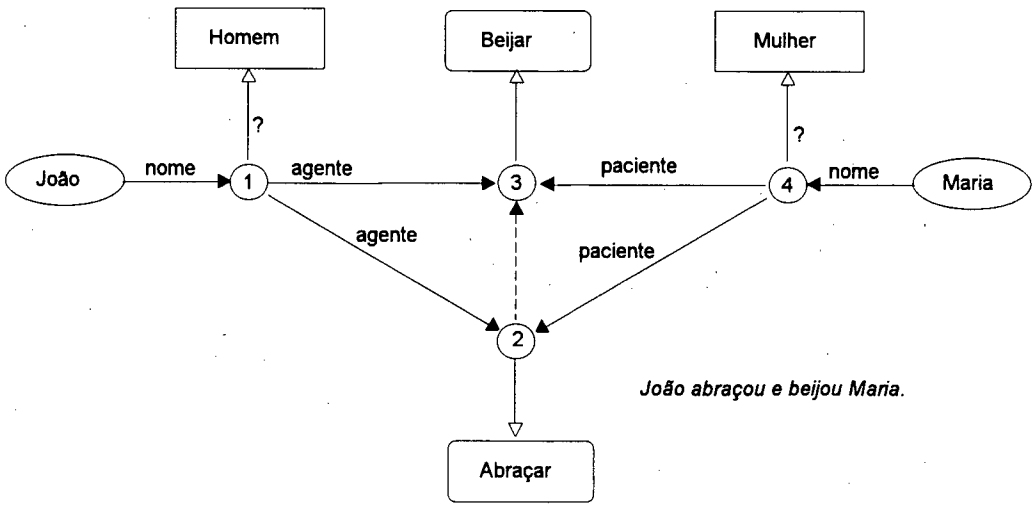


Fig. 5.18 - Exemplo no. 2 de representação de fato.

5.3.3 Exemplo 3

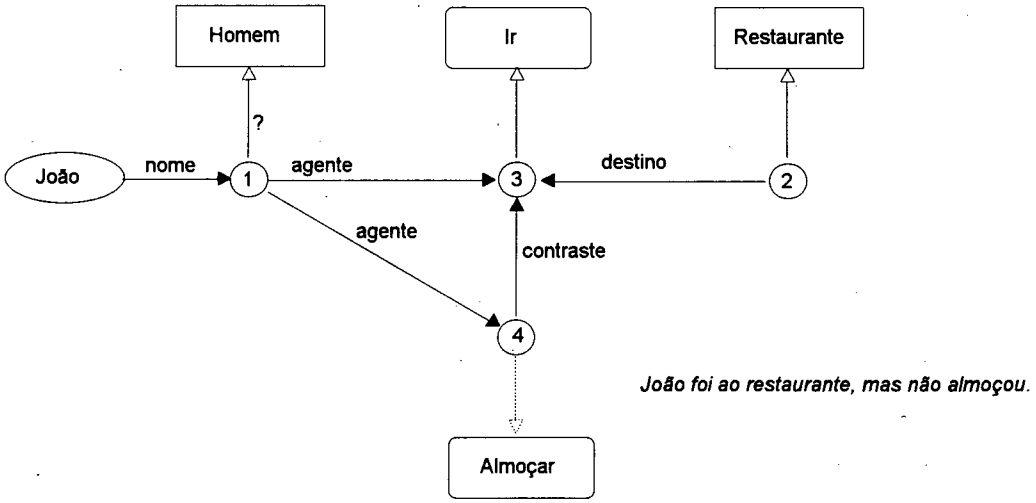


Fig. 5.19 - Exemplo no. 3 de representação de fato.

5.3.4 Exemplo 4

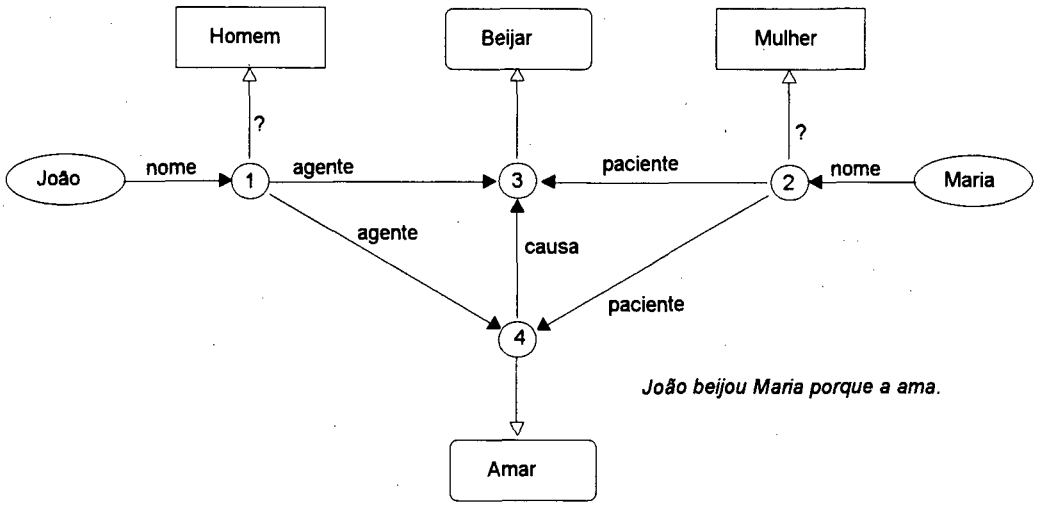


Fig. 5.20 - Exemplo no. 4 de representação de fato.

6 A base de dados

6.1 A base de classes e de fatos

A implementação física do sistema foi feita em Prolog. Todas as estruturas de dados mencionadas até agora foram implementadas como *fatos* Prolog, e serão discutidas a seguir.

6.1.1 Classes, sub-classes e instâncias

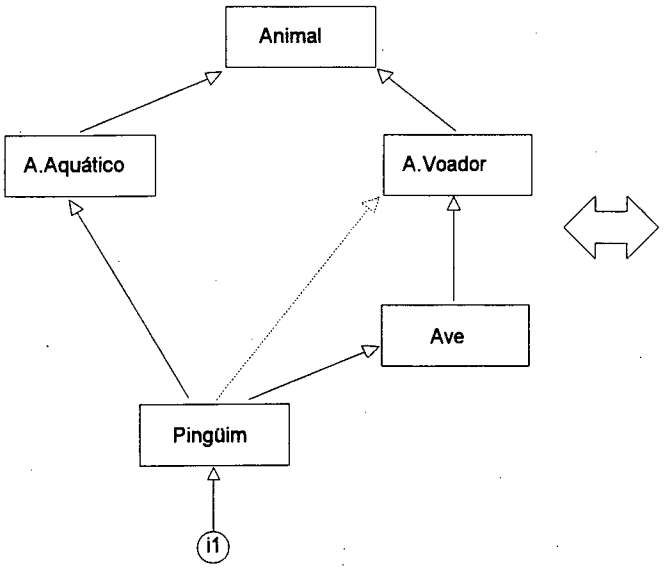
A representação de classes, subclasses e instâncias é feita das seguintes formas:

isa(**Subclasse, Superclasse, Sinal**) ou
isa(**Instância, Classe, Sinal**)

onde,

- **Subclasse** indica a classe *filha*. **Superclasse** indica a classe *mãe*. Se **Subclasse** não possui *mãe*, então **Superclasse** é igual a *nil*;
- **Instância** contém um apontador para uma instância definida em memória;
- **Sinal** indica se a relação é positiva (1), negativa (-1) ou incerta (0).

A seguir, temos um exemplo do uso dessa notação:




```
isa(animal,nil,1).
isa(animal_aquatico,animal,1).
isa(animal_voador,animal,1).
isa(ave,animal_voador,1).
isa(pinguim, ave,1).
isa(pinguim,animal_aquatico,1).
isa(pinguim,animal_voador,-1).
isa(i1,pinguim,1).
```

Fig. 6.1 - Implementação de classes e instâncias.

6.1.2 Atributos

Os atributos das classes são implementados da seguinte forma:

isatrib(**Atributo, Classe, Sinal**)

onde,

- **Atributo** indica o nome do atributo;
- **Classe** identifica a classe à qual o atributo está relacionado;
- **Sinal** explicita se **Atributo** é (+1), ou não é (-1), atributo da classe. Isso se deve em função da não-monotonicidade da representação. Um atributo que, a princípio, está relacionado a uma classe, posteriormente, em função de uma exceção qualquer, pode ter o seu sinal revertido. Esses casos devem estar explicitados para que, num processo de herança posterior, não sejam novamente associados à classe, de forma indevida.

Com base na figura 5.16, temos os seguintes fatos Prolog para descrever os atributos das classes:

```
isatrib(a1,animal,1).
isatrib(a2,animal_voador,1).
isatrib(a3,ave,1).
isatrib(a4,animal_aquatico,1).
```

Observe que o sistema se limita a registrar os atributos específicos de cada classe. A totalidade dos atributos de uma classe, ou instância, é obtida somente após o processamento das heranças envolvidas.

6.1.3 Valores

A representação de valores é feita através do seguinte fato Prolog:

$$\text{isval}\,(\textbf{Valor, Atributo,}\left\{\begin{matrix}\textbf{Instância}\\ \textbf{Classe}\end{matrix}\right\}\textbf{, Tempo1, Tempo2, Sinal})$$

onde,

- **Valor** pode ser um número, texto, um conjunto qualquer de caracteres com algum significado, ou a identificação de uma outra instância qualquer.
- **Atributo**, **Instância** e **Classe** já foram definidos anteriormente.

- **Tempo1** e **Tempo2** indicam o período de tempo em que o fato **isval** é válido.
- **Sinal** indica se é uma relação verdadeira (+1) ou falsa (-1).

O exemplo a seguir demonstra o uso desse artifício:

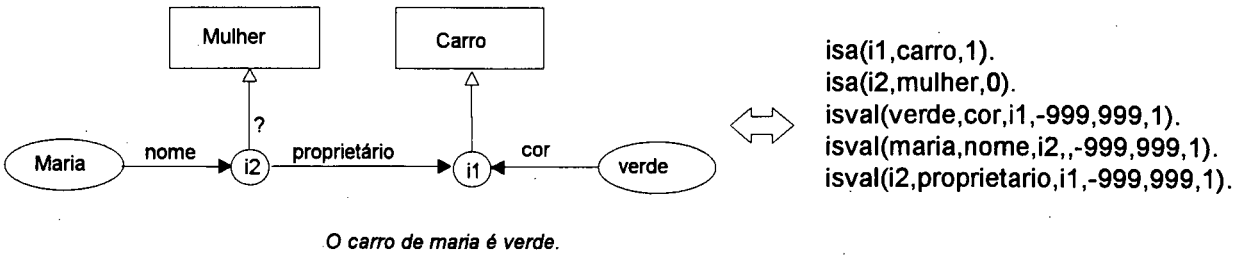


Fig. 6.2 - Implementação de atributos valorados.

O uso de *-999* até *+999* como período de tempo indica que, até informação em contrário, os fatos mencionados são verdades permanentes. O uso de *tempo1* e *tempo2* permite que se represente mudanças de valores, as quais são comuns em textos em linguagem natural. Examinando a figura 6.03,

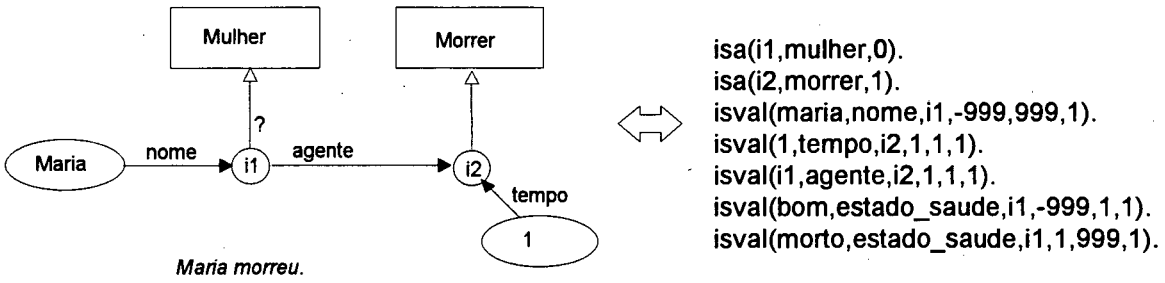


Fig. 6.3 - Representando mudanças de estado.

Este exemplo demonstra que, apesar de não estar representado no diagrama, o atributo *'estado_saúde'* pode ser inferido a partir do evento *'morrer'*, desde que o sistema esteja preparado para essa atividade. O poder semântico fornecido por este tipo de recurso é muito grande. Qualquer evento que aconteça após o evento *'morrer'* (tempo=1), envolvendo *'Maria'*, e que exija bom estado de saúde por parte dela, pode ser questionado semanticamente pelo sistema.

Tal recurso de inferência não foi implementado no sistema objeto desta dissertação, pois extrapolava o escopo definido originalmente. Entretanto, a estrutura definida prevê este tipo de situação, a qual pode posteriormente ser implementada.

6.2 Léxico

O léxico do sistema é composto por itens que são implementados através do seguinte fato Prolog:

lex(**Palavra, Tipo, Op1, Op2, Op3, Op4, Op5, Op6**)

onde **Palavra** pode ser uma palavra simples ou composta e, **Tipo** pode assumir os seguintes valores:

- **sub** - substantivo
- **adj** - adjetivo
- **adv** - advérbio
- **vb** - verbo
- **con** - conetivo
- **art** - artigo
- **prp** - preposição
- **cj** - conjunção
- **prn** - pronome

e **Op1,...,Op6**[6] dependem da classe gramatical da palavra informada.

6.2.1 Substantivos

Os substantivos são representados pelo seguinte fato Prolog:

lex(**Palavra,** sub, **Número, Gênero, Classe, Atributo,** nil, nil)

onde,

- **Palavra** contém o substantivo que se quer armazenar no léxico;
- **Número** indica se o substantivo está no singular (s), ou no plural (p);
- **Gênero** indica se o substantivo é masculino (m), feminino (f), ou indefinido (i);
- **Classe** indica se o substantivo lembra, usualmente, alguma classe. Por exemplo, *'McDonald's'* lembra a classe *'restaurante'*;

---

[6]Apesar de existirem estruturas mais elegantes em Prolog para representar cada classe gramatical, optou-se por esta em função de viabilizar o máximo possível uma futura implementação em linguagem procedural.

- **Atributo** indica, quando possível, o atributo, da classe, associado ao substantivo. Por exemplo, *'McDonald's'* pode ser associado ao atributo *'nome'* da classe *restaurante.*

6.2.2 Adjetivos

Os adjetivos são representados pelo seguinte fato Prolog:

lex(**Palavra,** adj**, Número, Gênero, Atributo, Valor,** nil, nil)

onde,

- **Palavra** contém o adjetivo que se quer armazenar no léxico;
- **Número** indica se o adjetivo está no singular (s), ou no plural (p);
- **Gênero** indica se o adjetivo é masculino (m), feminino (f), ou indefinido (i);
- **Atributo** indica o atributo normalmente associado ao adjetivo. Por exemplo, *'amarelo'* pode ser associado ao atributo *'cor';*
- **Valor** indica o valor que deve ser armazenado quando detectado a presença do adjetivo. Por exemplo, o adjetivo *'alto'* pode ser associado ao atributo *'estatura'* com valor *'>1,80m'.*

6.2.3 Advérbios

Os advérbios são representados pelo seguinte fato Prolog:

lex(**Palavra,** adv, nil, nil**, Atributo, Valor,** nil, nil)

onde,

- **Palavra** contém o advérbio que se quer armazenar no léxico;
- **Atributo** indica o atributo que se deve associar a um verbo em função da presença do advérbio. Por exemplo, *'rapidamente'* pode ser associado ao atributo *'velocidade';*
- **Valor** indica o valor que deve ser armazenado quando detectado a presença do advérbio. Por exemplo, o advérbio *'rapidamente'* pode ser associado ao atributo *'velocidade',* com valor *'veloz'.*

6.2.4 Verbos

Os verbos são representados pelos seguintes fatos Prolog:

lex(**Palavra,** vb, inf, nil, nil, nil, **Radical, Infinitivo**) **ou**
lex(**Palavra,** vb, **Tempo, Pessoa, Número,** nil, nil, **Infinitivo**)

onde,

- **Palavra** indica o verbo que se quer armazenar no léxico;
- No primeiro caso, **Radical** informa o radical do verbo, usado para se fazer conjugações automaticamente e, **Infinitivo** indica a forma infinitiva do verbo;
- No segundo caso, **Tempo** informa o tempo do verbo (presente, pretérito perfeito, pretérito imperfeito, pretérito mais que perfeito, futuro, futuro do pretérito, particípio passado, gerúndio - modos imperativo e subjuntivo, bem como tempos compostos, não foram contemplados). **Pessoa** indica qual a pessoa associada ao verbo (1ª, 2ª e 3ª), **Número** indica se é singular ou plural e, **Infinitivo** indica qual a forma infinitiva do verbo.

6.2.5 Conetivos

Os conetivos são representados pelo seguinte fato Prolog:

lex(**Palavra,** con, nil, nil, nil, nil, nil, nil)

onde a **Palavra** indica o conetivo que deve ser armazenada no léxico.

6.2.6 Artigos

Os artigos são representados pelo seguinte fato Prolog:

lex(**Palavra,** art, **Tipo**, **Gênero, Número,** nil, nil, nil)

onde,

- **Palavra** indica o artigo que deve ser armazenado no léxico;
- **Tipo** indica se o artigo é definido (d) ou indefinido (i);
- **Gênero** indica se é masculino (m) ou feminino (f);
- **Número** indica se está no singular (s) ou no plural (p).

6.2.7 Preposições

As preposições e as locuções prepositivas são representadas pelo seguinte fato Prolog:

lex(**Palavra,** prp, **Número, Gênero, Atributo, Singular,** nil, nil)

onde,

- **Palavra**, **Número** e **Gênero** são similares aos definidos anteriormente;
- **Atributo**, quando presente, indica o tipo de relação que pode existir entre dois objetos, desde que não esteja sendo usada junto com um verbo (transitividade indireta). Por exemplo, **de** em *"o carro **de** Maria"* indica uma relação de posse;
- **Singular** indica qual o item do léxico que é o singular da preposição.

6.2.8 Conjunções

As conjunções são representadas pelo seguinte fato Prolog:

lex(**Palavra,** cj, **Tipo, Ação, Atributo,** nil, nil, nil)

onde,

- **Palavra** é a conjunção que se quer armazenar no léxico;
- **Tipo** indica o tipo de conjunção (por exemplo, subordinativa causal);
- **Ação** indica qual ação deve receber o **Atributo**. Se for igual a ANT, o atributo vai para a ação anterior à conjunção. Se for POS, vai para a posterior;
- **Atributo** indica o atributo que vai fazer a ligação entre duas ações. Por exemplo, em *"Maria foi ao supermercado **a fim de** comprar comida"*, o uso de ***a fim de*** dá-nos a idéia de *'meta'* (atributo) que deve ser associado à primeira oração (ANT).

6.2.9 Pronomes

Os pronomes são representados pelo seguinte fato Prolog:

lex(**Palavra,** prn, **Tipo**, **Pessoa/Número, Gênero,** nil, nil, nil)

onde,

- **Palavra** indica o pronome que deve ser armazenado no léxico;
- **Tipo** indica se o pronome é pessoal, possessivo, etc.;

- **Gênero** indica se é masculino (m) ou feminino (f);
- **Pessoa** indica a pessoa dos pronomes pessoais;
- **Número** indica se o pronome está no singular (s) ou no plural (p) (para os que não são pessoais).

6.3 A gramática

Este sistema possui duas gramáticas distintas: uma, sintática, estática, que permite analisar textos de acordo com algumas estruturas sintáticas mais comuns (não todas), e que está implementada no código do programa sob a forma de regras; a outra, semântica, mutável, responsável pelo armazenamento de regras vinculadas à utilização dos verbos e dos relacionamentos que estes possuem com as demais classes gramaticais, e que está implementada junto ao léxico. Os fatos Prolog que contêm a gramática semântica são os seguintes:

isgen(**Verbo,** agente(**Classes**))
isgen(**Verbo,** paciente(**Classes**))
isgen(**Verbo,** compl(**Preposições, Classes, Atributo**))

onde,

- **Verbo** indica qual verbo está sendo definido. Contém a forma infinitiva;
- **Classes** contém uma lista das classes que são pertinentes a cada caso. Por exemplo, se quisermos definir quem são os possíveis agentes do verbo *'morrer'*, podemos definir da seguinte forma: *"isgen(morrer, agente([ser_vivo]))"*. Assim, o parser vai reconhecer que qualquer instância da classe *'ser_vivo'*, ou suas herdeiras (*'animal'*, *'ser_humano'*, etc.), pode morrer. Se, por acaso, quisermos permitir que membros da classe *'carro'* também possam *'morrer'* (sentido figurado), é só anexá-la à lista;
- **Preposições** contém uma lista de preposições que são válidas para um determinado caso de transitividade indireta. **Atributo** contém o atributo que vai fazer a ligação entre o objeto dessa transitividade e a ação propriamente dita. Por exemplo, em *"isgen(ir,compl([para,até],[local],destino))"*, está-se dizendo que o verbo *ir*, quando conjugado com as preposições *'para'* e *'até'*, espera que tenha como continuação uma instância da classe *'local'*, significando *'destino'*. Assim, a ligação pode ser feita.

Essa estrutura é bastante prática e, permite, além disso, que se resolva muitos casos de ambigüidade, comuns na língua humana. O exemplo a seguir demonstra um desses casos:

isgen(viajar,compl([de],[local],origem)) (*eu viajei **de** Florianópolis para Curitiba*)
isgen(viajar,compl([de],[veículo],meio_transporte)) (*eu viajei **de** carro para Curitiba*)

Este exemplo mostra que, um mesmo verbo, conjugado com uma mesma preposição, pode ter interpretações semânticas diferentes, em função da classe do objeto que estiver ligado à ação. Neste caso, o parser, em função da classe do objeto que vem a seguir, decide qual a interpretação correta.

6.4 Sinônimos

Qualquer sistema de compreensão de linguagem natural necessita comparar as informações que está interpretando, para poder fazer as inferências mínimas para o sucesso do processo de interpretação. Entretanto, as palavras que são sinônimos não possuem nada em comum, a não ser o significado semelhante. A representação desses casos é feita pelo seguinte fato Prolog:

sin(**Verbo1,Atributos1,Verbo2,Atributos2),** onde
Verbo*n* = *verbo,*
Atributos*n* = **[[Atributo,Valor], ...]**
Valor = { *valor* | classe([[**Atributo,Valor**], ...]) }

Assim, se quisermos fazer uma relação entre *'comer'* e *'almoçar'*, basta informar o fato:

sin(almoçar,[ ],comer,[[ocasião,almoço([ ])]])

ou seja, *'almoçar'* é o mesmo que *"comer no almoço"*. O sistema desenvolvido para esse trabalho gera essas estruturas automaticamente, a partir de exemplos que sejam fornecidos no processo de atualização. A seguir, alguns exemplos de sinônimos:

a) comer ⇔ ingerir comida
sin(comer,[ ],ingerir,[[paciente,comida([ ])]])

b) devorar ⇔ comer rapidamente

sin(devorar,[ ],comer,[[velocidade,rápida]]), ou
sin(devorar,[ ],ingerir,[[paciente,comida([ ])],[velocidade,rápida]])

c) jantar ⇔ comer à noite

sin(jantar,[ ],comer,[[horario,noite]]), ou
sin(jantar,[ ],ingerir,[[paciente,comida([ ])],[horario,noite]])

d) beber ⇔ ingerir bebida

sin(beber,[ ],ingerir,[[paciente,bebida([ ])]])

7 O Analisador (Parser)

O processo de interpretação de textos é feito por um módulo específico, aqui denominado *Parser*, e que possui embutido dentro de si a gramática sintática, conforme mencionado anteriormente. Não pode ser considerado um algoritmo poderoso, uma vez que está capacitado para analisar somente textos sintática e semanticamente mais simples. Entretanto, o poder fornecido pela gramática semântica, que é dinâmica, permite uma grande flexibilidade na interpretação de textos relacionados aos mais variados assuntos. Optou-se como estratégia de análise a de fazer a checagem da validade semântica para cada alternativa de estrutura sintática encontrada, parando-se o processo na primeira alternativa válida. Esquematicamente, o parser pode ser visto como a seguir:

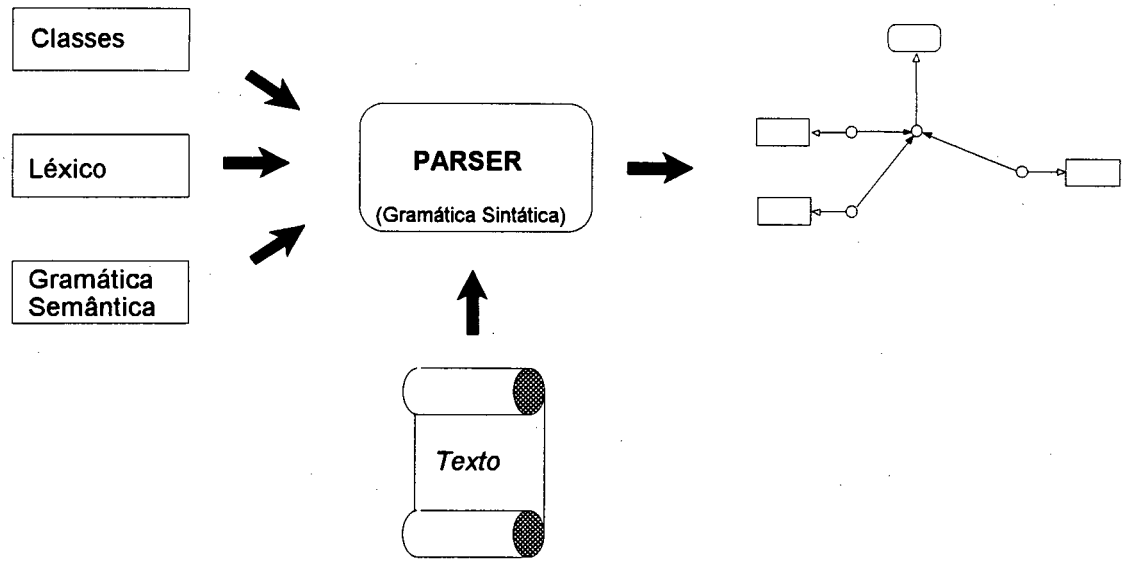
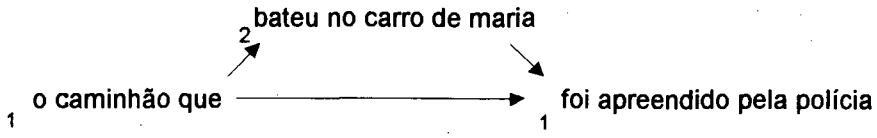


Fig. 7.1 - Esquema funcional do parser

A análise de períodos é feita por um processo de redução do texto, onde níveis são atribuídos às orações, conforme a sua localização e papel de subordinação dentro do período. Cada oração recebe um número de nível (1,2,3,...). A interpretação é iniciada pelas que possuem nível mais alto, e prossegue até chegar às de nível mais baixo. Cada oração de nível mais alto é inserida como um objeto *pendente de ligação* na oração de nível mais baixo correspondente. Este objeto *pendente de ligação* pode trazer, junto de si, informação dizendo que o agente, ou o paciente, ou ambos, da oração processada está(ão) incógnito(s). Essa informação será referenciada, doravante, como *pendência.* Por exemplo, o período *"o caminhão que bateu no carro de Maria foi apreendido pela polícia"* é tratado da seguinte maneira:



Observação: Neste exemplo, bem como em outros adiante, a conjunção *'que'* está representada separadamente do resto da oração subordinada, o que é uma incorreção gramatical. Essa representação foi escolhida para facilitar a demonstração do processo de redução do texto, onde a conjunção desempenha um papel importante de *ponte* entre as orações.

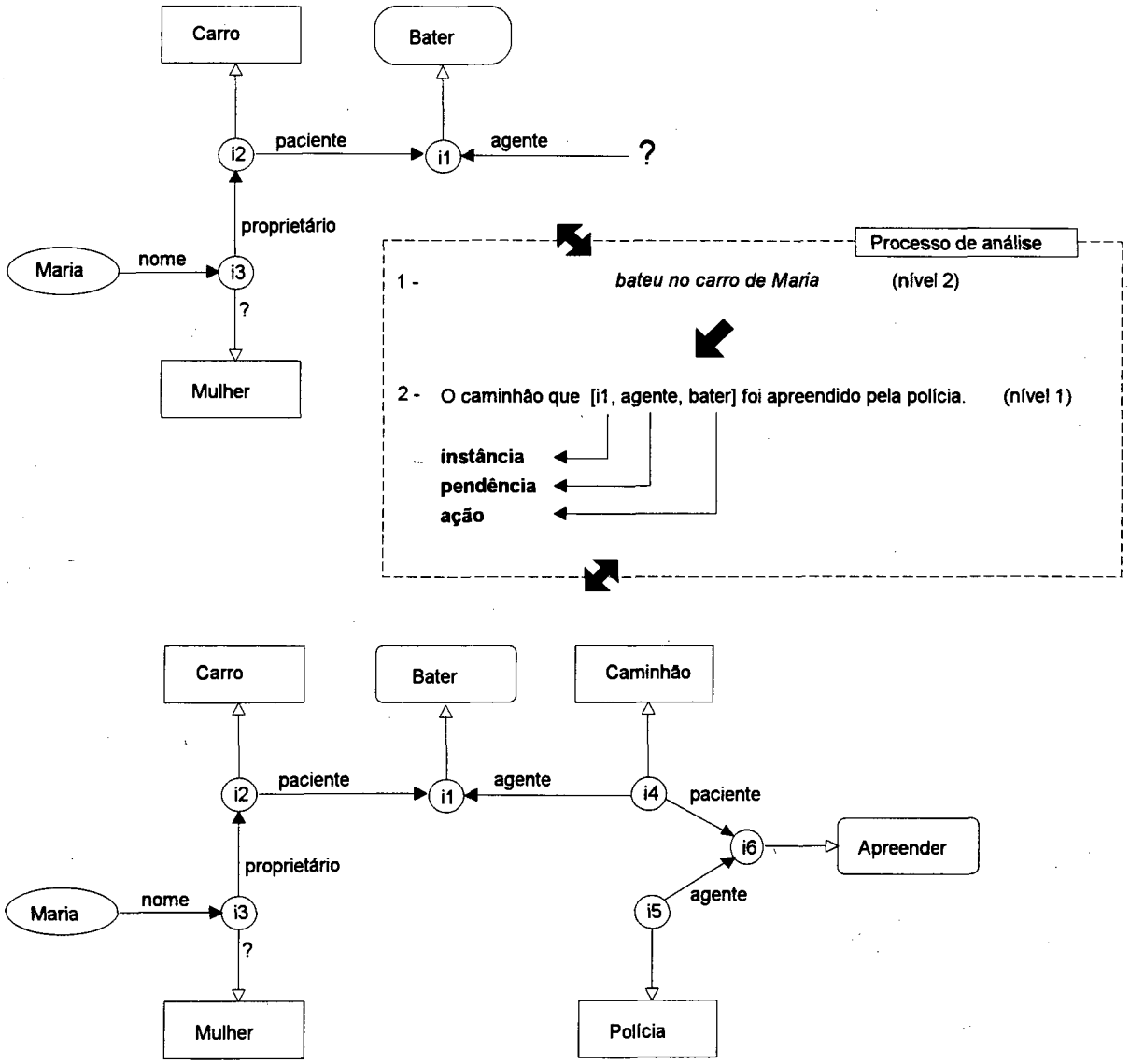


Fig. 7.2 - Exemplo de redução do período

Nesse caso, existem duas informações relacionadas a *'caminhão'*. A primeira - *"ele bateu no carro de Maria"*; a segunda - *"ele foi apreendido pela polícia"*. O processo de interpretação do período é iniciado pela oração subordinada (adjetiva) de nível mais alto (=2). A lacuna existente nessa oração, o agente não identificado, é repassada à oração de nível 1, para posterior tratamento. O objeto *'caminhão'* participa das duas orações, fazendo a ligação entre elas. A figura 7.2 exemplifica a evolução desse processo. Nela, está dito que a ação identificada pela instância *i1* não possui agente conhecido, o qual deve ser procurado na oração de nível mais baixo (*"o caminhão foi apreendido pela polícia"*). Esse processo é repetido para todas as orações do período, do nível mais alto para o mais baixo. Após o tratamento da última oração de um período, resta apenas um registro do tipo:

[*instância, pendência, verbo*],

que no exemplo acima, provavelmente, seria "[*in, nil, apreender*]".

Um período mais complexo teria tratamento semelhante, apenas com mais níveis de subordinação. Por exemplo, no período:

O ladrão, que assaltou o banco que não tinha dinheiro, foi preso pelo policial que não tinha arma,

temos a seguinte hierarquização de soluções:

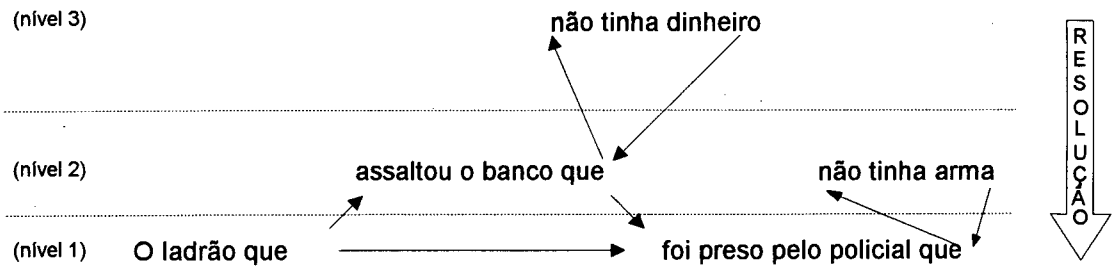


Fig. 7.3 - Exemplo de redução do período

Note que esse tipo de tratamento é necessário somente para alguns casos de subordinação (adjetivas), em que uma descrição complementar sobre um objeto é feita através de uma oração separada, e precisa ser interpretada para posterior vinculação ao objeto em questão.

Da mesma forma que os exemplos acima, alguns casos "exóticos" também podem ser corretamente interpretados, apesar de não serem bom português (poderíamos dizer que é português em notação polonesa). Vejamos o seguinte exemplo:

> *O homem que o carro que o motorista que a polícia prendeu, dirigia, atropelou, passa bem.*

Nesse caso, a hierarquização de soluções seria a seguinte:

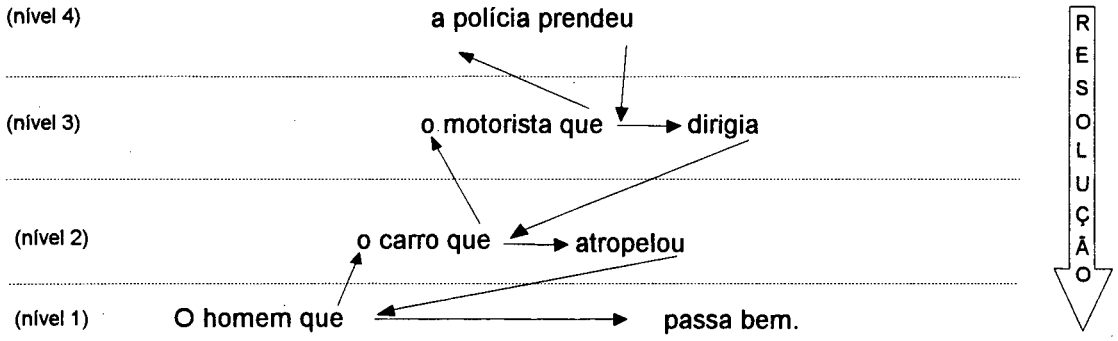


Fig. 7.4 - Exemplo de redução de um período "diferente"

ou seja, *"a polícia prendeu o motorista que dirigia o carro que atropelou o homem que passa bem"*.

Simplificadamente, o algoritmo do parser é o seguinte (os números indicam os ítens deste capítulo que descrevem cada passo do algoritmo):

1. Obter o texto e identificar as palavras;
   Para cada período do texto:
   2. Identificar as orações (principais, subordinadas, coordenadas), atribuindo-lhes níveis para resolução;
      Para cada oração identificada (das de nível mais alto para as de nível mais baixo):
      3. Identificar e criar os objetos (instâncias de classes que aparecem no texto);
      4. Identificar relações que possam existir com alguma subordinação previamente tratada (orações pendentes de nível mais alto);
      5. Identificar os adjetivos e atribuir os valores correspondentes aos objetos;

6. Identificar relações nominais existentes entre objetos, definidas pelo uso de determinadas preposições ou locuções prepositivas;
7. Identificar predicado, agentes e pacientes da ação, bem como transitividades existentes na oração;
8. Gerar estrutura semântica que represente o que foi interpretado;
9. Identificar e tratar alguns casos de anáforas;
10. Eliminar objetos redundantes e verbos de ligação (concretizando a relação atributo-objeto);

Uma das características desse parser é aceitar orações com erros gramaticais. Seu analisador sintático não bloqueia orações com erros de concordância, conseguindo interpretar corretamente a maioria das situações de erro. Sua versão inicial era bastante mais rigorosa sintaticamente, mas optou-se por inibir essa característica para melhorar o desempenho do processo. Erros léxicos (palavras escritas erroneamente) não são aceitos.

Vamos discutir agora, passo a passo, cada etapa do processo de análise e interpretação do texto.

7.1 A Obtenção do texto

A obtenção do texto pode ser feita pela simples digitação do mesmo, ou pela leitura de arquivo que contenha o texto do fato. O texto pode ter tantos períodos e orações quantos se desejar. As facilidades de edição são fornecidas pelo próprio interpretador Prolog utilizado, ou por qualquer editor de textos que gere arquivos TXT. Conforme mencionado anteriormente, não existe limitação de assuntos para os textos. Desde que as estruturas gramaticais utilizadas sejam simples e a gramática semântica esteja atualizada, qualquer tipo de texto pode ser informado. A seguir, um exemplo de texto interpretado nas fases de testes do sistema:

> *Pedro espera por Maria no McDonald's. Joana, que é garçonete, traz os sanduíches quando Maria chega. Pedro e Maria não comeram os sanduíches porque eles estavam frios. Eles saem do McDonald's e vão ao cinema a fim de assistir Don Juan de Marco.*

Ao iniciar, o parser separa o texto em palavras, tendo o cuidado de identificar possíveis locuções (Ex.: "a fim de"). O texto acima seria transformado na seguinte lista de palavras:

['Pedro', espera, por, 'Maria', no, 'McDonald"s', ptr, 'Joana', que, 'é', 'garçonete', traz, os, 'sanduíches', quando, 'Maria', chega, ptr, 'Pedro', e, 'Maria', 'não', comeram, os, 'sanduíches', porque, eles, estavam, frios, ptr, eles, saem, do, 'McDonald"s', e, 'vão', ao, cinema, 'a fim de', assistir, 'Don Juan de Marco', ptr]

7.2 Atribuição de níveis de resolução

A partir desse momento, o parser quebra o texto em períodos e começa a analisá-los, um a um. O primeiro passo é percorrer o período, da esquerda para a direita, atribuindo níveis às palavras. Isso se faz porque, conforme mencionado anteriormente, na análise de alguns casos de orações subordinadas (adjetivas), é necessário que se faça uma ligação entre objetos que aparecem na oração principal e na subordinada, motivado pela qualificação que está implícita. Essa hierarquização em níveis permite a resolução de pendências (agentes/pacientes incógnitos) entre as orações. Resumidamente, o algoritmo para esse processo é o seguinte:

1. Faz-se NIVEL = 0, e limpa-se PILHA de níveis em aberto;
2. Obtém-se PALAVRA;
3. Se PALAVRA = conjunção (no caso, *que*), ou PALAVRA = preposição seguida de verbo ou *que*, então:
   3.1 Atribui NIVEL a PALAVRA;
   3.2 Incrementa NIVEL;
   3.3 Coloca NIVEL em PILHA;
   3.4 Vai para 2;
4. Se PALAVRA = verbo então:
   Retira NIVEL de PILHA;
5. Atribui NIVEL a PALAVRA;
6. Volta para 2.

Se considerarmos o período *"Joana, que é garçonete, traz os sanduíches quando Maria chega"* como entrada, vamos ter o seguinte produto desse processo:

[['Joana',0], [que,0], ['é',1], ['garçonete',1], [traz,0], [os,0], ['sanduíches',0], [quando,0], ['Maria',0], [chega,0], ptr]

Isso faz com que os dois trechos a seguir sejam analisados separadamente, e depois ligados.

1 - ['é', 'garçonete'] e

2 - ['Joana', que, **pendência**, traz, os, 'sanduíches', quando, 'Maria', chega, ptr]

7.3 Identificação de objetos

A partir de agora, o processo é executado separadamente para cada oração. A identificação dos objetos mencionados no texto se faz pela análise dos substantivos que aparecem na oração. Alguns casos ambíguos são tratados (ex.: *morro* (substantivo - colina) vs. *morro* (verbo - morrer)). Para cada substantivo, seja ele simples ou composto, é criada uma instância da classe correspondente (vide 6.2.1). Valores padrões são herdados nesse momento. Redundâncias são criadas. Estas serão eliminadas posteriormente. Se processarmos a oração *"Pedro espera por Maria no McDonald's"*, temos o seguinte resultado:

[[obj0001,homem], espera, por, [obj0002,mulher], no, [obj0003,restaurante], ptr]

sendo que temos na base de dados os seguintes fatos Prolog:

```
isa(obj0001,homem,0).
isa(obj0002,mulher,0).
isa(obj0003,restaurante,0).
isval('Pedro',nome,obj0001,-999,999,1).
isval('Maria',nome,obj0002,-999,999,1).
isval('McDonald's',nome,obj0003,-999,999,1).
```

A estrutura definida no léxico permite a identificação automática das classes pelos atributos *nome*, ou qualquer outro que se defina. O valor correspondente indica, normalmente, uma instância de uma determinada classe. Assim, *Pedro*, normalmente, é uma instância de *homem*. Note que o sinal que vincula a instância à classe é zero, indicando a provável classe para aquela instância. Qualquer confirmação posterior faz com que a classe seja mudada, se for diferente da original, ou muda o sinal para 1, se for a mesma.

7.4 Tratamento de pendências

Conforme visto anteriormente, o processamento de orações em múltiplos níveis gera pendências, as quais precisam ser tratadas à medida em que as orações de nível mais

baixo vão sendo interpretadas. Se analisarmos o exemplo mencionado em 7.1, as seguintes orações:

1 - ['é', 'garçonete'] e
2 - ['Joana', que, **pendência**, traz, os, 'sanduíches',...],

e abstraindo-se do tratamento que é dado ao verbo de ligação *ser*, que será discutido mais à frente, podemos considerar que a análise da primeira oração gera a seguinte pendência, a qual é inserida na segunda oração:

['Joana', que, **[obj0002, agente, ser]**, traz, os, 'sanduíches',...]

A segunda oração, por sua vez, após a identificação de seus objetos, fica da seguinte forma:

[[obj0003,mulher], que, **[obj0002, agente, ser]**, traz, os, [obj0004,alimento],...]

Nesse momento, o parser procura na relação de objetos, dos que são mencionados à esquerda da pendência na oração, aquele que pode, semanticamente (conforme definido na gramática), preencher o papel estipulado na pendência. No exemplo mencionado, *obj0003* desempenha este papel. Se, por acaso, *obj0003* não pudesse desempenhar semanticamente o papel exigido, a alternativa sintática seria descartada e uma próxima alternativa, se houvesse, seria considerada. A ligação é feita através do seguinte fato Prolog:

isval(obj0003, agente, obj0002, -999, 999, 1).

A pendência é eliminada, e a segunda oração fica assim:

[[obj0003,mulher], traz, os, [obj0004,alimento],...]

7.5 Identificação de Adjetivos

O processo de redução da oração continua pela identificação e estabelecimento de atributos adicionais aos objetos, os quais, normalmente, são representados por adjetivos. Para que isso seja feito, estabelece-se o núcleo de alcance de cada objeto e, os adjetivos que se encontram dentro desse núcleo são atribuídos ao objeto em questão. Pode-se

dizer que o núcleo de um ou mais objetos termina quando é encontrado um verbo, uma preposição ou conjunção. Por exemplo, na oração *"O grande carro amarelo do pequeno homem narigudo bateu na casa branca"*, os objetos são *'carro'*, *'homem'* e *'casa'*. Os núcleos estão delimitados pela preposição *do* (de+o) e pelo verbo *'bateu'*. Assim, *'grande'* e *'amarelo'* são atributos de *'carro'*; *'pequeno'* e *'narigudo'* são atributos de *'homem'* e, *'branca'* é atributo de *'casa'*.

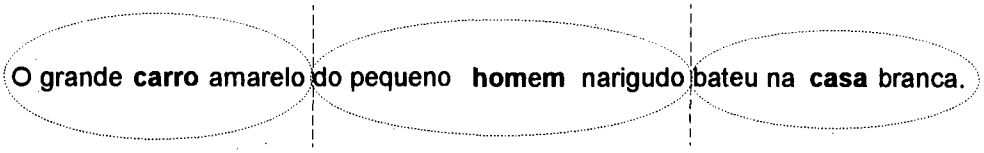


Fig. 7.5 - Identificação de adjetivos

Para cada adjetivo identificado na oração, um fato *isval* é gerado na base de dados para o objeto em questão.

7.6 Identificação de relações nominais existentes entre objetos

Assim como no caso dos adjetivos, existem relações nominais entre objetos que precisam ser tratadas. Estas relações são, normalmente, identificadas pela presença de algumas preposições ou locuções prepositivas. Somente os casos mais simples são tratados, pois as informações necessárias a um processamento mais complexo de cada caso extrapolaria o escopo do trabalho. Um exemplo é o caso da preposição *'de'*, que pode possuir diferentes significados, dependendo da natureza dos objetos envolvidos:

Sentença	**Significado**
o carro *de* Maria	propriedade
o carro *de* brinquedo	finalidade
o carro *de* plástico	material
o motor *de* carro	componente

No sistema em questão, em se tratando de relações nominais entre objetos, apenas o primeiro uso da preposição *'de'* foi aproveitado. Assim, a frase *"o carro de Maria..."*, a qual é representada da seguinte forma:

```
[[obj0001,carro], de, [obj0002,mulher],...]
```

seria reduzida a:

```
[[obj0001,carro], ...]
```

sendo que a relação de propriedade seria representada por um fato *isval*, da seguinte forma:

```
isval(obj0002, proprietário, obj0001, -999, 999, 1).
```

Assim como este, existem outros casos que merecem o mesmo tratamento, tais como: *'sob'* (embaixo_de), *'sobre'* (em_cima_de), etc.

7.7 Identificação e tratamento de predicados

O processo de tratamento dos predicados é feito de acordo com o seguinte algoritmo:

1. Eliminar advérbios da oração, salvando-os na forma de uma lista de atributos e valores;
2. Obter o predicado, bem como a voz utilizada (ativa ou passiva);
3. Obter agentes (sujeito) e pacientes (objeto direto) da ação, quando for o caso, usando as regras armazenadas na gramática;
4. Obter demais atributos da ação, usando as regras armazenadas na gramática;
5. Registrar a ação através de um fato *"isa(objxxx, Verbo, 1)"* e vincular a esta instância todos os atributos coletados em 1, 3 e 4.

Note que, apesar de parecer simples, este processo é bastante complexo, uma vez que precisa considerar todas as informações contidas na gramática, para levar a cabo a tarefa de interpretar a ação. Qualquer incompatibilidade entre o que está na gramática e o que está no texto faz com que o processo pare e, usando a facilidade de backtracking do Prolog, reinicie a partir de uma nova alternativa da gramática. Caso não exista, o sistema pesquisa a existência de homônimo, isto é, outro verbo com mesma conjugação. Se existir, reinicia o processo para este novo verbo. Caso contrário, o processo emite uma mensagem de erro avisando que a oração não pode ser gramaticalmente interpretada. Dessa forma, o poder de interpretação do sistema será proporcional à qualidade e à variedade das informações contidas na gramática. Por exemplo, para a oração *"Pedro viajou de noite, de carro, de Florianópolis para Curitiba"*, se na gramática semântica estiver registrado que:

a) *viajar* + *de* + [palavra da classe *tempo*] = horário_viagem e;
b) *viajar* + *de* + [palavra da classe *veículo*] = meio_transporte e;
c) *viajar* + *de* + [palavra da classe *local*] = origem e;
c) *viajar* + *para* + [palavra da classe *local*] = destino,

então o parser poderá interpretar corretamente a oração, gerando a seguinte rede semântica:

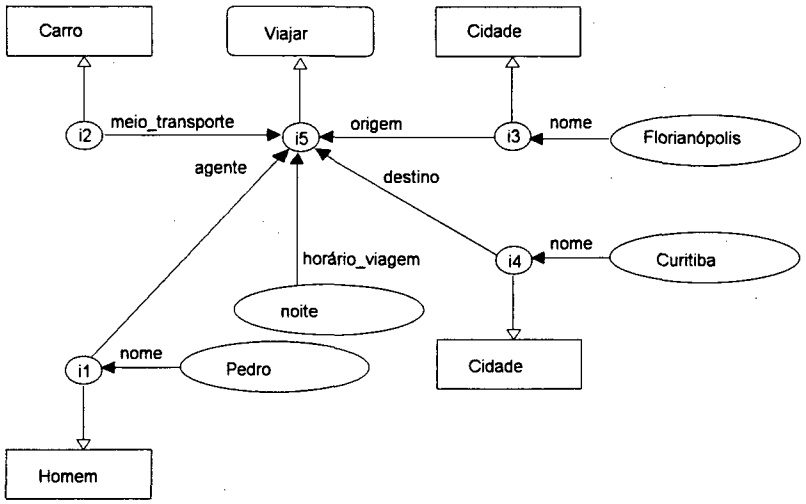


Fig. 7.6 - Exemplo de interpretação de predicado

Outro caso que exemplifica o poder da gramática é o clássico *"o navio inglês entrava o navio francês no porto de Santos"*. Nesse caso, o sistema tenta, inicialmente, interpretar a oração sob o enfoque do verbo *'entrar'*. Entretanto, a gramática para este verbo , que resumidamente é:

isgen(entrar,agente([veiculo])); e
isgen(entrar,compl([em,na,no],[local],local)),

não admite objeto direto (*"o navio francês"*). Identificado este problema, o sistema procura um homônimo para *'entrava'*, e vai encontrá-lo no verbo *'entravar'*, que possui a seguinte gramática:

isgen(entravar,agente([veiculo]));
isgen(entravar,paciente([veiculo]));
isgen(entravar,compl([em,na,no],[local],local)).

Essa gramática, por sua vez, se adapta às características da oração, permitindo a sua correta interpretação e geração da seguinte rede semântica:

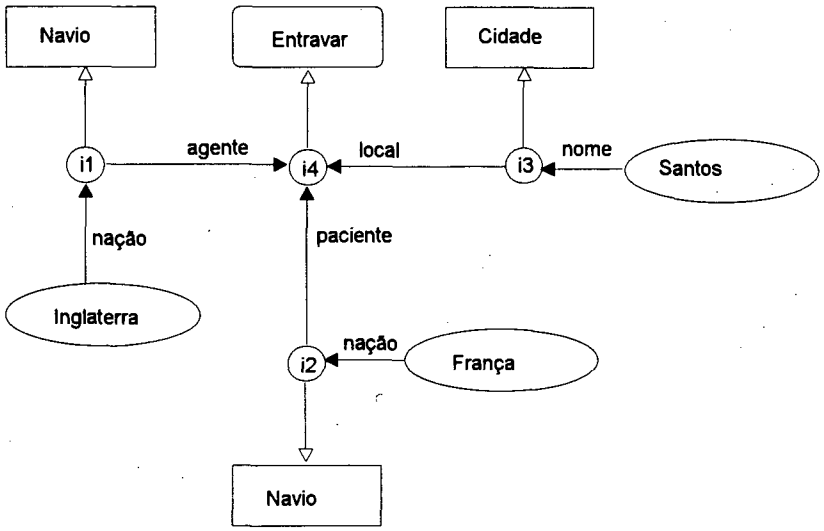


Fig. 7.7 - Exemplo de desambiguação de predicado

7.8 Geração das estruturas semânticas

Na realidade, o processo de geração da rede semântica está diluído ao longo de todo o processo de interpretação do texto. Especificamente, neste ponto é gerada a estrutura semântica relacionada ao verbo, fazendo a interligação com os demais objetos identificados no texto.

7.9 Tratamento de anáforas

Alguns casos de anáforas são tratados nesta parte do processo, a fim de permitir que textos mais *naturais* possam ser interpretados. O processo inicia quando o parser está identificando os objetos do texto. Ao identificar uma instância de uma classe desconhecida, normalmente representada por um pronome pessoal (reto ou oblíquo), o parser inclui um objeto na base de dados, vinculado a uma classe indeterminada. Por exemplo, no texto *"Pedro lanchou no McDonald's. Ele comeu e foi ao cinema."*, ao analisar o segundo período, o parser depara-se com o pronome *'ele'*. Neste momento, é gerado o fato Prolog (1) *"isa(objxxx,classe_ind,1)"*, o qual indica que o objeto em questão não possui uma classe determinada. Além dos atributos do objeto que estão explicitados na oração, o parser cadastra também um atributo para gênero (= masculino),

e um para número (= singular), para serem usados no processo de tratamento de anáforas.

Ao chegar a esta etapa do processo, o parser inicia a resolução das anáforas pesquisando o grupo de classes, associado ao papel desempenhado pelo objeto em questão, na gramática. No exemplo acima, o parser descobre que o objeto em questão é o agente da ação, e que, segundo a gramática do verbo *'comer'*, este objeto deve ser uma instância da classe *'animal'*.

Feito isso, o parser inicia uma busca de trás para frente, em direção ao início do texto, buscando um ou mais objetos cujas características sejam compatíveis com as do objeto em questão. Restrições de gênero, número e de classes têm que ser respeitadas. Identificado o objeto, ele substitui o objeto em questão. Seus atributos são complementados, através de um processo de unificação. O objeto indeterminado é eliminado da base de dados.

No exemplo acima, após saber que o objeto em questão era um *'animal'*, somente um *(ele)*, e masculino, o parser inicia a busca e encontra *'Pedro'*, o qual atende às exigências da pesquisa. Nesse momento, *'Pedro'* é colocado como agente da ação *'comer'*, e o outro objeto (indeterminado) é eliminado da base de dados.

O mesmo procedimento permite resolver outros casos, como:

- *Pedro lanchou um Big-Mac no McDonald's. Ele comeu-o e foi ao cinema;*
- *Pedro, João, Maria e Lígia foram ao restaurante. Eles fizeram o pedido enquanto elas iam ao banheiro;*
- *Maria bateu o carro. Ele foi para o ferro-velho enquanto que ela foi para o hospital.*

Neste processo, possíveis casos de ambigüidade não são resolvidos. Em textos como *"Pedro levou João ao restaurante. Ele pagou a conta"*, o primeiro objeto que satisfaz ao critério de pesquisa é aproveitado - no caso, *João* - o que não é correto.

Diversas pesquisas já foram feitas para tratar esse tipo de problema, sendo uma delas relatada em (Gordon, Grosz, Gilliom, 1993), que leva em consideração o centro de atenção em um discurso, e dá mais importância à evolução dos papéis desempenhados por um objeto ao longo do texto. Entretanto, em função do compromisso com o escopo do projeto, optou-se pela não implementação deste recurso.

7.10 Eliminação de objetos redundantes e verbos de ligação

Durante todo o processo de interpretação do texto, muita redundância é gerada. Cada objeto identificado é cadastrado na base de dados, com seus respectivos atributos, mesmo que já tenha sido identificado anteriormente. Isso ocorre porque, no momento em que está tratando um objeto, o parser ainda não possui todas as informações necessárias para julgar se ele é, ou não é, idêntico a outro objeto, já cadastrado.

Da mesma forma, os verbos de ligação (ser, estar, parecer, ...) também são cadastrados como "ações", apesar de não o serem. Por exemplo as orações: a) *"o carro é amarelo"* e b) *"Maria é a mulher que bateu o carro"*, geram as seguintes estruturas semânticas:

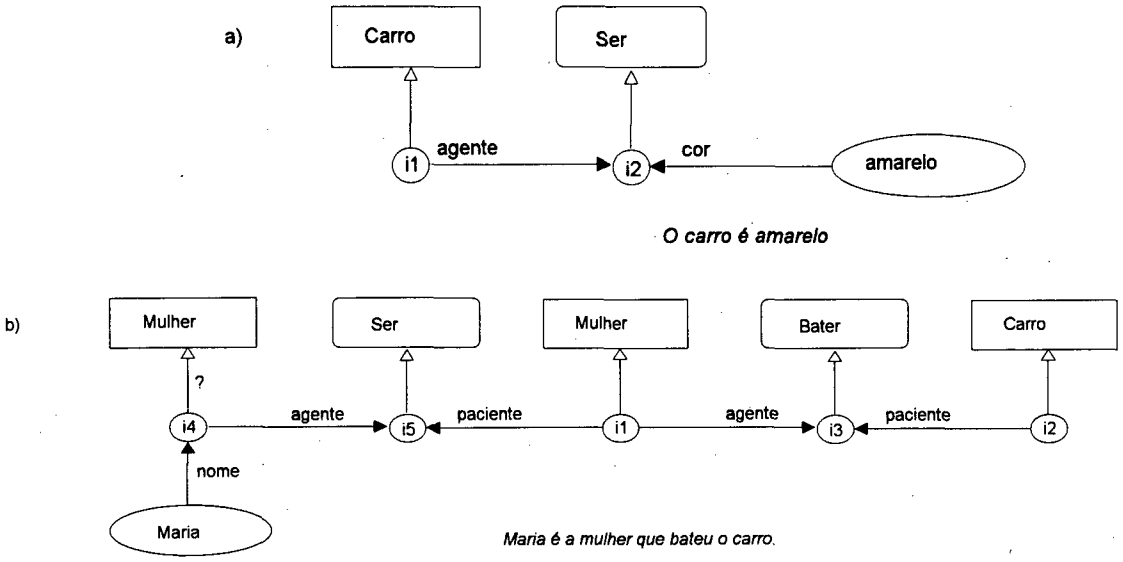


Fig. 7.8 - Redundância de objetos e verbos de ligação

Nos dois casos acima existem redundâncias que podem ser eliminadas, fazendo com que a estrutura resultante seja mais simples, e fácil de ser manipulada.

A eliminação das redundâncias e dos verbos de ligação é feita através de um processo de unificação. Nesse processo, dois os mais objetos são substituídos por somente um, o qual vai aglutinar todos os atributos dos demais. Todas as referências externas feitas aos demais objetos são, também, reorientadas para esse objeto "unificador".

No caso dos verbos de ligação, os atributos da "ação" são unificados com os do objeto a que se refere a "ação" (*'carro'* ← *'amarelo'*), ou dois objetos ligados pela "ação" são

unificados (mulher(*'Maria'*) ← mulher(*"que bateu o carro"*)). Aplicado aos exemplos citados anteriormente, esse processo geraria as seguintes estruturas semânticas:

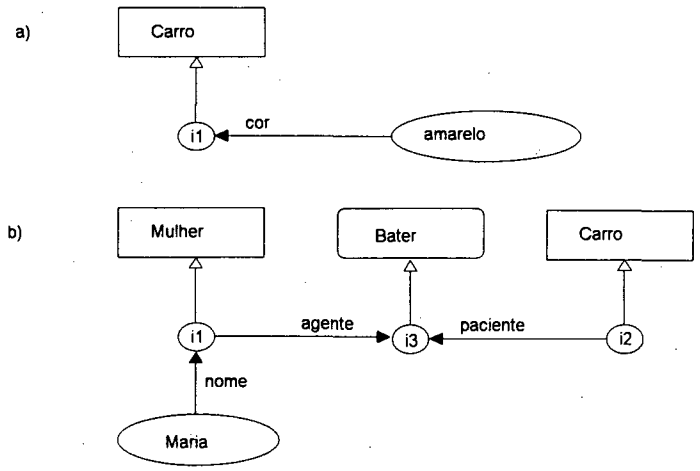


Fig. 7.9 - Estruturas semânticas sem redundâncias

No exemplo b), a dúvida sobre o fato de Maria ser uma mulher (identificado por "?" no primeiro diagrama) desapareceu durante o processo de unificação, representando a realidade expressada no texto (*"Maria é a mulher..."*).

No caso das redundâncias de objetos, existe um processo que procura todos os objetos *Yi* semelhantes a um determinado objeto *X*, para posterior unificação. Os critérios de seleção são os seguintes:

- As classes de *X* e *Yi* têm que ser compatíveis entre si, isto é, ou são iguais, ou a classe de *X* tem que ser hierarquicamente sucessora da classe de *Yi*. Por exemplo, *X* = *'homem'* e *Yi* = *'homem'*, ou *X* = *'homem'* e *Yi* = *'mamífero'*;

- Atributos usados para identificar objetos não podem possuir valores diferentes com sinais positivos. Por exemplo, o nome de uma mulher não pode ser Maria e Joana ao mesmo tempo.

- *X* tem que poder substituir *Yi* nos papéis por ele desempenhados nas diversas ações, sem gerar incompatibilidades semânticas entre as estruturas de dados geradas e a gramática;

São contadas todas as compatibilidades existentes entre *X* e os *Yi* relacionados. Com base nessa contagem, é montado um *ranking* dos mais compatíveis, usando-se um

número padrão (arbitrado) de compatibilidade (1,2,...), que representa o número mínimo de atributos que devem coincidir (parâmetro do sistema) entre *X* e cada *Yi*. Obtido esse *ranking*, é feito o processo de unificação de *X* com todos os objetos *Yi* selecionados.

8 O processo de generalização

8.1 Discutindo o problema

Conforme foi exemplificado no início desta dissertação, o conjunto de conhecimentos relacionados a um assunto que está sendo discutido é de extrema importância à tarefa da compreensão. Se imaginarmos Einstein discutindo teoria da relatividade com um aborígene da Austrália, fica bem fácil visualizar a dificuldade que um encontrará para se expressar, bem como o outro para compreender, mesmo que estejam falando a mesma língua. São mundos, conceitos, idéias, experiências diferentes e, principalmente, desconhecidas ora por um, ora por outro. Falta uma base comum de conhecimentos que permita a existência de uma efetiva comunicação.

Essa base de conhecimentos, mais conhecida como "senso comum", tem sido bastante discutida ao longo dos últimos trinta anos, por cientistas da computação. Ninguém mais duvida da sua importância. Muitos sistemas já foram desenvolvidos procurando utilizar recursos de senso comum para terem mais capacidade de interpretação.

Mas, o que é senso comum? Imaginemos a seguinte situação: um marciano chega à Terra e, depois de aprender uma de nossas línguas, sai para passear. Caminhando por uma calçada, vê um homem, trôpego, mal se agüentando em pé, vindo em sua direção com uma garrafa na mão, ingerindo seu conteúdo em grandes goles. Impressionado com o que vê, registra esse fato em sua memória. Mais tarde, compra uma garrafa idêntica e experimenta seu conteúdo. Fica tonto e, mais tarde, com uma grande dor de cabeça. Dias depois, após ter visto muita coisa e, principalmente, pessoas bêbadas brigando, provocando acidentes de trânsito, etc., o marciano pode chegar às seguintes conclusões:

- o ser humano, em geral, consome bebidas alcoólicas;
- quando isso acontece, ele costuma perder o controle sobre seu corpo, mente e ações;
- por causa disso, ele pode se envolver em brigas, acidentes. Isso pode acarretar prejuízos para si e para os outros.

Pode-se notar, com este exemplo, que o processo de aprendizado do marciano ocorreu de forma progressiva e incremental. À medida que ia presenciando, ou vivenciando, os fatos, comparava-os com outros já vivenciados, percebendo o que tinham em comum e o que tinham de diferente. Esse processo permitia a formação, na sua mente, de fatos mais

genéricos, em que os papéis poderiam ser desempenhados por qualquer um; e permitia que se estabelecesse relações entre esses fatos, também de forma genérica, as quais poderiam ser confirmadas, ou negadas, por fatos posteriores. Após experimentar inúmeras situações, o marciano já tinha certeza das suas conclusões, o suficiente para poder exprimi-las.

A esse processo chamamos de aprendizado indutivo, através de exemplos, com conseqüente generalização de conceitos. Quando esses conceitos são compartilhados por muitas pessoas, são chamados de "senso comum".

É óbvio que não pretendemos que nosso sistema tenha a capacidade de generalização do marciano do exemplo. Entretanto, procuramos dotá-lo da capacidade de, identificando seqüências de ações semelhantes, poder generalizá-las, de forma a construir, pouco a pouco, a sua (restrita) base de "senso comum".

Pode-se querer questionar isso. A resposta é que a maior parte do trabalho gasto na construção de sistemas que envolvam compreensão de linguagem natural, ou mesmo sistemas especialistas, não é gasta na elaboração do sistema propriamente dito mas, reside na construção das bases de conhecimentos necessárias ao desempenho das tarefas a que se propõem os sistemas. Se pudermos, no futuro, construir essas bases de forma automática, a partir de exemplos, então um grande passo terá sido dado.

O uso dessas estruturas genéricas, aprendidas pelo sistema, no processo de interpretação de novos textos, de forma a dar-lhe a capacidade de trabalhar por expectativa, é uma interessante área de pesquisa a ser trabalhada no futuro

8.2 Simplificando a notação utilizada

Conforme foi afirmado anteriormente, o processo de generalização deve gerar fatos em que os papéis possam ser desempenhados por qualquer um, desde que semanticamente aceito. Assim, por exemplo, se generalizarmos o conceito expressado pela oração *"João dirige o carro"*, devemos obter algo como *"homens dirigem carros"*.

Nesta forma, a maioria dos atributos dos objetos tende a desaparecer, por causa da generalização dos conceitos, onde não mais importa saber o nome, a idade ou o sexo de uma pessoa, mas sim, saber o que fez, quais as causas, conseqüências, fatos correlatos, etc.

Entretanto, em alguns casos, isso não ocorre. Na oração *"João adoeceu porque comeu rapidamente"*, o processo de generalização deve gerar algo como *"homens adoecem porque comem rapidamente"*. Note que o atributo *'rápido'* é vital para a compreensão do texto.

Assim sendo, a notação tende a ficar mais simples, sem a presença da maioria dos atributos valorados. O exemplo acima ficaria assim:

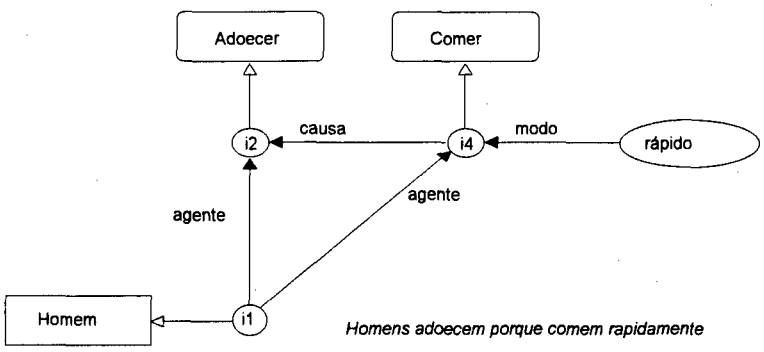


Fig. 8.1 - Rede semântica genérica

Embora, à primeira vista, não pareça uma simplificação muito grande, devemos nos lembrar que, na maioria dos casos, cada objeto desses pode possuir vários atributos, os quais deixam de acompanhá-los às estruturas genéricas.

Um pequeno acréscimo que se torna necessário neste momento é o que permite o registro da possibilidade de duas instâncias distintas serem idênticas. Isso ocorre porque o processo de generalização é dinâmico e, muitas vezes, não se tem todas as informações necessárias para julgar essa igualdade. Por exemplo, no diagrama 8.2:

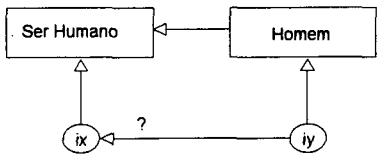


Fig. 8.2 - Unicidade de instâncias

está dito que existe a possibilidade de que *ix* e *iy* sejam idênticos, faltando apenas uma afirmação para confirmar essa suposição. Mais adiante, nos exemplos, ilustraremos melhor essa característica.

8.3 O método de generalização

O método de generalização de fatos utilizado neste sistema pode ser descrito da seguinte forma:

> *Dada uma ação A1 a generalizar, se existir uma ação A2 na base genérica, tal que A2 é complementar de A1, então gerar a ação A3 com o produto da complementação de A1 por A2. Se não existir A2, então generalizar A1 puramente, eliminando os atributos desnecessários.*

A2 é complementar de *A1* quando:

a) o verbo de *A1* é o mesmo que o verbo de *A2*;

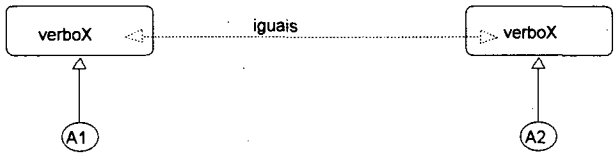


Fig. 8.3 - Ações complementares - regra 1

b) para qualquer atributo *a1* de *A1*, se ele identifica um objeto (*o1*) da classe *C1*, então *A2* tem que possuir o mesmo atributo *a1*, identificando um objeto (*o2*) de classe *C2*, sendo que *C2* é idêntico a *C1*, ou *C2* possui uma classe *C3* antecessora (mãe, avó, etc.) comum a *C1*;

A figura a seguir exemplifica essa regra. Note que as linhas tracejadas unindo *C1* e *C2* a *C3* indicam que *C3* pode ser mãe, avó, bisavó, etc., de *C1* e *C2*.

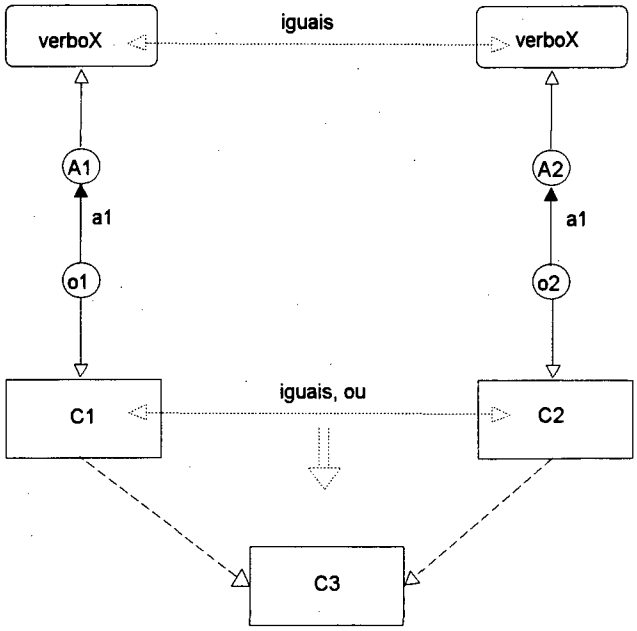


Fig. 8.4 - Ações complementares - regra 2

c) para qualquer atributo *a1* de *A1*, se ele identificar uma outra ação *Ax*, então também deve existir o mesmo atributo *a1* em *A2*, o qual deve identificar uma ação *Ay* e, *Ay* tem que ser complementar de *Ax*;

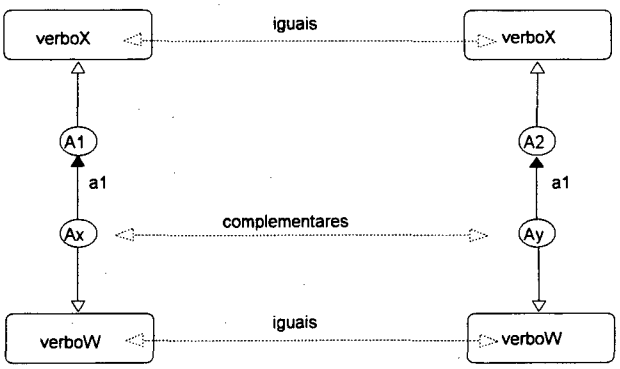


Fig. 8.5 - Ações complementares - regra 3

d) para qualquer atributo *a1* de *A1*, se ele não se enquadra nas regras *b* e *c*, então ele deve também ser atributo de *A2*, e seus valores devem ser idênticos (modificadores de ação).

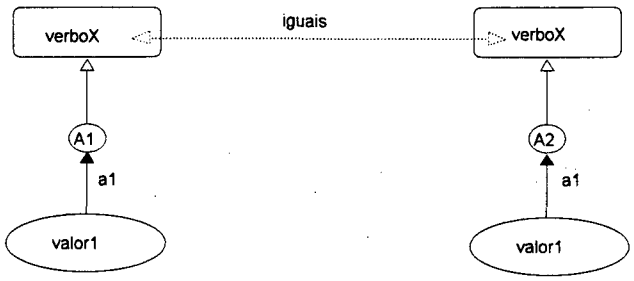


Fig. 8.6 - Ações complementares - regra 4

Nota: Atente para o fato que, até a regra *b*, todo conhecimento adquirido já estava expressado na gramática. Entretanto, a regra *c* vai identificar as relações entre as ações, enquanto a regra *d* identifica os fatores que modificam, ou caracterizam, essas relações. É aí que reside o processo de aprendizagem do sistema.

A geração de *A3*, que é a complementação de *A1* por *A2*, é feita da seguinte forma:

- o verbo de *A3* é o mesmo de *A2*;
- todo atributo que se enquadrar na regra *b*, citada anteriormente, deve apontar para *o2*, se *C1* e *C2* forem idênticas. Nesse caso, *o1* deve ser eliminado e, se ele é referenciado em qualquer atributo de outro objeto, então deve ser substituido por *o2*. Nos casos em que *C1* e *C2* são diferentes, o sistema cria um objeto (*o3*) da classe *C3*, pela unificação de *o1* e *o2*, e o associa ao atributo de *A3* em questão. Nesse caso, o sistema verifica se *o1* e *o2* podem ser eliminados. O objeto que não for referenciado será eliminado. O que for referenciado, se puder ser substituido por *o3*, será eliminado; caso contrário, ficará marcado como sendo passível de ser substituído por *o3*, quando for o caso.
- todo atributo que se enquadrar na regra *c* deve ter suas ações *Ax* e *Ay* complementadas, através da invocação recursiva do processo de generalização, gerando *Az*, que passa a ser apontado pelo atributo de *A3*;
- aproveita-se os valores dos atributos que se enquadrarem na regra *d*.

8.4 Um caso prático de generalização

Analisando as duas frases a seguir:

1) *Pedro bateu o carro*, e
2) *Maria morreu porque bateu o carro*,

temos os seguintes diagramas genéricos:

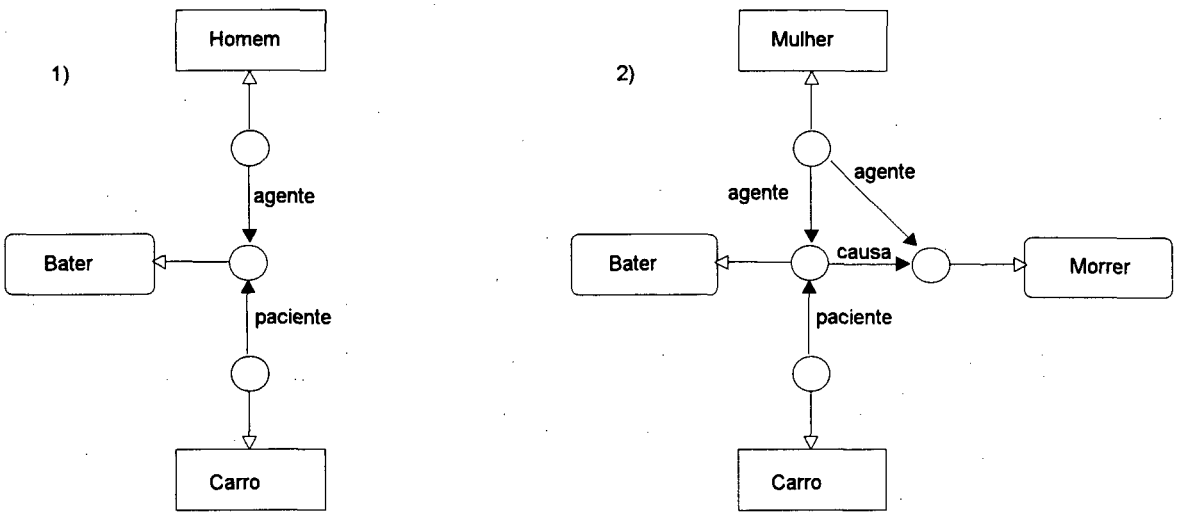


Fig. 8.7 - Exemplo de generalização - antes

São dois fatos similares, com personagens diferentes e implicações diferentes. Ao generalizar o segundo fato, o sistema verifica que *'morrer'* (ação principal) não possui ainda referência para tentativa de complementação. Dessa forma, o processo de generalização se resume a copiar os atributos da ação para a base de fatos genéricos. Entretanto, ao identificar *'bater'* como *'causa'* de *'morrer'*, o processo é repetido e, um possível caso (regra *a*) de complementação é encontrado. Aplicando a regra *b*, o sistema obtém *'ser humano'* como *'agente'* comum aos dois fatos. *'carro'* não apresenta problemas também. Ao generalizar essas informações o sistema gera informações que permitem elaborar o seguinte diagrama:

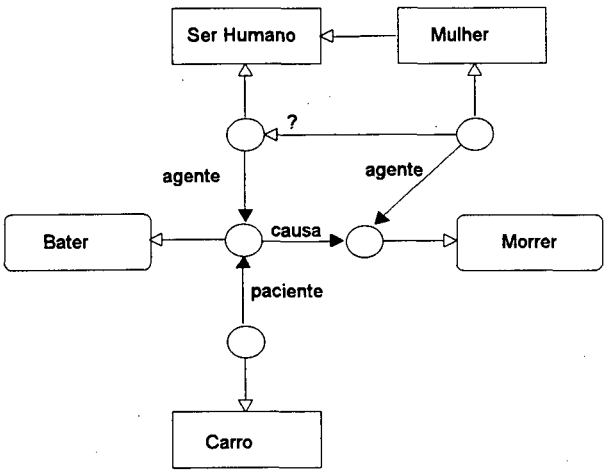


Fig. 8.8 - Exemplo de generalização - durante

Esse diagrama nos diz o seguinte: um *'ser humano'* pode *'bater'* um *'carro'*. Se esse *'ser humano'* for uma *'mulher'*, ela pode *'morrer'* por causa do fato *'bater'*. Note que o processo de generalização foi feito apenas para a ação *'bater'* o *'carro'*. Não existe afirmação que suporte uma generalização para a ação *'morrer'* tendo como causa *'bater'* o *'carro'* para um agente *'homem'* (o que permitiria generalizá-lo para *'ser humano'*). Mesmo assim, fica registrado que as duas instâncias (*'ser humano'* e *'mulher'*) podem ser uma só.

Se agora fornecermos ao sistema a seguinte afirmação: *"Pedro morreu porque bateu o carro"*, o diagrama seria transformado em:

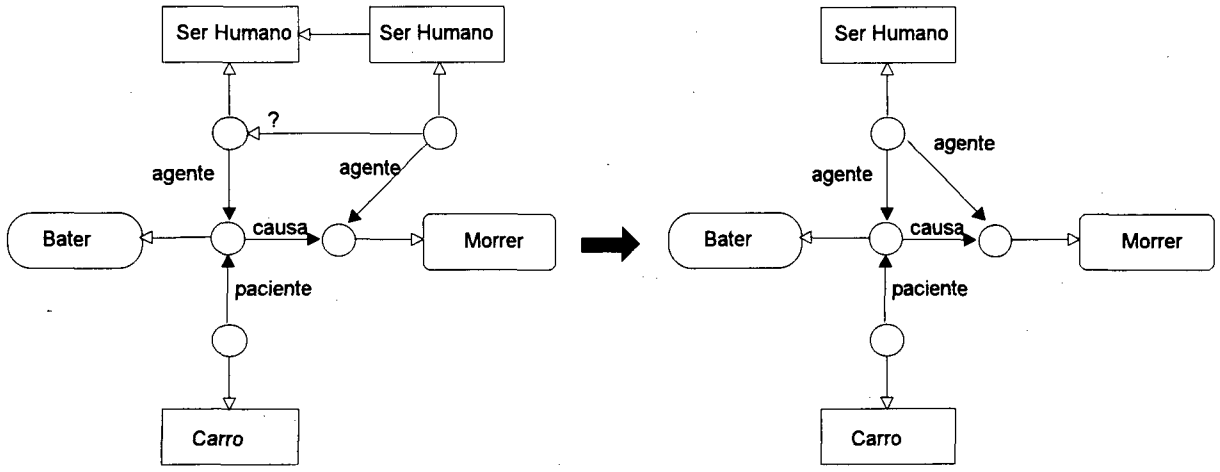


Fig. 8.9 - Exemplo de generalização - depois

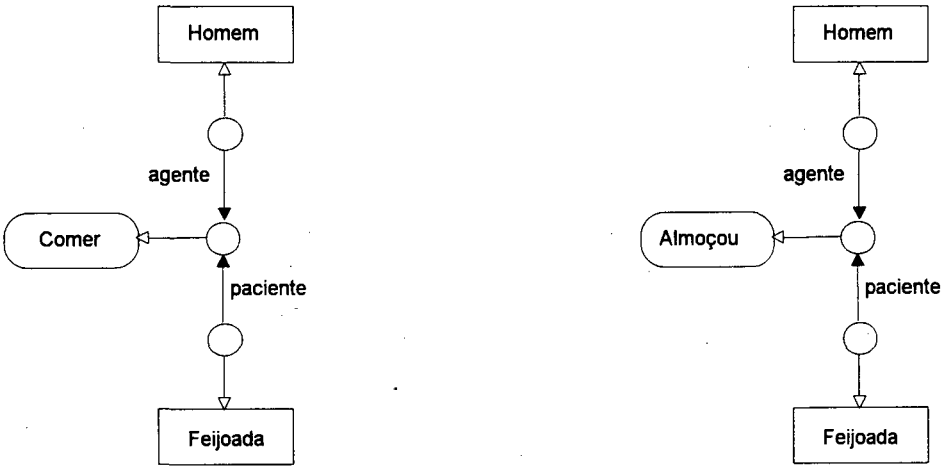
Num primeiro instante, o sistema generaliza a afirmação, gerando o diagrama da esquerda. Note que o *'agente'* de *'morrer'* agora é *'ser humano'* (*'mulher'* + *'homem'*). Nesse momento, a *dica* que afirmava que as duas instâncias (*'mulher'* e *'ser humano'*) poderiam ser idênticas é utilizada para gerar o diagrama da direita, onde uma das instâncias foi eliminada, diminuindo o tamanho do diagrama resultante e aumentando o seu poder de inferência. Essa unificação somente pode ser feita no momento em que as duas instâncias passam a pertencer a mesma classe. Note que o efeito final de todo o processo é o mesmo que seria obtido pela análise e generalização do período *"Maria e Pedro morreram porque bateram o carro"*.

8.5 A importância dos sinônimos

Apesar de não ter sido implementada a generalização com uso de sinônimos, vamos discorrer um pouco sobre ela.

Muitos dos sistemas voltados para a interpretação da linguagem humana já implantados, notadamente os sistemas concebidos por Schank e colaboradores, implementam suas estruturas episódicas baseadas em primitivos (por ex. dependência conceitual). Isso fornece grande poder de manipulação de informações aos sistemas, uma vez que o conjunto de primitivos que se deve manipular é pequeno, e permite a elaboração de inferências semânticas mais facilmente.

No nosso caso, um resultado parecido pode ser obtido através do uso de sinônimos. Antes de mais nada, vejamos os seguintes exemplos:



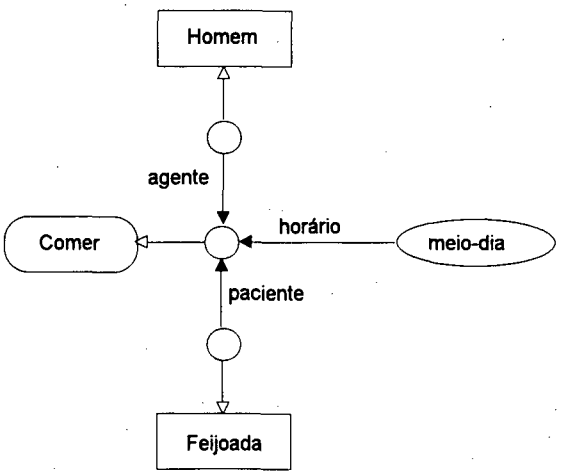
a) João comeu feijoada *b) João almoçou feijoada*

Fig. 8.10 - Usando sinônimos na generalização

Para nós, os fatos mencionados acima são bastante semelhantes, enquanto que, para o sistema, não possuem nada em comum. Entretanto, o cadastramento do sinônimo:

almoçar = comer ao meio-dia

conforme descrito em 6.4, permite que se redesenhe o diagrama *b* da seguinte forma:



a) João almoçou (comeu ao meio-dia) feijoada

Fig. 8.11 - Usando sinônimos na generalização - aplicando o sinônimo

Agora, o sistema já pode identificar uma base de comparação entre *a* e *b*.

O uso de sinônimos, nesses casos, conferiria um maior poder de comparação e, conseqüentemente, de generalização ao sistema. O problema está, e isso tem sido uma crítica constante aos trabalhos de Schank, em saber qual o limite de detalhamento viável, a partir do qual o aumento na capacidade de comparação não compensaria a perda correspondente na riqueza da representação.

Entretanto, um meio termo é desejável e, certamente, se implementado, aumenta muito a capacidade de generalização do sistema, bem como aumenta a capacidade de inferência dos sistemas que forem usar essas informações.

8.6 Problemas do processo de generalização adotado

O processo de generalização de fatos funciona, mas também apresenta problemas e, principalmente, limitações no seu uso. A abstração progressiva, feita pela busca de uma classe mãe comum a duas filhas, pode levar a resultados indesejáveis. Vejamos o seguinte fato:

> *Pedro foi ao McDonald's. Ele comeu um Big-Mac com batatas fritas. Pedro pagou a conta e foi para o supermercado para comprar frutas.*

Num primeiro instante, o parser gera a seguinte estrutura episódica (simplificada):

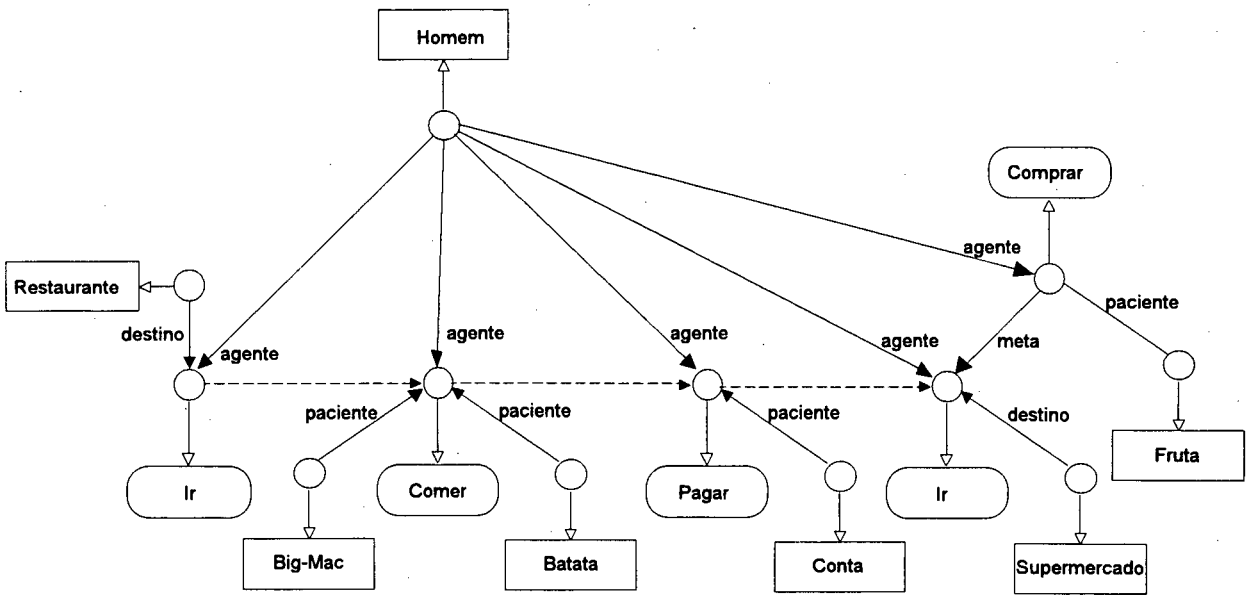


Fig. 8.12 - Problemas da generalização

Durante o processo de generalização, ao analisar as duas ocorrências do verbo *ir*, o sistema descobre que *'restaurante'* e *'supermercado'* possuem *'local'* como classe-mãe comum. Sendo assim, ele substitui as duas ocorrências pela terceira, comum, descaracterizando o fato, como demonstrado a seguir:

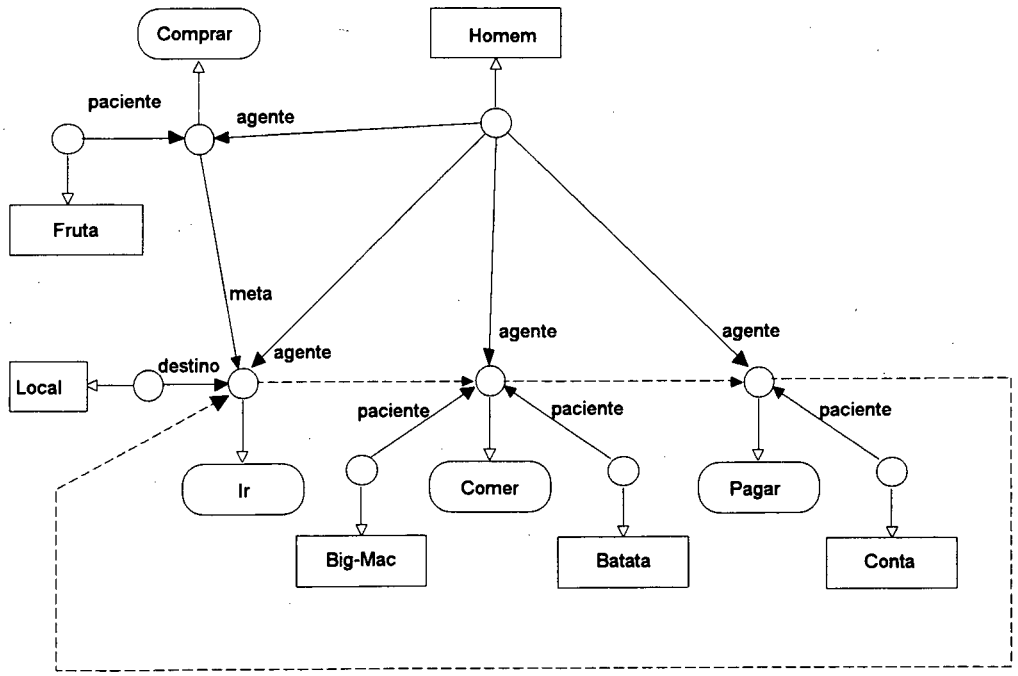


Fig. 8.13- Problemas da generalização

Isso ocorre porque nem sempre uma classe pode ser generalizada. No exemplo acima, tanto *'restaurante'* como *'supermercado'* são importantes para situar as ações e as personagens. Entretanto, em outros exemplos como "*Pedro vendeu o supermercado*", a generalização poderia ser feita tranqüilamente. Uma possível solução para este problema seria definir, para cada tipo de ação, quais atributos seriam *generalizáveis*, e quais não o seriam. No exemplo acima, *'destino'* de *'ir'* não seria generalizável.

A decisão de generalizar, ou não, ainda é uma incógnita para nós. A verdade é que, dentro do nosso cérebro, esse processo é feito inconscientemente, e de alguma forma sabemos o que generalizar, quando generalizar e como generalizar.

9 O Funcionamento do Sistema

9.1 As opções do sistema

Descreveremos a seguir, suscintamente, as opções do sistema:

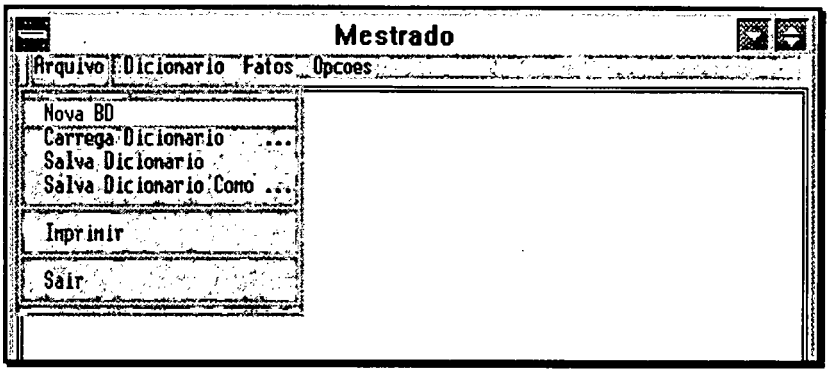


Fig. 9.1 - Menu "Arquivo"

- Nova BD - Permite a inicialização de uma nova base de dados, incluindo um novo dicionário, novas bases de fatos episódicos e genéricos. Usado para fins de testes, para não comprometer resultados já alcançados.
- Carrega Dicionário - Permite que se carregue um dicionário de dados específico, exportado anteriormente.
- Salva Dicionário - Exporta o dicionário de dados para um formato texto, o qual pode ser aproveitado futuramente.
- Salva Dicionário Como - Cria cópia do dicionário de dados atual, não comprometendo o que está armazenado.
- Imprimir - Não implementada.
- Sair - Sair do sistema.

Nota: O processo de salvamento da base de dados é automático, e é feito quando se sai do sistema ou quando se quer salvar o dicionário de dados.

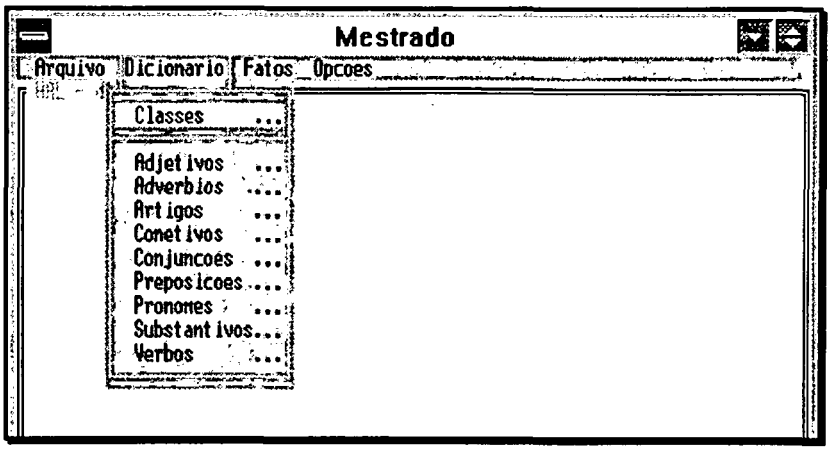


Fig. 9.2 - Menu "Dicionário"

- Classes - Permite o cadastramento de classes, suas hierarquias e atributos.
- Adjetivos, etc. - Permite o cadastramento de palavras dessas categorias.

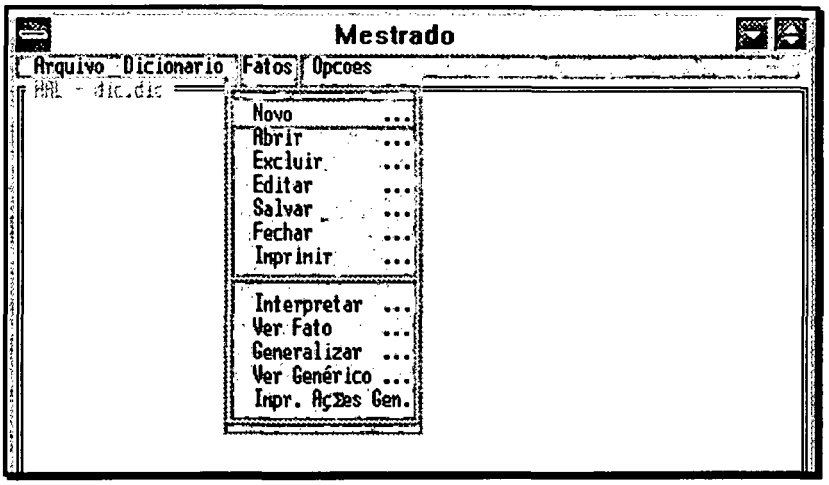


Fig. 9.3 - Menu "Fatos"

Novo - Permite o cadastramento de novos fatos, os quais podem ser compostos de diversos períodos.

Abrir - Trazer para a memória um fato já digitado e armazenado em meio magnético;

Excluir - Excluir um fato do meio magnético. Não implementado.

Editar - Permite a edição do texto do fato para inserções, modificações e exclusões.

Salvar - Permite o armazenamento do texto do fato, bem como da sua interpretação, em meio magnético.

Fechar - Elimina o fato da memória.

Imprimir - Imprime o fato com a sua interpretação.

Interpretar - Analisa o texto do fato, gerando a rede semântica que representa a interpretação dada pelo sistema.

Ver Fato - Permite que se visualize a rede semântica gerada pelo sistema para o fato.
Generalizar - Generaliza o fato, complementando memória genérica anterior.
Ver Genérico - Permite visualização da memória genérica.
Impr. Ações Genéricas - Permite impressão da memória genérica.

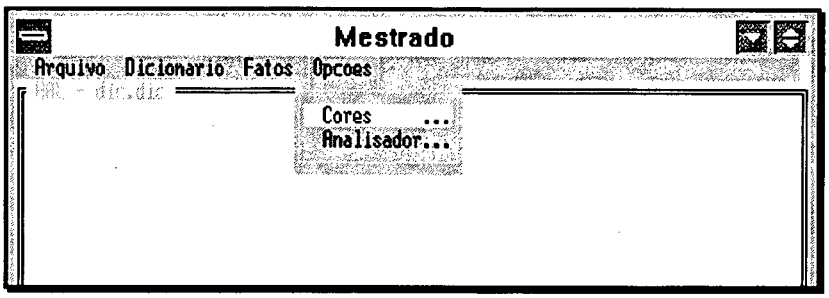


Fig. 9.4 - Menu "Opções"

Estas opções não foram implementadas, mas representam:
Cores - Para permitir uma adaptação às preferências do usuário quanto às cores do sistema.
Analisador - Opções de análise que venham a aumentar a eficiência (velocidade) ou eficácia (veracidade) do processo de análise.

9.2 Diálogos implementados

9.2.1 - Cadastramento de classes

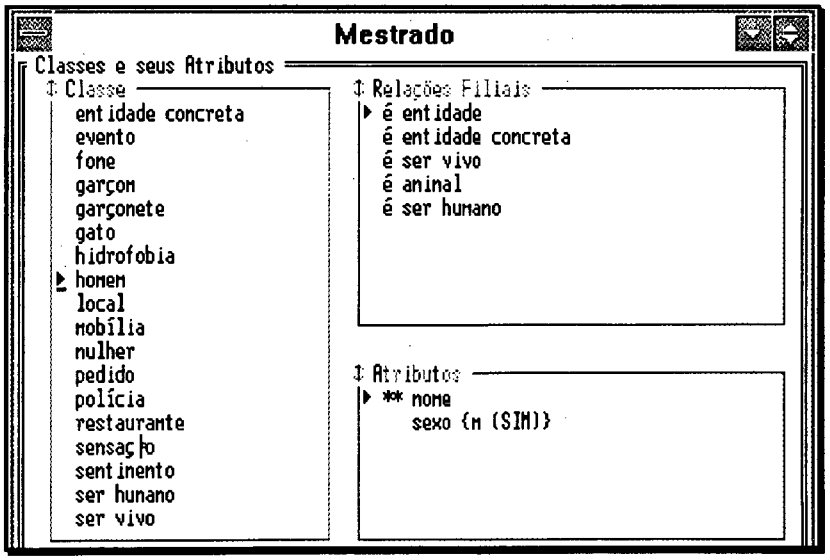


Fig. 9.5 - Cadastramento de classes

Este diálogo compõe-se de três listas:

- Classes - Contém todas as classes cadastradas.
- Relações Filiais - Exibe toda hierarquia que leva até a classe corrente, desde a mais remota até a classe mãe.
- Atributos - Exibe todos os atributos herdados ou próprios da classe. Valores padrões também aparecem.

Nota - O botão de inclusão permite a inclusão de novas classes, mães ou atributos, dependendo de onde está o cursor.

9.2.2 - Cadastramento de adjetivos

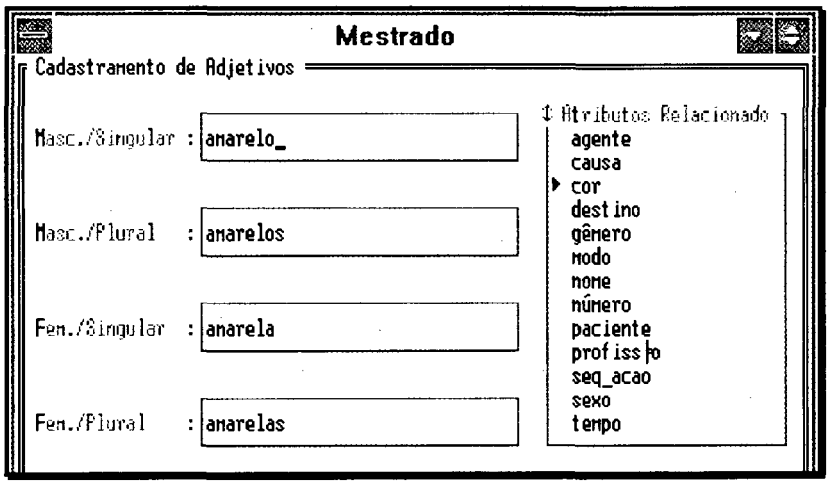


Fig. 9.6 - Cadastramento de adjetivos

Neste diálogo, optou-se pelo cadastramento das quatro formas do adjetivo. Note que o atributo a que se refere o adjetivo está referenciado.

9.2.3 - Cadastramento de preposições e locuções prepositivas

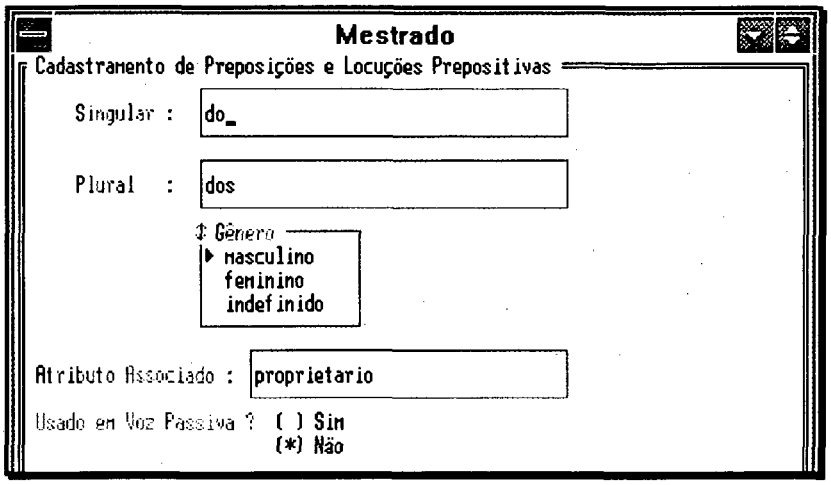


Fig. 9.7 - Cadastramento de preposições

Neste diálogo, deve-se notar a informação do atributo associado ao uso da preposição, bem como a indicação de ser usado na voz passiva, ou não. O primeiro caso para que se possa identificar relações entre objetos a partir de relações nominais. O segundo para auxiliar o analisador sintático no seu trabalho.

9.2.4 - Cadastramento de substantivos

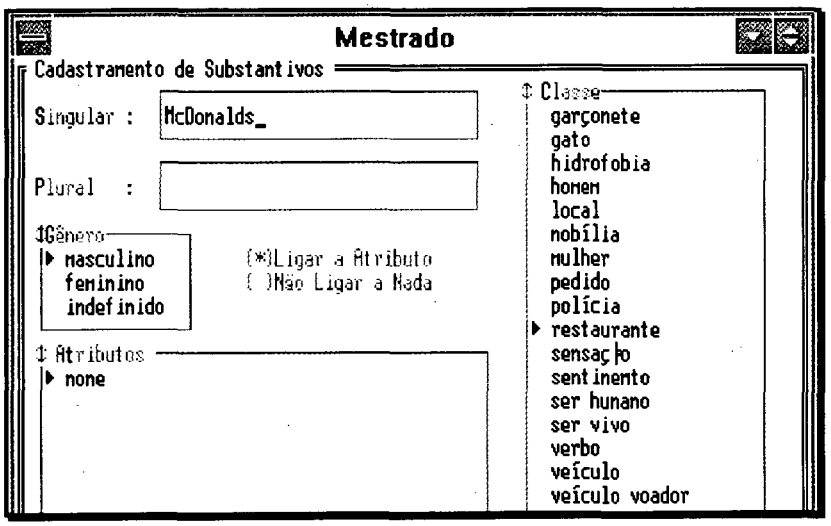


Fig. 9.8 - Cadastramento de substantivos

Neste diálogo, deve-se notar que além da informação da classe a que se refere o substantivo, também deve-se informar o atributo associado à classe que receberá este substantivo.

9.2.5 - Cadastramento de verbos

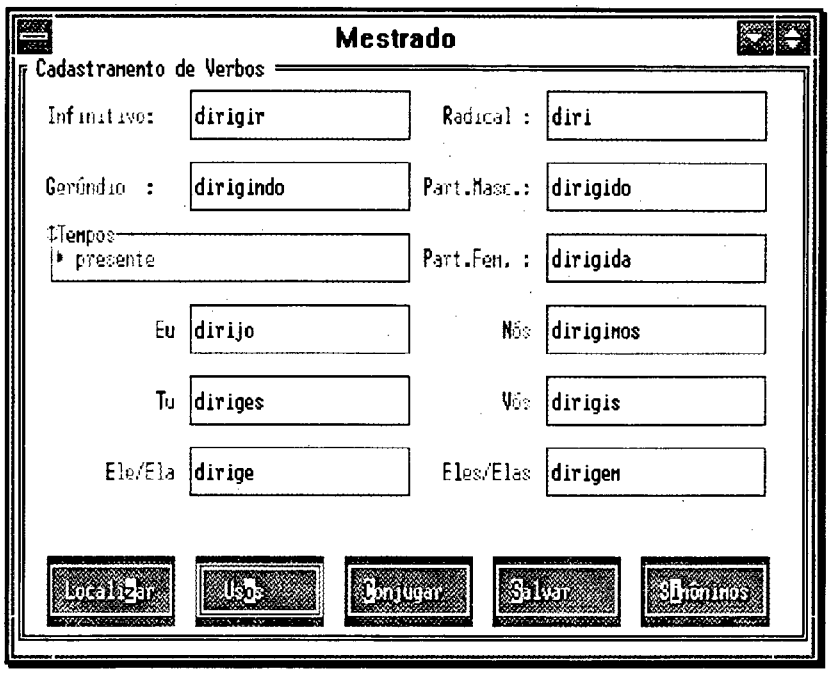


Fig. 9.9 - Cadastramento de verbos

Este é o diálogo típico do cadastramento de verbos. As conjugações para os diversos tempos são selecionadas na lista "Tempo". O diálogo a seguir é invocado qundo se opta pelo botão "Usos".

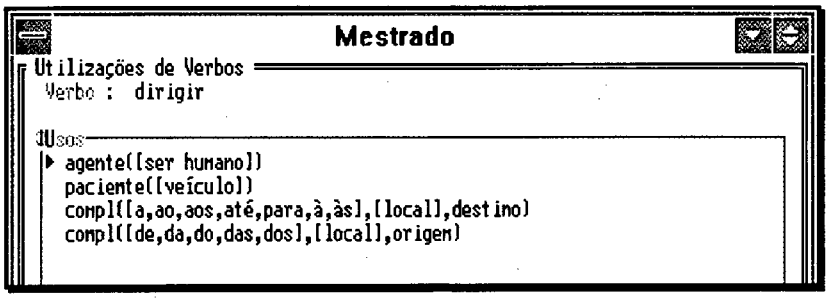


Fig. 9.10 - Utilização de verbos

Aqui estão listadas as utilizações para o verbo em questão. Essas utilizações vão compor os possíveis atributos das ações. Em caso de inclusão, ou alteração, é invocado o seguinte diálogo:

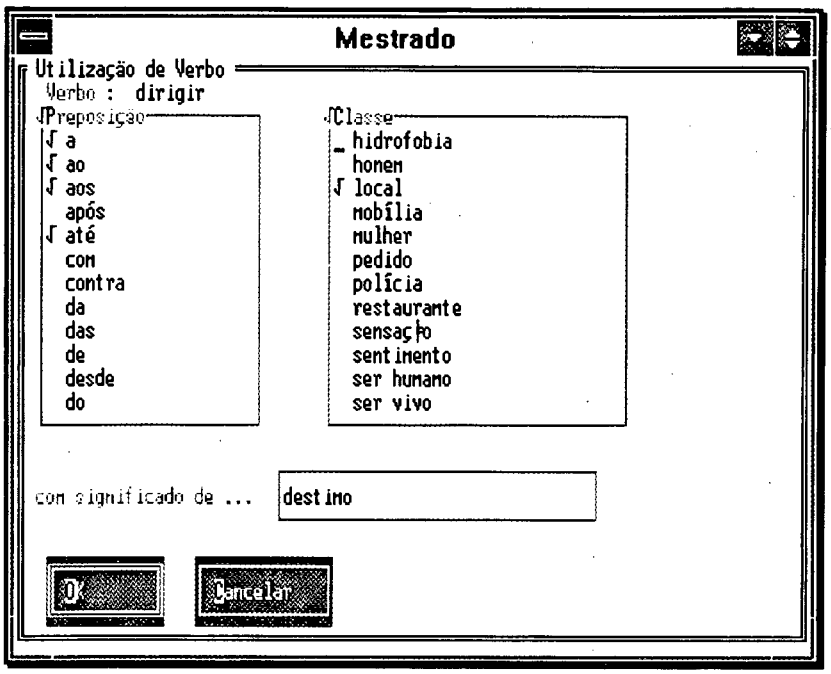


Fig. 9.11 - Cadastramento de utilização de verbos

onde são informadas as preposições e as classes envolvidas, bem como o significado que se deve dar à utilização.

Os sinônimos (ou equivalências verbais) são cadastrados invocando-se o seguinte diálogo, a partir do botão "Sinônimos" da tela da figura 9.9:

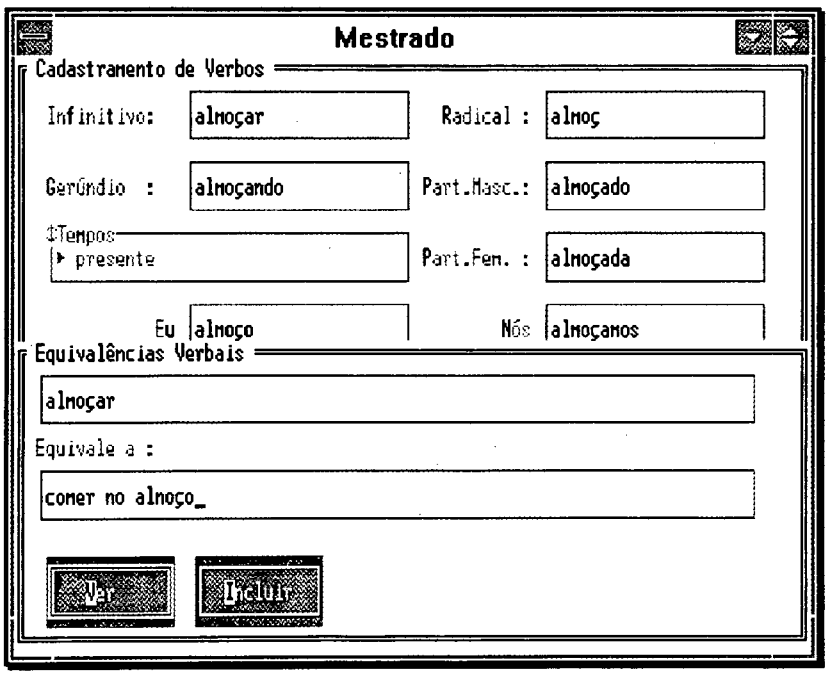


Fig. 9.12 - Cadastramento de sinônimos de verbos

A interpretação do sinônimo gera a seguinte tela:

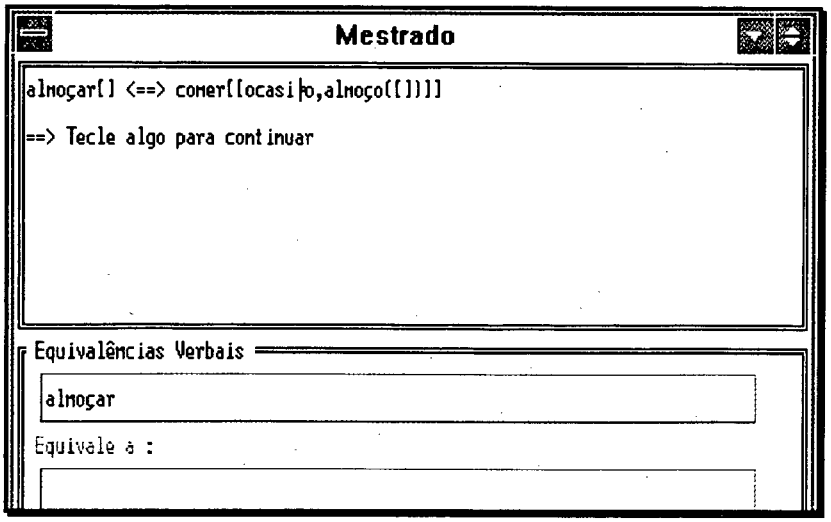


Fig. 9.13 - Sinônimo cadastrado

No cadastramento dos verbos, a maioria das conjugações é feita automaticamente (figuras 9.14 e 9.15, a seguir), bastando, para isso, informar o radical do verbo e um verbo de conjugação semelhante que já tenha sido cadastrado (o primeiro de cada caso tem que ser totalmente digitado).

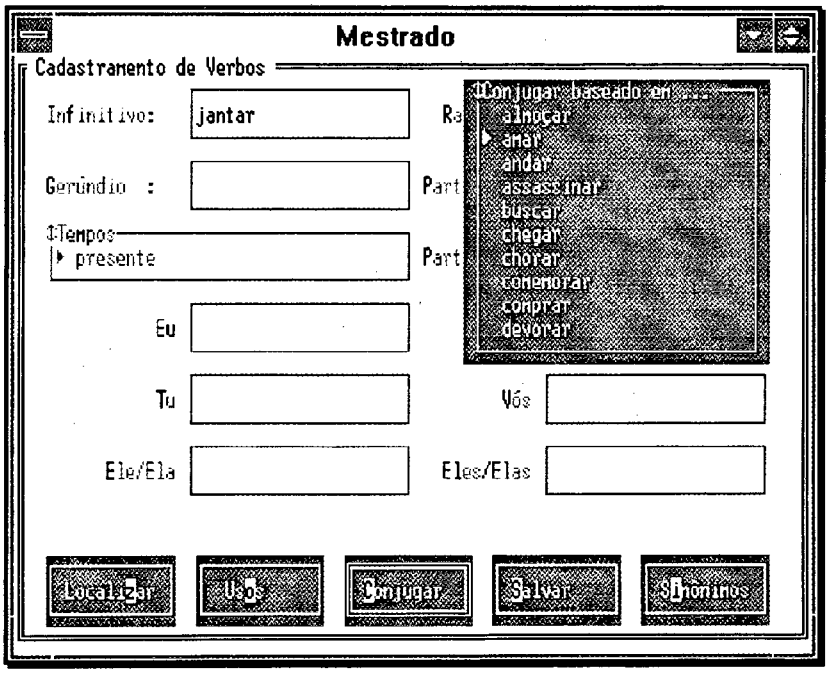


Fig. 9.14 - Seleção de base para conjugação de verbos

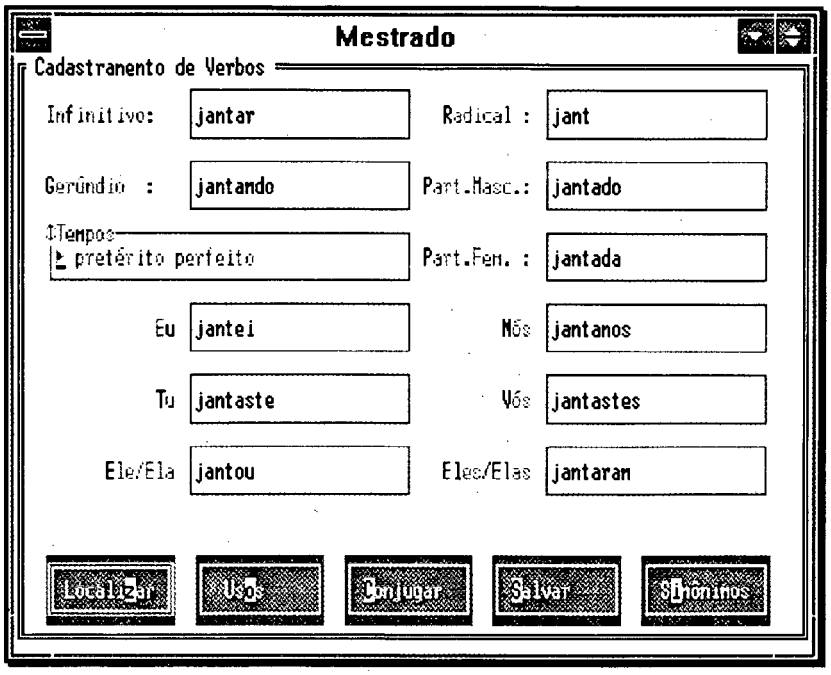


Fig. 9.15 - Conjugação automática

9.3 Interpretando um período

O processo de interpretação do período inicia pela sua digitação (Fatos - Editar).

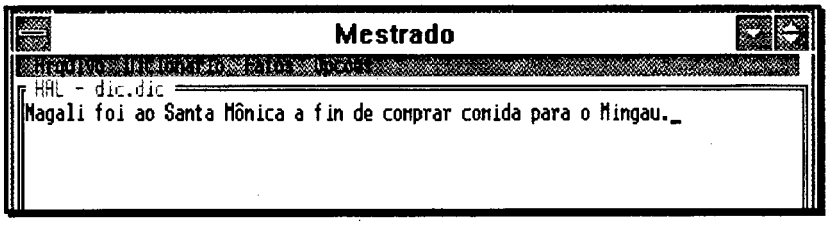


Fig. 9.16 - Exemplo de período

A seguinte tela é gerada quando mandamos o sistema interpretar o período informado (Fatos - Interpretar).

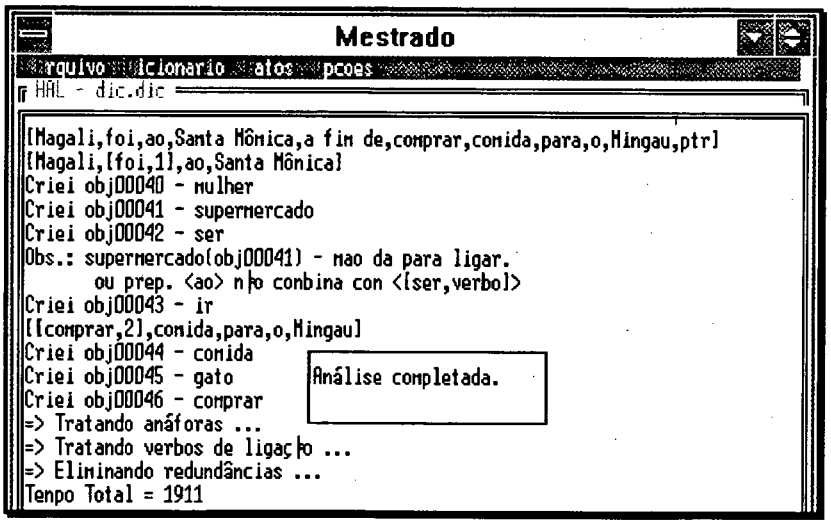


Fig. 9.17 - Interpretação do período

Optou-se por deixar as mensagens de "trace" do sistema para permitir um perfeito acompanhamento dos passos dados. O produto dessa interpretação é o seguinte:

Eventos e ações do texto:	Objetos mencionados no texto :
obj00043 - ir (SIM):	obj00040 - mulher (TALVEZ):
. agente - obj00040 (SIM)	. gênero - f (SIM)
. destino - obj00041 (SIM)	. nome - Magali (SIM)
. meta - obj00046 (SIM)	. sexo - f (TALVEZ)
. seq_acao - 1 (SIM)	obj00041 - supermercado (TALVEZ):
. tempo - < (SIM)	. gênero - m (SIM)
obj00046 - comprar (SIM):	. nome - Santa Mônica (SIM)
. paciente - obj00044 (SIM)	obj00044 - comida (SIM):
. receptor - obj00045 (SIM)	. gênero - f (SIM)
. seq_acao - 2 (SIM)	obj00045 - gato (TALVEZ):
. tempo - = (SIM)	. gênero - m (SIM)
	. nome - Mingau (SIM)

Uma vez interpretado, pode-se generalizar o período através da opção apropriada (Fatos - Generalizar). A tela a seguir mostra os passos da generalização feita. As mensagens de um objeto complementando ele mesmo aparecem porque o processo é feito em dois passos. A figura a seguir demonstra esse fato.

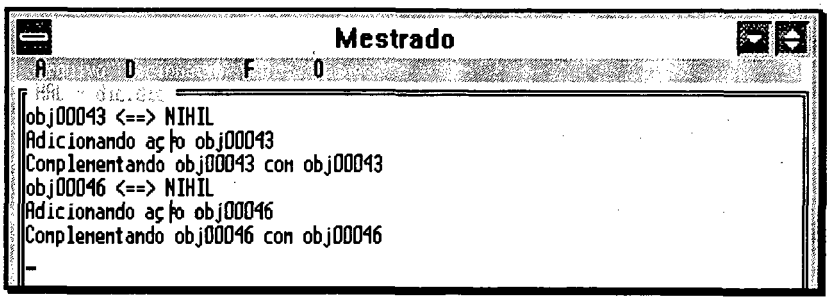


Fig. 9.18 - Generalização do período

Se quisermos visualizar os fatos (memória) genéricos, optamos por "Fatos - Ver genéricos". A tela a seguir nos mostra isso.

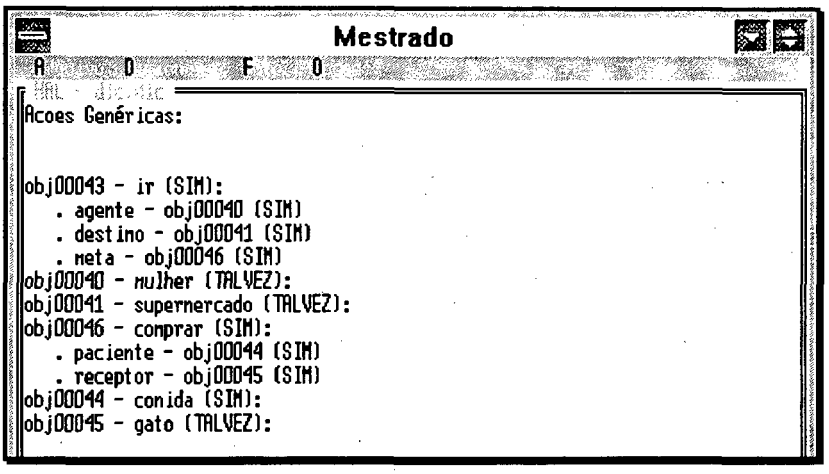


Fig. 9.19 - Produto da generalização do período

9.4 Interpretando e generalizando mais de um fato

A seguir temos as telas de demonstração da memória genérica após a interpretação e generalização dos períodos mencionados em 8.4.

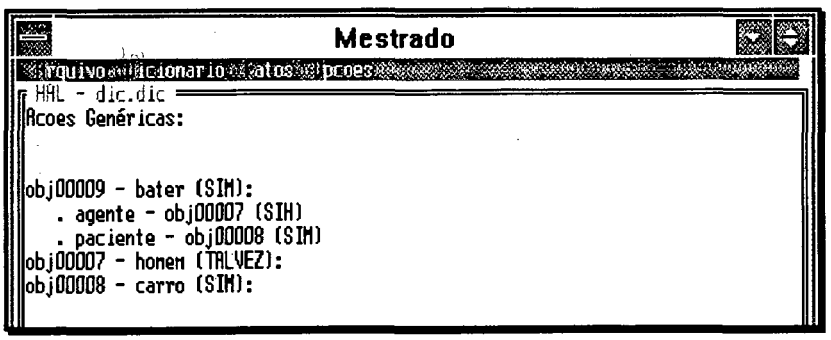


Fig. 9.20 - Generalização após "Pedro bateu o carro"

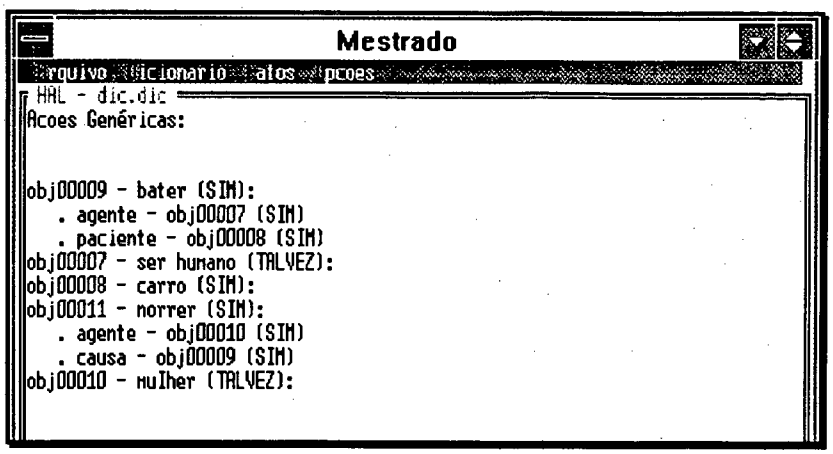


Fig. 9.21 - Generalização após "Maria morreu porque bateu o carro"

Note que existe a distinção entre *mulher* e *ser-humano*, uma vez que somente foi mencionado que uma mulher morreu por causa de batida de carro. A frase a seguir elimina esta distinção.

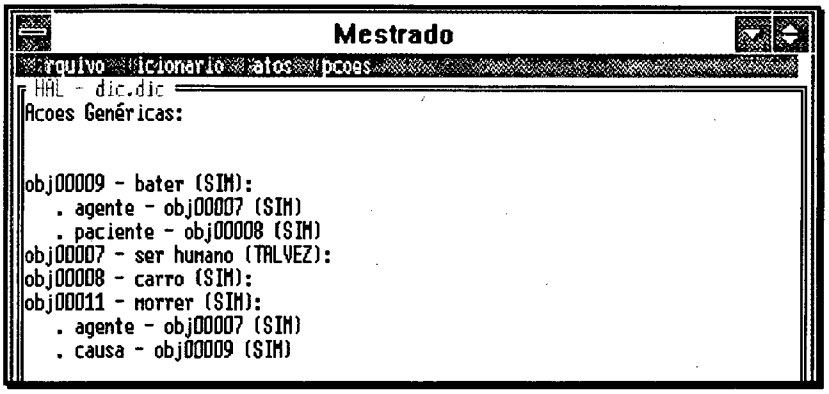


Fig. 9.22 - Generalização após "Pedro morreu porque bateu o carro"

Agora, a memória genérica está de acordo com os diagramas da figura 8.9.

10 Conclusão

O desenvolvimento deste trabalho permitiu a experimentação de uma série de sucessos e de fracassos. Como sucessos, podemos citar o aprendizado em si mesmo, a concretização de algo imaginado, a certeza de que é viável aprofundar-se cada vez mais na pesquisa da compreensão da linguagem humana. Como fracasso, podemos citar a frustração de sempre encontrar um degrau a mais na escada de problemas, após termos vencido o último (pelo menos achávamos que era o último).

Com relação aos objetivos inicialmente propostos, temos a comentar o seguinte:

a) Acreditamos que a escolha de redes semânticas para representar as informações (fatos episódicos e genéricos) foi acertada. Bastante fácil de trabalhar com a linguagem de programação adotada (Prolog), não ofereceu grandes dificuldades durante o desenvolvimento.
b) Inicialmente tencionávamos utilizar quadros (frames) para representar a gramática (principalmente o uso dos verbos), mas a necessidade de codificação de demônios para cada quadro era um obtáculo à dinâmica que desejávamos para o processo de aprendizado dessa gramática.
c) O algoritmo utilizado no desenvolvimento do parser, apesar de semanticamente flexível, apresenta certa rigidez sintática, isto é, não aceita estruturas sintáticas mais complexas ou diferentes das quais foi preparado para reconhecer. Acreditamos que a conjugação do poder semântico da gramática adotada com um algoritmo de reconhecimento sintático mais poderoso melhoraria muito o sistema em questão.
d) O processo de geração de fatos episódicos atingiu os objetivos propostos. Graças ao poder semântico da gramática, o parser consegue gerar estruturas espisódicas apropriadas para cada caso. Algumas limitações a nivel de léxico comprometem a eficácia deste processo, não permitindo ao parser identificar muitas relações nominais possíveis entre objetos de um texto, normalmente associadas ao uso de preposições (veja 7.6).
e) O processo de generalização adotado foi desenvolvido numa tentativa de reproduzir, em pequena escala, o processo feito pelo cérebro humano. Não cremos que tenhamos obtido sucesso completo. Conforme mencionado em 8.6, a decisão de generalizar ou não dois objetos semelhantes é mais complexa do que inicialmente parecia. Conforme a situação, dois objetos podem ser generalizados (em um objeto somente) sem prejuizo semântico para a rede, enquanto que em outra situação pode originar um desastre que compromete todo o significado da rede. Talvez, a definição

de cenários para os fatos venha facilitar essa decisão. Ao associarmos um cenário a um fato, as informações relacionadas à descrição do cenário não poderiam ser generalizadas sob pena de descaracterização do fato. Assim, poderíamos generalizar somente as informações relacionadas às personagens e objetos envolvidos. Outra solução, talvez, fosse não perder os níveis intermediários de generalização, permitindo que informações mais detalhadas fossem recuperadas a qualquer tempo. Essa alternativa, com certeza, iria tornar mais complexa a estrutura da rede semântica genérica, bem como comprometer os tempos de processamento. Uma terceira alternativa seria manter ponteiros nas estruturas genéricas para as estruturas episódicas correspondentes, de forma a poder se obter um padrão específico quando necessário.

O sistema desenvolvido como objeto deste trabalho não possui aplicação prática senão a de conter, em si mesmo, o conhecimento necessário para que se possa desenvolver algo mais avançado. Acreditamos que a idéia seja viável, mas sua utilização na prática demandaria novos estudos, exigindo um parser com potencial sintático muito maior e uma gramática mais completa e flexível.

Futuros Trabalhos

Alguns futuros trabalhos que poderão ser desenvolvidos, de forma a dar continuidade a este, são:

- um estudo aprofundado das técnicas de aprendizado de conceitos a partir de exemplos para que se possa melhorar a capacidade de generalização do sistema;
- um novo parser, mais aperfeiçoado, que possa utilizar as bases genéricas geradas pelo próprio sistema;
- um sub-sistema que possa responder a perguntas a respeito dos conceitos aprendidos, bem como sobre os fatos apresentados a ele;
- implementação do conceito de sinônimos de forma a poder reduzir os fatos a primitivos, mais fáceis de serem manipulados;
- implementação do conceito de cenários, bem como outras alternativas mencionadas anteriormente, para melhorar o processo de generalização.

Finalmente

Finalmente, temos a comentar que esta é uma área de pesquisas fascinante. Ver o sistema dando "seus primeiros passos" rumo a um processo de compreensão mais amplo foi, sem dúvida, empolgante. Uma das características mais "viciantes" da informática é poder ver concretizadas em um computador as idéias mais absurdas. A impressão que se tem é que não existem limites. Com certeza, essa experiência superou todas as outras de nossa vida acadêmica e profissional.

Como desafio final, fica aqui a clássica pergunta que intriga a todos que se aventuram na pesquisa da compreensão da linguagem humana pelos computadores: *Por que é tão relativamente simples ensinar os computadores a realizar tarefas extremamente complexas, as quais nenhum ser humano conseguiria fazer sozinho, enquanto é tão difícil ensiná-lo a compreender nossa linguagem, tão facilmente aprendida pelas nossas crianças?*

11 Bibliografia

ALLWOOD, Jens, ANDERSSON, Lars-Gunnar, DAHL, Osten. *Logic in Linguistics.* Cambridge, London: Cambridge University Press, 1979.

ALLEN, James F., LITMAN, Diane J. Plans, Goals, and Language. *Proceedings of the IEEE,* [s.l.], v.74, n.7, p. 939-47, jul. 1986.

BAREISS, Ray. *Exemplar-Based Knowledge Acquisition.* [s.l.]: Academic Press, Inc - Harcourt Brace Jovanovich Publishers, 1989.

BARR, Avron, FEIGENBAUM, Edward A. *The Handbook of Artificial Intelligence.* Los Altos, CA: Morgan Kaufmann Publishers Inc, 1982, 2 v., V. 2.

___. ___. Los Altos, CA: Morgan Kaufmann Publishers Inc, 1983, 2 v., V. 1.

BARTON Jr.,G.E., BERWICK, R.C., RISTAD,E.S.*Computational Complexity and Natural Language.* Cambridge, MA: The MIT Press, 1987.

BERWICK, Robert C. *The Acquisition of Syntatic Knowledge.* Cambridge, MA: The MIT Press, 1985.

BOBROW, Daniel G., WINOGRAD, Terry. An Overview of KRL, a Knowledge Representation Language. In: *Readings in Knowledge Representation.* Los Altos, CA: Morgan Kaufmann Publishers Inc., 1985, p. 263-265.

BOOCH,Grady. *Object Oriented Design with Applications.* Redwood City, CA: Benjamin/Cummings Publishing Company Inc.,1991.

BRACHMAN, Ronald J. On The Epistemological Status of Semantic Networks. In: Findler, Nicholas V. *Associative Networks - Representation and Use of Knowledge by Computers.* Orlando: Academic Press, 1979, p. 3-50.

BRADY, Michael, BERWICK, Robert C. *Computacional Models of Discourse.* Cambridge, MA: MIT Press,1983.

BRATKO, Ivan. *Prolog - Programming for Artificial Intelligence.* [s.l.] : Addison-Wesley Publishing Company, 1986.

BRUCE, B., MOSER, M. G. Case Grammar. In: Stuart C. Shapiro, *Encyclopedia of Artificial Intelligence.* New York: Wiley-Interscience, 1990, 2 v., V.1.

CARBONELL, J., LANGLEY, P. Machine Learning. In: Stuart C. Shapiro, *Encyclopedia of Artificial Intelligence*, New York: Wiley-Interscience, 1990, 2 v., V.1.

CEGALLA, Domingos Paschoal. *Novíssima Gramática da Língua Portuguesa.* São Paulo: Companhia Editora Nacional, 1992.

CHOMSKY, N. *Aspects of the Theory of Syntax.* Cambridge, MA: MIT Press, 1965.

COLBY, K. *Artificial Paranoia.* New York: Pergamon Press, 1975.

DENNETT, Daniel C. Cognitive Wheels: The Frame Problem of AI. In: Hookway, C. *Minds, Machines and Evolution: Philosophical Studies.* Cambridge, London: Cambridge University Press, 1984, p. 129-151.

DIMANZO, Mauro, ADORNI, Giovanne, GIUCHIGLIA, Fausto. Reasoning About Scene Descriptions. *Proceedings of the IEEE*, v.74, n.7, jul 1986, p. 1013-1025.

DYER, Michael George. *In-Depth Understanding.* Cambridge, MA: The MIT Press, 1983.

FIKES, Richerd, KEHLER, Tom. The Role of Frame-based Representation in Reasoning. *Communications of the ACM*, v.28, n.9, set. 1985, p. 904-920.

FINDLER, Nicholas V. *Associative Networks - Representation and Use of Knowledge by Computers.* Orlando: Academic Press, 1979.

FOX, Barbara A. Interactional Reconstruction in Real-Time Language Process. *Cognitive Science,* v.11, 1987, p. 365-387.

GAZDAR, Gerald, MELLISH, Chris. *Natural Language Processing in Prolog.* Wokingham, England: Addison-Wesley Publishing Company, 1989.

GORDON, Peter C., GROSZ, Barbara J., GILLIOM, Laura A. Pronouns, Names, and The Centering of Attention in Discourse. *Cognitive Science*, v.17, 1993, p. 311-347.

GROSZ, Barbara J., JONES, Karen Sparck, WEBBER, Bonnie Lynn. *Readings in Natural Language Processing*. Los Altos, CA: Morgan Kaufmann Publishers, 1986.

GROSZ, Barbara J., APPELT, D.E., MARTIN, P.A., PEREIRA, F.C.N. TEAM: An Experiment in the Design of Transportable Natural Language Interfaces. *Artificial Intelligence*, v.32, 1987, p. 173-243.

HAMBURGER, Henry, CRAIN, Stephen. Plans and Semantics in Human Processing of Language. *Cognitive Science*, v.11, 1987, p. 101-136.

HARDT, S.L. Conceptual Dependency. In: Stuart C. Shapiro, *Encyclopedia of Artificial Intelligence*. New York: Wiley-Interscience, 1990, 2 v., V.1.

HAYES, Patrick J. The Logic of Frames. In: *Readings in Knowledge Representation*. Los Altos, CA: Morgan Kaufmann Publishers Inc., 1985, p. 287-295.

HILL, J.C. Language Acquisition. In: Stuart C. Shapiro, *Encyclopedia of Artificial Intelligence*. New York: Wiley-Interscience, 1990, 2v. V1.

HIRST, Graeme. Semantics. In: Stuart C. Shapiro, *Encyclopedia of Artificial Intelligence*. New York: Wiley-Interscience, 1990, 2 v., V.2.

HORNSTEIN, Norbert. *Logic as Grammar - An Approach to Meaning in Natural Language*. Cambridge, MA: The MIT Press, 1986.

KOHONEN, Teuvo. *Self-Organization and Associative Memory*. 3.ed. Berlin: Springer-Verlag, 1989.

LANGACKER, Ronald W. An Introduction to Cognitive Grammar. *Cognitive Science*, v.10, 1986, p. 1-40.

LEAKEY, Richard E. *A Evolução da Humanidade*. São Paulo : Companhia Melhoramentos de São Paulo, 1981.

LEBOWITZ, Michael. Generalization From Natural Language Text. *Cognitive Science,* v.7, 1983, p. 1-40.

LYONS, John. *Introdução à Linguística Teórica.* [s.l.]: Companhia Editora Nacional, 1979.

MAIDA, A.S. Frame Theory. In: Stuart C. Shapiro, *Encyclopedia of Artificial Intelligence.* New York: Wiley-Interscience, 1990, 2v., V.1.

MCKEOWN, Kathleen R. Discourse Strategies for Generating Natural-Language Text. *Artificial Intelligence,* v.27, 1985, p. 1-41.

___. Language Generation: Applications, Issues and Approaches. *Proceedings of the IEEE,* v.74, n.7, 1986, p. 961-68.

MICHALSKI, R.S. Concept Learning. In: Stuart C. Shapiro, *Encyclopedia of Artificial Intelligence.* New York: Wiley-Interscience, 1990, 2 v., V. 1.

MIIKKULAINEN, Risto, DYER, Michael G. NLP With Modular PDP Networks and Distributed Lexicon. *Cognitive Science,* v.15, 1991, p. 343-399.

MINSKY, Marvin. *Semantic Information Processing.* Cambridge, MA: The MIT Press, 1968.

___. A Framework for Representing Knowledge. In: *Readings in Knowledge Representation.* Los Altos, CA: Morgan Kaufmann Publishers Inc, 1985, p. 245-262.

___. *The Society of Mind* . New York: Touchstone, 1988.

MURPHY, Gregory L. Comprehending Complex Concepts. *Cognitive Science,* v.12, 1988, p. 529-562.

NEWPORT, Elissa L. Maturational Constraints on Language Learning. *Cognitive Science,* v.14, 1990, p. 11-28.

PAZZANI, Michael. A Computational Theory of Learning Causal Relationships. *Cognitive Science,* v.15, 1991, pp. 401-424.

PEREIRA, Fernando C. N., WARREN, David H. D. Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Augmented Transition Networks. *Artificial Intelligence*, v.13, 1980, p. 231-278.

PETRICK, Stanley R. Parsing. In: Stuart C. Shapiro, *Encyclopedia of Artificial Intelligence*. New York: Wiley-Interscience, 1990, 2 v., V.2.

REINKE, R.E., MICHALSKI, R.S. Incremental Learning of Concept Description. In: J. E. Hayes; D. Michie; J. Richards. *Machine Intelligence 11*. Oxford: Claredon Press, 1988, p. 263-88.

RICH, Elaine. *Inteligência Artificial*. São Paulo: McGraw-Hill, 1988.

RUMBAUGH, James, et al. *Modelagem e Projetos Baseados em Objetos*. Rio de Janeiro: Editora Campus, 1994.

SANDEWALL, Erik. Nonmonotonic Inference Rules for Multiple Inheritance with Exceptions. *Proceedings of the IEEE*, v.74, n.10, out. 1986, p. 1345-1353.

SCHA, R. J. H., BRUCE, B. C., POLANYI, L. Discourse Understanding. In: Stuart C. Shapiro, *Encyclopedia of Artificial Intelligence*. New York: Wiley-Interscience, 1990, 2 v., V.1.

SCHANK, Roger C., CARBONELL, Jaime G. RE: The Gettysburg Address. In: Findler, Nicholas V. *Associative Networks - Representation and Use of Knowledge by Computers*. Orlando: Academic Press Inc., 1979, p. 327-362.

___, CHILDERS, Peter G. *The Cognitive Computer*. Reading, MA: Addison-Wesley Publishing Company Inc., 1984.

___. Language and Memory. In: Barbara J. Grosz; Karen Sparck Jones; Bonnie Lynn Webber. *Readings in Natural Language Processing*. Los Altos, CA: Morgan Kaufmann Publishers, 1986, p. 171-91.

SCHUBERT, Lenhart K., GOEBEL,Randolph G., CERCONE,Nicholas J. The Structure and Organization of a Semantic Net For Comprehension and Inference. In: Findler, Nicholas V. *Associate Networks - Representation and Use of Knowledge by Computers*. Orlando: Academic Press, 1979, p. 121-175.

SEARLE, John R. Minds, Brains and Programs . *The Behavioral and Brain Sciences*, v.3, 1980, p. 417-424.

SOWA, John F. *Conceptual Structures: Information Processing in Mind and Machine.* Reading, MA: Addison-Wesley Publishing Company, 1984.

STERLING, Leon, SHAPIRO, Ehud. *The Art of Prolog.* Cambridge, MA: The MIT Press, 1987.

TURING, Alan M. Computing Machinery and Intelligence. *Mind*, v.59, n.2236, out. 1950, p. 433-460.

WATKINS, M. Episodic Memory. In: Stuart C. Shapiro. *Encyclopedia of Artificial Intelligence.* New York : Wiley-Interscience, 1990, 2 v., V.1.

WEISCHEDEL, Ralph M. Knowledge Representation and Natural Language Processing. *Proceedings of the IEEE*, v.74, n.7, jul. 1986, p. 905-920.

WEIZENBAUM, J. ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine. *Communications of the ACM*, v.9, n.1, jan. 1966, p. 36-44.

WINOGRAD, Terry. *Language as a Cognitive Process* . Reading, MA: Addison-Wesley Publishing Company, 1983, v. 1: Syntax.

WOODS, William A. Important Issues in Knowledge Representation. *Proceedings of the IEEE*, v.74, n.10, out. 1986, p. 1322-34.