Conceitos de Inteligência Artificial aplicados a Lingüística

Bertilo Frederico Becker
Disciplina de Lingüística Computacional I

SUMÁRIO

A Inteligência Artificial é um conjunto de princípios e de recursos destinados a facilitar o processamento computacional de atividades características do comportamento inteligente das pessoas humanas. Existem várias tentativas de representar o conhecimento e os processos de raciocínio, mas um modelo unificado está ainda além do alcance da pesquisa. Ela leva a um enfoque mais generalista e a uma abordagem interdisciplinar que enriquece as várias áreas do conhecimento. Mas os efeitos colaterais sobre os paradigmas da pesquisa os tornam com essa busca ao mesmo tempo mais poderosos e mais rigorosamente controlados, pelo esforço de generalização.

ABSTRACT

Artificial Intelligence, is a set of principles and techniques used to help with the computational processing of those activities claimed to be characteristic of intelligent behavior of human people. Several trys for representing knowledge and thinking processes have not yet succeeded in bringing an unified model. Research in this direction brings today more generalistic and interdisciplinary views that enrich all of the knowledge areas it involves. But the side effects on the resarch paradigms make them at once more powerfull and more rigorous, thanks to the generalization effort.

INTRODUÇÃO

A Inteligência Artificial (IA) é parte da ciência da computação. Ela se destina a desenvolver sistemas que demonstrem um com-

portamento que para pessoas humanas seria considerado inteligente. Alguns sistemas assim têm sido comercializados para usuários nas áreas de química, biologia, geologia, engenharia e medicina, servindo para resolver problemas com um nível de desempenho semelhante ao dos profissionais dessas áreas. Outros sistemas inteligentes permitem a execução repetitiva de ações de sensoriamento remoto e manipulação por dispositivos robóticos. E, o que é mais interessante para o assunto presente, existem programas capazes de responder a perguntas em dialetos simples de alguma linguagem natural e, também fazer traduções.

Existem fortes indícios de que os sistemas de IA (e os computadores certamente) exercerão crescente impacto sobre a sociedade dos próximos anos. Em especial nas atividades técnico-científicas.

Existe uma conexão intuitiva entre a complexidade de um dispositivo mecânico e o grau de inteligência de seu construtor. Relógios e máquinas industriais do século XIX foram apenas precursores do computador, cuja complexidade ultrapassa em várias ordens de grandeza os mais intricados daqueles mecanismos. Inicialmente construídos para executar com maior rapidez as tarefas repetitivas dos algoritmos de cálculo, em especial nas repartições estatais e nas grandes organizações comerciais, logo os computadores invadiram as áreas de pesquisa das universidades, depois foram introduzidos no controle acadêmico e na administração financeira e de pessoal, como aconteceia também em mais e mais empreendimentos públicos e particulares. Mas sua construção não pode mais ser creditada a um indivíduo ou a um pequeno grupo. O computador reflete um estágio avançado de consciência coletiva, assim como o estado a que chegou a ciência como fenômeno social da humanidade.

O presente trabalho reflete em sua estrutura o estado da arte em IA, constituindo-se em uma colcha-de-retalhos das várias abordagens que se fazem do assunto. Primeiramente, um conceito de inteligência é apresentado, mostrando-se, com exemplos em PROLOG, o tipo de representação que se faz do processo de inferência. Depois, aborda-se o problema da busca de solução em um conjunto de estados que um problema pode apresentar, as várias maneiras de suprir a máquina com os recursos que ela precisa para suas decisões, e que levam o nome de conhecimentos. A seguir, três enfoques para esse conhecimento são sucessivamente mostrados: as redes semânticas, as molduras espaço-temporais, e as gramáticas. Na conclusão, a moral da história.

1. INTELIGÊNCIA

Embora fazer contas e escrever as capitais dos países do mundo sejam indícios de inteligência em pessoas, os computadores podem fazê-lo sem no entanto demonstrar um *comportamento* inteligente, ou uma *compreensão* desses processos. Por isso, não é útil discutir se uma máquina é ou não inteligente, mas se um programa de computador pode ser feito para exibir um *comportamento inteligente*. Por exemplo, o reconhecimento fisionômico de pessoas e o raciocínio por analogia são ainda, no computador, pouco mais que jogos interessantes. É que as pessoas o fazem usando processos inconscientes para os quais não se criou até agora os adequados modelos computacionais.

Os primeiros bons programas de IA foram os que resolviam quebra-cabeças e jogos como o xadrez. Eles se baseiam em técnicas de pesquisa de alternativas e de redução de problemas complexos a problemas mais simples. Em aberto nessa área está a capacidade de criar representações conceituais para os problemas, coisa que a máquina ainda não faz por si.

Para demonstrar a capacidade de representar soluções simples e elegantes para quebra-cabeças conhecidos, mostra-se a seguir um programa em PROLOG que resolve o problema conhecido como das oito rainhas (Fig. 1.1): Dadas oito rainhas e um tabuleiro de xadrez, colocar as rainhas no tabuleiro de modo que elas não se ataquem mutuamente. Em xadrez, uma rainha pode deslocar-se um número arbitrário de posições, em direção horizontal, vertical ou diagonal. Para achar uma solução a tal problema, as rainhas devem ser colocadas de modo que se tenha uma em cada linha e coluna, e ainda, que duas rainhas não possam 'ver-se' ao longo de linhas diagonais. Como cada rainha vai estar em uma linha (x = 1,2,3,4,5,6,7,8), a solução consistirá da relação das respectivas colunas (Y) em que as rainhas devem ser colocadas. Para controlar a restrição sobre diagonais, numeram-se as 15 diagonais ascendentes como

u = x − y, u em [-7,7] (desde 1-8 até 8-1).

e as diagonais descendentes como

v = x + y, v em [2,16] (desde 1+1 até 8+8).

A solução do problema ficará por conta do comando:

```
? _ solução(S).
```

que será respondido com o valor de S, que deve ser *instanciado* com a lista de colunas Y em que as rainhas devem ficar, ao longo das linhas numeradas de 1 a 8. Para começar, ter-se-ão listas de linhas, de colunas, de diagonais ascendentes e de diagonais descendentes disponíveis, isto é, ainda não ocupadas por rainhas. A cada rainha que for colocada no tabuleiro, o processo consiste em retirar das respectivas listas os elementos ocupados. A ordem em que as coisas acontecem é determinada por instanciação e tentativa, ocorrendo retrocesso ('backtracking') sempre que para alguma combinação (x,y) de linha e coluna não estiver disponível alguma das diagonais x-y ou x+x. Quando as linhas e colunas estiverem todas ocupadas, então o problema estará resolvido.

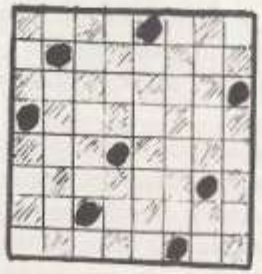

FIG. 1.1 Uma solução do problema das oito rainhas

A solução mostrada na Fig. 1.2 é elegante e relativamente curta, graças ao mecanismo de generalização, introduzido na *busca*, que na verdade resolve o problema de N rainhas, se se tiver as correspondentes listas de linhas, colunas e diagonais devidamente inicializadas.

Entre os sistemas importantes de IA acham-se os que desenvolvem raciocínio lógico e prova automática de teoremas a partir de uma base de dados e de relações entre esses dados. Sistemas desse tipo servem ainda para extrair os fatos relevantes a uma questão, de dentro de um grande banco de dados, ou para relacionar os fatos que justifiquem determinada hipótese.


```
solução(S) :- busca(S. % consiste em construir a lista S
        [1,2,3,4,5,6,7,8], % dadas as listas iniciais: linhas X
        [1,2,3,4,5,6,7,8],                    % colunas Y
        [-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7],  % diagonais
        [2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]).
busca ([ ], [ ],-,-,-). % tudo vazio: começa a volta das chamadas
                %              recursivas
busca([Y|Y1],[X|X1],Dy,Du,Dv) :- % separa Y e X das listas
        tira(Y,Dy,Dy1),     % retira uma coluna.
                            % instanciando Y
        U is X - Y,         % calcula a diagonal descend.
        tira (U,Du,Du1).    % e a retira da lista, se lá estiver
                            % senão deve tentar outro Y
        V is X + Y,         % calcula a outra diagonal
        tira(V,Dv,Dv1),     % e a retira, ou retrocede
                            % para tentar outro Y
        busca(Y1,X1,Dy1,Du1,Dv1).   % busca com listas menores
% até que X1 esteja vazio
tira(A,[A|Lista],Lista).            % tira se for o primeiro
tira(A.[B|Lista],[B|Nova]) :-       % tira do meio da lista
        tira(A.Lista,Nova).   % tirando do segundo em diante
```

FIG. 1.2 Programa PROLOG para o problema das oito rainhas

Outro exemplo, suficientemente simples para ser entendido por inspeção, é composto de regras de tradução e de transformação para reduzir uma expressão lógica e suas proposições básicas, e então verificar a validade da expressão (Fig. 1.3).

Para usar este programa com o objetivo de provar um teorema, este deve ser submetido como um comando de tradução, seguido de um comando de execução. O comando de tradução decompõe o teorema em proposições individuais, que são então afirmadas (comando 'assert' do PROLOG) para o banco de dados do programa. Proposições redundantes são eliminados do banco de dados (comando 'retract'). Para que uma mesma proposição *C*, ocorrendo mais de uma vez, não seja redundantemente afirmada, ela será sinalizada com a inclusão de um fato *feito(C)*, que é consultado pelo programa. Como é mais eficiente detectar uma contradição do que achar uma tautologia (prova da fórmula), convém submeter a fórmula negada e esperar por uma contradição do que achar uma tautologia (prova da fórmula), convém submeter a fórmula negada e esperar por uma contradição:

?- traduz (~ ((a => & b) & (b => c) => (a => c))), executa.

Este exemplo, usando PROLOG, usa e abusa da estrutura interna dessa linguagem, cujo funcionamento se baseia exatamente sobre um algoritmo de 'prova de teoremas em lógica proposicional' (BRATKO, 86:397).

```
% Regra para cláusulas contraditórias:
    [ clausula(X), clausula("X) ] --->
       [ write('Achou contradição'). pare ].
    % Remove uma cláusula verdadeira:
    [ clausula(C), em(P.C), em("P.C) ] ---) [ retract (C) ].
    % Regra para simplificar uma cláusula:
       [ substitui(clausula(C). clausula(C1) ].
    % Um caso especial da resolução:
    [clausula(P),clausula(C),apaga("P.C.C1), not feito(P,C,P)]
       ---) [ assert(clause(C1)), assert( feito(P.C.P)) ].
    % Outro caso especial da resolução:
    [clausula("P),clausula(C),apaga(P,C,C1), not feito("P,C,P)]
       ---) [assert(clausula(C1)), assert(feito("P,C,P)) ].
    % Caso geral da resolução:
    [ clausula(C1), apaga( P,C1,CA).
      clausula(C2), apaga("P,C2,C8), not feito(C1,C2,P) ] ---)
       [assert(clausula(CA v CB)), assert(feito(C1,C2,P)) ].
    % Última regra: a resolução empacou:
    [ ] ---) [ write('Não é contraditóprio'), pare].
```

FIG. 1.3 Declarações em PROLOG para prova de teoremas

```
% apaga(X,E,E1) significa que vai tirar uma sub-expressão
%               disjuntiva X de uma expressão. E dando outra
%               expressão E1
apaga(X, X v Y, Y).
apaga(X, Y v X, Y).
apaga(X, Y v Z, Y v Z1) :- apaga(X,Z,Z1).
apaga(X, Yv Z, Y1 v Z) :- apaga(X,Y,Y1).

% em(P,E) significa que P é uma sub-expressão disjuntiva
%         na expressão E
em(X,X).
em(X,Y) :- apaga(X,Y,-).
```

FIG. 1.4 Cláusulas PROLOG de eliminação e pertinência


```
% op(Prioridade,Classe,Símbolo)
   :- op(100,fu,").          % Negação
   :- op(110,xfy,&).         % Conjunção
   %- op(120,xfy,v).         % Disjunção
   :- op(130,xfy,=>).        % Implicação
   :- op(800,xfx,--->).      % Operador de ação
```

FIG. 1.5 Definição dos operadores lógicos e suas prioridades

```
% Regra que procura AÇÕES para executá-las:
    executa :- Cond ---> Ação, testa(Cond), faz(Acao).
    e
% Verifica uma Condição:
    testa([ ]).              % Nada para testar
    testa([Primeiro | Resto]) :-
            call(Primeiro), testa(Resto).
% Faz a AÇÃO:
    faz([pare]) :- !.        % Parar a execução
    faz([ ]) :- executa.     % Terminou uma execução
    faz([Primeiro | ;Resto]) :-
            call(Primeiro), faz(Resto).

% Regra para substituir A por B:
    substitui(A,B) :- retract(A) ,!, assert(B).

% Regras para tradução:
    traduz(F & G) :- !, traduz(F), traduz(G).
                % Decompõe a conjunção
    traduz(Fórmula) :- muda(Fórmula,Nova) ,!, traduz(Nova).
                % Usa equivalência antes de traduzir
    traduz(Fórmula) :- assert(cláusula(Formula)).
                % Por último coloca no banco de dados

% Regras para mudar fórmulas:
    muda("(X),X) :- !.              % Elimina dupla negação
    muda(X => Y, "X v Y) :- !.      % Elimina a implicação
    muda ("(X & Y), "X v "Y) :- !.  % Aplica De Morgan
    muda ("X v Y), "X & "Y) :- !.   % Aplica De Morgan
    muda(X & Y v Z,(X v Z)&(Y v Z)) :- !,  % Aplica distribuição
    muda(X v Y & Z,(X v Y)&(X v Z)) :- !,  % Aplica distribuição
    muda(X v Y, X v Y1) :- muda(Y,Y1),!.   % Sub-expressão
    muda(X v Y, X1 v Y) :- muda(X,X1),!.   % Idem
    muda("X,"X1) :- muda(X,X1).            % Idem
```

FIG. 1.6 Núcleo do programa de prova de teoremas

A compreensão da linguagem natural foi também exaustivamente pesquisada. Sistemas foram criados para interpretar sentenças em inglês e buscar em um banco de dados (BD) as informações relevantes às questões assim formuladas; para traduzir de uma língua para outra; e para acumular dados em um BD a partir de textos em linguagem natural. Mais tais sistemas estão ainda longe de ter a proficiência das pessoas nessas tarefas, embora se mostrem úteis em diversas aplicações. As melhorias constantes nessa área provêm de melhores representações para o conhecimento e a crescente importância da expectativa (qual *novo* pode vir a ser relevante para os *dados* disponíveis) nesses sistemas. Mas os modelos computacionais assim construídos se baseiam em teorias que não fecham muito bem com o que se conhece sobre os processos mentais da fala e da compressão da linguagem.

Outras áreas comuns aos pesquisadores de IA são a programação automática de computadores, sistemas capazes de aprender por si, sistemas especialistas em diversos setores da atividade profissional, a robótica e o reconhecimento de formas e padrões.

2. BUSCA

A busca sempre reflete um estado de insatisfação.

Uma situação que não é satisfatória constitui um problema.

A busca da solução de um problema constitui um dos mais difíceis problemas da I.A. A riqueza de meios de busca de que dispõe a mente humana ainda não pode ser simulada por um computador. Provavelmente, porque inclui julgamentos estéticos e cinestésicos relacionados aos aspectos bioquímicos e endocrinológicos do processamento mental.

Os modelos computacionais de busca de solução dependem fortemente da representação interna do problema, isto é, de sua formulação em termos dos recursos computacionais. As representações mais comuns, e também as mais poderosas representações gerais, são em forma de grafo ou matriz de estados, e em forma de árvore 'e/ou'.

Um grafo de estados (e sua correspondente matriz) consiste em um conjunto de situações possíveis que o problema pode apresentar, e os modos de se passar de um desses estados a outro, em busca daquele, ou de um daqueles que correspondam a uma solução. Assim, por exemplo, no *jogo-da-velha* (Fig. 2.1) os estados do problema são todos os possíveis estágios, alguns dos quais são mostrados no diagrama baixo. Um caminho desde o *nodo inicial* até um *nodo final* constitui uma solução do problema. Neste jogo, um nodo final tem uma linha, uma coluna ou uma diagonal marcada toda com X ou toda com O.

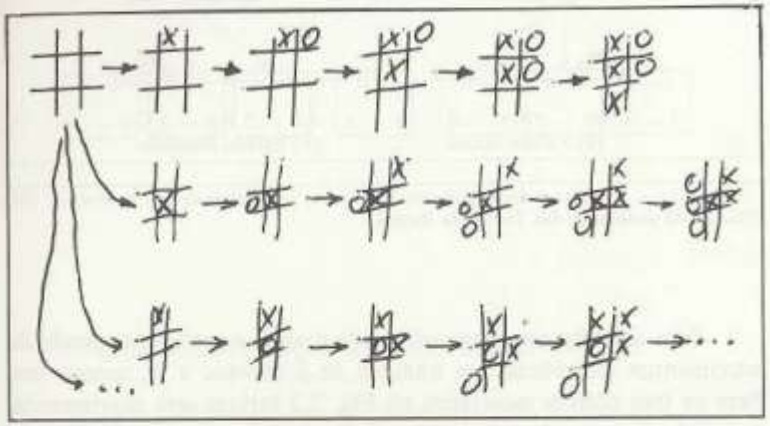

FIG. 2.1 Parte do gráfico de estados para o jogo-da-velha

O conjunto de estados possíveis de um problema é dito *espaço de estados*. No caso do jogo-da-velha, é possível jogá-lo de aproximadamente 9! = 362.880 maneiras diferentes, às quais, no entanto podem ser reduzidas, por considerações de simetria, a menos de 500 estados distintos.

Para resolver-se um problema, pode-se pesquisar seu espaço de estados à procura de um caminho que leve à solução desejada. Isso pode ser extremamente oneroso em alguns casos, quando tal espaço é muito grande. Por exemplo, para o jogo de xadrez, o espaço de estados contém mais do que 10. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000 estados. Outras abordagens se fazem necessárias, como a técnica de redução de problema. Consiste em generalizar o problema de modo a poder decompô-lo em termos de problemas menores, algum dos quais possa ser resolvido imediatamente, sendo os outros, então reduzidos pela mesma técnica.

O problema das Torres de Hanoi é um exemplo interessante (Fig. 2.2). Dada uma pilha A de discos ou moedas de tamanhos decrescentes, pede-se transferi-los a uma outra pilha C, um a um, cuidando de manter a ordem decrescente, e usando uma pilha auxiliar B para isso.

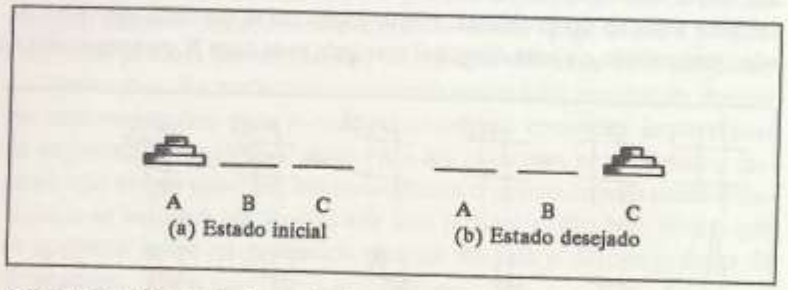

FIG. 2.2 O problema das Torres de Hanoi

Para um número arbitrário N de discos, a solução depende de movimentos individuais em número de 2 elevado a N, menos um. Para os três ediscos mostrados na Fig. 2.2 seriam sete movimentos. A seqüência exata de movimentos é a solução do problema. Para resolvê-lo, pode-se subdividi-lo em três partes: Mover N-1 discos de A para B, mover um disco de A para C, e então mover os N-1 discos de B para C. Mover N-1 discos é um problema menor que mover N discos, e mover um único tem uma solução trivial. O primeiro e o terceiro desses problemas podem, por sua vez, ser decompostos de maneira similar (Fig 2.3). Ali, as folhas da árvore, lidas da esquerda para a direita, constituem a seqüência de movimentos unitários que é a solução do problema.

Os problemas práticos e os aspectos teórico-matemáticos das técnicas de busca têm sido discutidos na literatura de IA (BARR, 81, vol. 1). Consistem basicamente em determinar que partes do espaço de estados devem ser computados preferencialmente, e qual parte pode ser descartada antecipadamente.

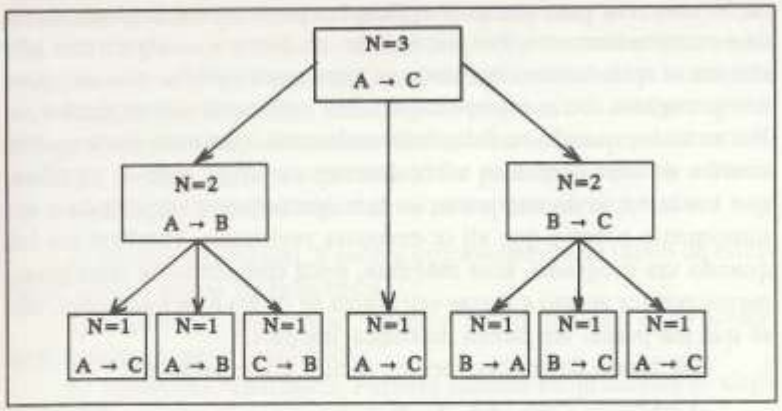

FIG. 2.3 Árvore de subproblemas para as Torres de Hanoi com N=3

3. CONHECIMENTO E SUA REPRESENTAÇÃO

Todas as pessoas 'normais' são capazes de executar algumas tarefas, mais ou menos especializadas. Para que isso seja possível, considera-se que elas tenham o *conhecimento* necessário. Por isso, se uma pessoa pode comportar-se inteligentemente ao fazer algo, diz-se que ela *conhece* o assunto.

Considera-se como conhecimento a informação a respeito da estrutura e das propriedades de objetos, a respeito de métodos de operação sobre eles, e a respeito de fatos ocorridos com eles. A natureza da inteligência e do conhecimento, do ponto de vista filosófico, psicológico ou educacional, tem sido desconsiderada a favor de uma conceituação mais pragmática, de uma abordagem operacional e de implementação de programas que *se comportam* de maneira inteligente, usando dados e estruturas de controle adequados, assim considerados como 'conhecimento' para a IA.

Uma das tarefas primordiais dos pesquisadores em IA tem sido a de desenvolver estruturas de dados e algoritmos de manipulação dos mesmos, suficientemente abstratos para permitir operações genéricas, e ao mesmo tempo suficientemente eficientes em sua implemen-

tação concreta para permitir aplicações particulares factíveis técnica e economicamente. Por outro lado, os dados e os algoritmos não são em si mais 'conhecimento' que uma enciclopédia: eles são apenas o registro ou a *representação* das estruturas correspondentes. No entanto, quando se fala, informalmente, que uma enciclopédia contém os conhecimentos sobre diversos assuntos, quer-se significar que um leitor pode comportar-se inteligentemente adquirindo o conhecimento sobre o que ali se encontra registrado. Também em IA, quando um programa, uma máquina, pode comportar-se inteligentemente por ter acesso a certas estruturas de dados e de operações, diz-se que ela possui um banco de conhecimentos.

Um banco de conhecimentos inclui, principalmente:

a) Objetos, e uma relação de fatos a respeito desses objetos.
b) Eventos e ações envolvendo objetos.
c) Habilidades para atuar sobre objetos.
d) Meta-conhecimento, informando o que é conhecido, e como pode ou deve ser aprendido e usado.

Os conhecimentos devem, pois, ser *adquiridos, recuperados e usados* pelo sistema de IA para que ele se comporte inteligentemente. Três tipos básicos de respostas podem ser extraídas de um sistema de IA: fatos, inferências e deduções. Eles podem ser exemplificados, de maneira simples, pelas perguntas seguintes:

1) Pombas são aves?
2) Todas as aves têm asas?
3) Pombas têm asas?

A pergunta 1) será respondida por consulta aos fatos registrados a respeito do objeto 'pomba'. A pergunta 2) pode ter dois modos de resposta: primeiro, consultando uma descrição do objeto genérico 'ave', se esta descrição estiver presente; ou consultando todos os objetos que tenham a propriedade de serem aves, e verificando se para todos eles consta o fato de terem asas. Neste segundo caso, a resposta seria obtida por inferência a partir dos dados individuais. A pergunta 3) também será respondida diferentemente, de acordo com o tipo de representação disponível: no caso de haver um objeto genérico, ou classe, chamado 'ave', ao qual deve pertencer o objeto 'pomba', então a resposta será obtida por dedução silogística, pois o fato 'ter asas' estará registrado no nível da classe 'ave' e não

no objeto 'pomba' pertencente a ela. No outro caso, se o fato de ter asas e o fato de ser ave estiverem na descrição de cada objeto pertinente, então a resposta à questão 3) será obtida diretamente por consulta.

É claro que diferentes maneiras de processar os dados disponíveis para torná-los conhecimento devem alternar-se e mesmo misturar-se na abordagem de um problema. Os principais tipos de raciocínio geralmente disponíveis em um ou outro sistema de IA são:

1) Raciocínio formal. Envolve processamento sintático de estruturas, usando regras de inferência.
2) Raciocínio procedural. Permite seguir modelos de processamento para chegar a resultados.
3) Raciocínio analógico. Permite deduzir propriedades de objetos por semelhança com outros objetos conhecidos.
4) Raciocínio hiponímico. Permite generalizar ou abstrair, formando classes.
5) Meta-raciocínio. Permite administrar o conhecimento, produzindo julgamentos sobre sua importância e sua qualidade.

Como se mostrou acima, os mecanismos usados para obter soluções dependem de como os dados estão representados. Por exemplo, nas perguntas sobre pombas e aves, alguns dados estão explícitos, outros devem ser obtidos a partir deles. Muito conhecimento explícito torna o sistema rápido em dar respostas, mas muito grande em termos de espaço de memória. Por outro lado, se muito conhecimento está implícito, exige-se menos espaço, mas leva mais tempo torná-lo acessível.

Outra diferença relevante é o modo como o conhecimento vai codificado. Ele pode ser predominantemente procedural, exigindo do codificador do conhecimento o domínio sobre os detalhes de manuseio dos dados, sua forma, e a seqüência de operações necessárias para obter a resposta. Ou pode ser predominantemente declarativo, quando o codificador especifica a estrutura dos dados e das relações formais entre eles, e o próprio sistema é responsável pelos detalhes do manuseio.

A descrição dos dados de uma família, por exemplo, pode ser feita em PROLOG de forma totalmente declarativa (Fig. 3.1). A 'máquina inferencial' com que essa linguagem é implementada fica encarregada dos detalhes do método de pesquisa quando se formulam e submetem perguntas ao programa.

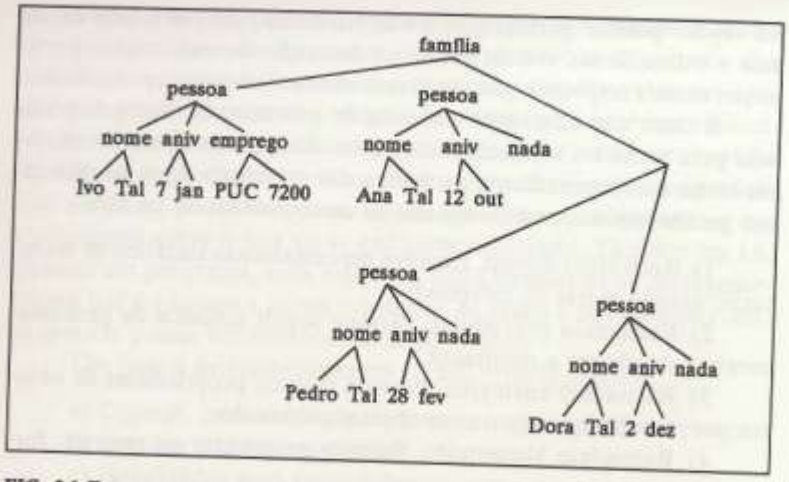

FIG. 3.1 Estrutura de dados de uma família

Mostra-se na Fig. 3.2 um programa codificado para esse problema, contendo fatos sobre algumas famílias e mais algumas relações. Deve-se observar que em PROLOG os simbolos que iniciam com maiúscula são variáveis, com minúscula são dados. Por isso os nomes próprios aparecem com inicial minúscula.

Algumas possíveis perguntas a esse programa são:

Pergunta pelos maridos de sobrenome Silva:

```
?- família(pessoa(nome(X,silva),-,-,),-,-).
```

Pergunta por quem seja pai de 3 ou mais filhos:

```
?- família(X,-,[-,-,-|-]).
```

Pergunta pelos filhos que fazem anos em dezembro:

```
?- filho(A),aniversário(A,aniv(-,dez)).
```

Pergunta por alguém que ganhe mais de dez mil:

```
?- existe(X),salário(X,Q), Q > 10000.
```



```
família ( pessoa(nome(ivo,tal), aniv(7,jan), emprego(puc,7200)),
          pessoa(nome(ana,tal), aniv(12,out), nada),
          [ pessoa(nome(pedro,tal),aniv(28,fev),nada),
            pessoa(nome(dora,tal),aniv(2,dez),nada)
          ] ).
familia(
 pessoa(nome(joaquim,tal),aniv(17,jun),emprego(simpala, 8100),
 pessoa(nome(maria,tal), aniv(25,set), emprego(simpala, 4320),
 [ ]  ).
familia(
 pessoa(nome(antonio,qual), aniv(27,jan), emprego(puc,7900)),
 pessoa(nome(ana,qual), aniv(22,ago), emprego(puc,12700),
 [ pessoa(nome(enio,qual),aniv(28,mai),emprego(dosul,5300)),
   pessoa(nome(flora,qual),aniv(27,ago),nada)
   pessoa(nome(dario,qual),aniv(9,dez),nada)
   pessoa(nome(manoel,qual),aniv(2,abr),nada)
 ] ).
marido(X) :- familia(X,-,-). % X é pessoa que é o marido
esposa(Y) :- familia(-,Y,-). % Y é a pessoa que é a esposa
filho(Z) :- familia(-,-,Filhos), membro(Z.Filhos).
aniversário(pessoa(-,Data,-),Data).
salário(pessoa(-,-,emprego(-,S)),S).
salário(pessoa) :- marido(Pessoa); esposa(Pessoa); filho(Pessoa).
      % Qualquer pessoa de alguma família
membro(E,[E|Resto]).   % É membro se for o primeiro
membro(E,[X|Resto]) :- membro(E,Resto).
                       % ou se for membro do resto da lista
```

FIG. 3.2 Programa representando fatos e relações em PROLOG

4. REDES SEMÂNTICAS

Uma das mais bem sucedidas formas de representar o conhecimento é a rede semântica. Ela reúne vários formalismos importantes da área da IA.

Uma rede, ou grafo, ou matriz, é uma entidade matemática consistindo de *objetos* (conceitos, situações, etc.) ligados por *relações*. Os objetos são chamados *nodos* e as relações são ditas *arcos*. Um exemplo aparece na Fig. 4.1.

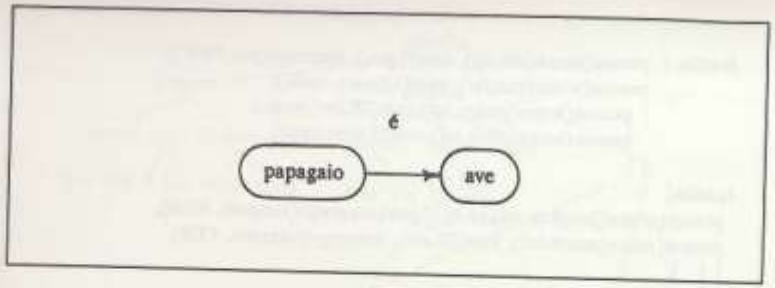

FIG. 4.1 Relação entre objeto

Novos objetos e/ou novas relações podem facilmente ser acrescentadas a essa rede (Fig. 4.2). Neste caso, 'papagaio' e conseqüentemente 'Louro' *herdam* o conhecimento de que 'ave' 'tem' 'asas'. Mais será dito sobre isso mais adiante, ao se tratar de *molduras e roteiros* (secção 5).

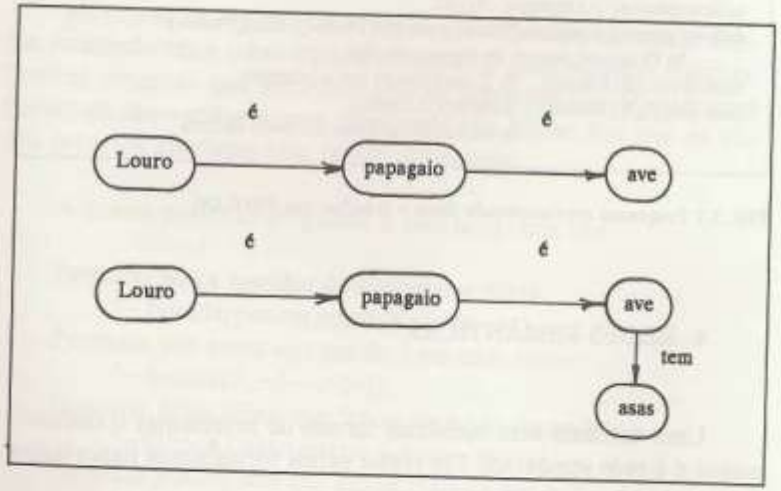

FIG. 4.2 Novos objetos e novas relações acrescentadas

Consideram-se *primitivas semânticas* (WILKS, 78) todos os termos que são usados sem definição dentro de um sistema, e é por isso que são ditas 'primitivas'. Têm uma acepção semelhante em todos os sistemas de compreensão da linguagem natural. Além disso, são primitivas no sentido de que todos os outros elementos do sistema devem ser decomponíveis em termos deles. A ligação entre as primitivas semânticas para formar sentidos de palavras ou frases forma uma rede semântica, mas tem muitas das características das *molduras*. No trabalho de Wilks, cinco classes de primitivas são usadas:

1) Entidades (homem, coisa, parte,...)
2) Ações (causar, ser, fluir, ...)
3) Casos (para, em, sobre, através, ...)
4) Qualificadores (bom, belo, claro, ...)
5) Tipos (como, modo, ...)

As considerações gerais que se podem fazer ao escolher um conjunto de primitivas semânticas levam aos seguintes requisitos (Wilks, 78):

1) Finitude. O conjunto deve ser pequeno comparado com o tamanho do dicionário com ele analisado.
2) Completude. O conjunto deve ser adequado e suficiente para o tipo de distinções e expressões a que se destina.
3) Independência. Primitivas não podem ser definidas em termos de outras primitivas.
4) Linearidade. A estrutura não deve conter laços ou ciclos de referências mútuas.
5) Elementaridade. Nenhum subconjunto de primitivas deve poder ser substituído por outro subconjunto menor.

Com este modelo pode-se, por exemplo, decompor o termo 'beber', como exemplificado na Fig. 4.3, onde se usou a linguagem LISP. Ali, segundo o modelo de Wilks, as caracterizações dos elementos são tidos como preferenciais, não taxativas. Assim, "meu carro bebe muita gasolina", ou "ao sair da água João bebeu o ar a plenos pulmões" podem ser perfeitamente decompostas segundo o modelo, mantendo as características de *ANI (classe de seres animados) e LIQU (líquido) como preferenciais apenas, nos respectivos contextos.

```
[BEBER]:   ( (SUJ *ANI)
             [INGERIR]
             (OBJ (COISA LIQU)) )

[INGERIR]: ( (SUJ *ANI)
             (CAUSAR)
             ((OBJ COISA)
              ((ESTAR DENTRO)
               (ESTE *ANI)
               (ATRAVES (PARTE (ESTE *ANI)))
            ) ) )
```

FIG. 4.3 Primitivas semânticas expressas em LISP

Outro modelo para a representação do significado é a teoria da *dependência conceitual* (SCHANK, 75). Esta teoria se propõe a ser ao mesmo tempo um mecanismo de processamento computacional e uma teoria sobre o processamento humano da linguagem. Os requisitos básicos para essa representação são:

a) não ambigüidade;
b) unicidade.

A desambiguação é obtida pela representação do significado mais provável. A unicidade permite que sentenças como:

"Quero um livro",
"Quero obter um livro",
"Quero ter um livro",

tenham a mesma representação interna nesse sistema.

A teoria da dependência conceitual é um instrumento muito forte para lidar com atos físicos, mas tem grandes problemas com atos proposítais e modais.

5. MOLDURAS E ROTEIROS

Uma *moldura* caracteriza o conjunto de entidades relacionadas com um dado *ambiente de comportamento*. Por exemplo, a moldura

de uma 'sala de aula': alunos, professor, giz, quadro-negro, livros, cadernos, estudo, etc.

Um *roteiro* caracteriza uma seqüência temporal de eventos que se desenrolam sistematicamente em determinadas situações. Por exemplo, o roteiro de 'pegar o ônibus': cuidar os veículos que se aproximam, acenar para o motorista, esperar o ônibus parar e abrir a porta, subir, pagar a passagem e passar pela roleta, procurar um lugar, sentar, etc. O roteiro deve ter acesso, obviamente, a todas as molduras por ele referidas.

Grande evidência foi acumulada pelos psicólogos de que o conhecimento humano, enquanto norma de comportamento, é composto principalmente de molduras e roteiros. Algumas dessas molduras estão tão arraigadas no contexto cultural que é preciso muito esforço de análise para isolá-las e explicitá-las. É o caso da roupa, por exemplo. Igualmente, muitos dos roteiros mais importantes se constituem em hábitos e mesmo instintos, passando desapercebidos como representações de conhecimento. Por exemplo, os movimentos que compõem os atos de escrever, caminhar e mastigar.

Assim, para que um programa, ou uma máquina, possa comportar-se (quase) tão inteligentemente como uma pessoa, seu 'conhecimento' deve ser estruturado em molduras e roteiros. Além, é claro, das informações básicas: léxico, sintaxe, redes semânticas, etc.

Uma das maneiras de testar 'compreensão' por parte de uma máquina, é o nível de sua tradução para outra língua. Isso pode ser conseguido gerando-se uma representação 'pura', no sentido de independente da língua de origem, como a rede de primitivas semânticas ou o sistema de dependência conceitual. Sua recodificação para a mesma língua original gera paráfrases; para outras línguas, gera traduções. A qualidade dessas traduções mede o grau de compreensão da máquina (WILKS, 78).

Outra maneira de testar essa 'compreensão mecânica' é a resposta a questões. Pois, para que a resposta seja coerente, é preciso ter captado o sentido da pergunta.

Exemplos de molduras podem ser construídos a partir de objetos usuais:

moldura CADEIRA:

hipônimo_de: MÓVEL
num_de_pernas: um inteiro (por omissão 4)
encosto: um estilo (por omissão reto)
num_de_braços: de 0 a 2 (por omissão 0)

moldura CADEIRA_DO_PEDRO:

hipônimo_de: CADEIRA
num_de_pernas: 4
encosto: estofado
num_de_braços: 0

moldura RESTAURANTE:

hipônimo_de: CASA_COMERCIAL
tipo: um de [lancheria,bufê,serviço,pizzaria,churrascaria)
local: endereço
nome: nome_comercial
comida: um de [chinesa, francesa, árabe, frutos_do_mar]
horario: de hora_abre até hora_fecha
pagamento: um de [dinheiro,cartão,cheque,lava_pratos]
funcionamento: roteiro COMER_FORA
alternativas: lista_de_restaurantes (mesma comida)

moldura PASSOQUIM:

hipônimo_de: RESTAURANTE
tipo: lancheria
local: ali na esquina
nome: Shuan Yin & Filhos Ltda.
comida: chinesa
horário: das 11 às 3 horas
pagamento: um de [dinheiro,cheque]
funcionamento: roteiro COMER_FORA
alternativas: lista_de_restaurantes(comida chinesa)

roteiro COMER_FORA:

pessoas: freguês, garçom, caixa
elementos: RESTAURANTE, MESA, MENU, ALIMENTO, CONTA, PAGAMENTO, GORJETA
eventos:

1. freguês entra
2. freguês senta à MESA
3. garçom, traz MENU
4. freguês faz pedido
5. garçom, serve ALIMENTO
6. freguês come
7. garçom, traz CONTA
8. freguês deixa GORJETA
9. freguês dá PAGAMENTO ao caixa
10. freguês sai

início: evento 1
objetivo: evento 6

Os exemplos acima encontram-se bastante simplificados em relação às vivências e expectativas de uma pessoa. Mas contêm informações suficientes para diversas aplicações significativas em IA. Por exemplo, para 'compreender' o trecho: "Fui ao restaurante. Comi um baurú e um sorvete.", o sistema de IA pode completar as relações de significado com o roteiro de COMER_FORA e a ali citada moldura MENU.

Molduras e roteiros podem organizar-se em estruturas maiores, como planos, metas e temas. Planos são esquemas genéricos de roteiros. Metas são os objetivos de planos, e temas são estruturas que contêm os dados necessários para especificar metas. Uma história como

"Pedro estava com fome. Precisava de dinheiro para comer. Pegou uma arma e foi à farmácia."

não será 'compreendida' a não ser que se tenha a descrição de um plano. Aqui, 'compreender' significa achar elementos de coerência textual.

Numerosas variantes existem para as teorias até aqui descritas, as primitivas semânticas (WILKS, 78) e a teoria de dependência conceitual (SCHANK, 80). E muitos são os sistemas de IA que implementam essas e outras idéias (BARR, 81).

As *linguagens formais*, como propostas por Chomsky e usadas no projeto das linguagens de programação, têm algumas características que as tornam igualmente úteis à análise das linguagens naturais. Elas se compõem de:

1) Variáveis (ou categorias sintáticas), como <SENTENÇA>, <SUJEITO>, <ADJUNTO ADVERBIAL>;
2) Constantes (ou símbolos terminais), como sinais de pontuação ou itens lexicais.
3) Regras de produção como
<SENTENÇA> --> <SINTAGMA NOMINAL> <SINTAGMA VERBAL> <SINTAGMA NOMINAL> --> <ARTIGO> <NOMES>
4) Um símbolo sentencial, que é a variável de mais alto nível, como <SENTENÇA>, acima.

As linguagens formais agrupam-se segundo a hierarquia de Chomsky em quatro tipos: regulares, livres de contexto, sensíveis a contexto e irrestritas. A complexidade das regras, e o esforço computacional necessário para manuseá-las é significativamente diferente para cada tipo. Também o ferramental matemático necessário para prover o tratamento formal de cada tipo é significativamente mais elaborado de um tipo para outro. Essas classes estão contidas propriamente umas nas outras.

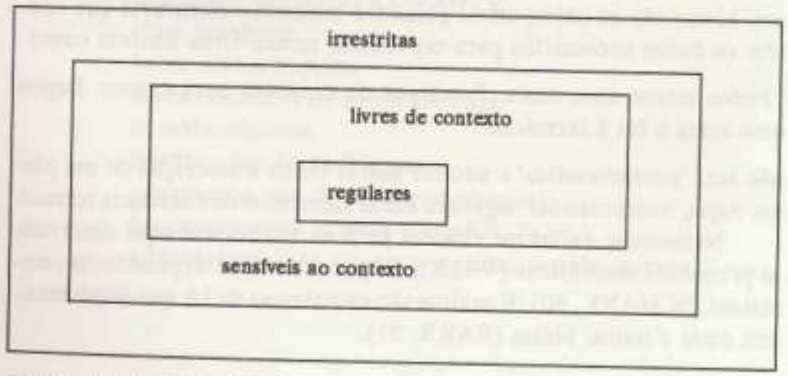

FIG. 6.1 As categorias de Chomsky para linguagens

As gramáticas que 'produzem' essas linguagens são por isso mesmo chamadas 'gerativas'. Cada categoria é capaz de gerar a classe correspondente, e as nela contidas. As gramáticas regulares e livres de contexto são insuficientes, por exemplo, para a partir delas produzir-se frases como: "Artur, Pedro e Paulo são os maridos de Maria, Joana e Ester, respectivamente". Por outro lado, as gramáticas sensíveis ao contexto se tornam ineficiente, por introduzirem ambigüidade e por conterem regras complexas e de difícil leitura. Além disso, não podem responder pelo significado igual contido em sentenças de forma superficial diferente.

A partir dessas constatações, surgiram as regras transformacionais. A gramática completa terá três classes de regras:

1) Regras gerativas, produzindo as sentenças simples e afirmativas ao nível das gramáticas livres de contexto;
2) Regras transformacionais, permitindo entre outras coisas, reescrever os elementos em outra ordem ou apagar alguns itens, o que corresponde a regras do nível irrestrito de gramáticas;
3) Regras morfológicas, para ajustar os itens presentes a sua forma fonológica (e ortográfica) correta.

Redes expandidas de transição, ou ATN (Augmented Transition Networks) são formas de representação para gramáticas de linguagens naturais, em substituição às gramáticas formais (WOODS,70). São *grafos* do tipo redes recursivas de transição, de que são um incremento. Estão baseadas nos mesmos autômatos finitos que emergem do processamento das gramáticas regulares, mas com o acréscimo de testes adicionais e efeitos colaterais, na forma de ações apostas a seus arcos. Com tal expansão, esses mecanismos tornam-se suficientemente poderosos para processar as linguagens naturais até o grau de manuseio exigido pelas tarefas que se esperam atualmente dos computadores, nesta área. Exemplo:

Os acréscimos feitos às redes recursivas para torná-las ATN incluem:

1) Registros de informações sobre *árvores de derivação* parciais entre saltos para diferentes sub-redes;
2) Os arcos da rede, além de conter nomes de classes gramaticais ou estruturas sintáticas, podem ter associados a eles testes arbitrários que devem estar satisfeitos antes que eles possam ser seguidos numa transição;

3) Ações especiais podem ser apostas a um arco, para que sejam executadas toda vez que aquele arco é seguido.

Formalmente, esses acréscimos são suficientes para tornar os mecanismos a recursivos dessas redes tão poderosos como os de uma *máquina de Turing*, que é um mecanismo formal capaz de computar tudo o que é computável (AHO, 73:29), correspondendo ao nível irrestrito de gramáticas.

Ao contrário das gramáticas irrestritas, que são de processamento ineficiente, as ATN são meios práticos de implementação dos métodos de geração e reconhecimento de linguagens naturais (BARR, 81:266, vol.1).

Outra vantagem das ATN é a possibilidade de incluir em sua representação diversas características que teoriamente pertencem a diferentes formas de gramáticas. Elas podem ser usadas, entre outras coisas, para testar novas idéias sobre as teorias gramaticais, existentes ou sob pesquisa.

Os testes e as ações apostos aos seus arcos tornam as ATN muito bem apetrechadas para manipular os encargos das gramáticas transformacionais: casos especiais e exceções, tão bem como as regras gerais, estão sob perfeito controle. A principal desvantagem das ATN quando aplicadas ao processamento da linguagem natural, é sua dificuldade em manipular enunciados agramaticais, embora significativos. Isso decorre de sua forte dependência das estruturas sintáticas.

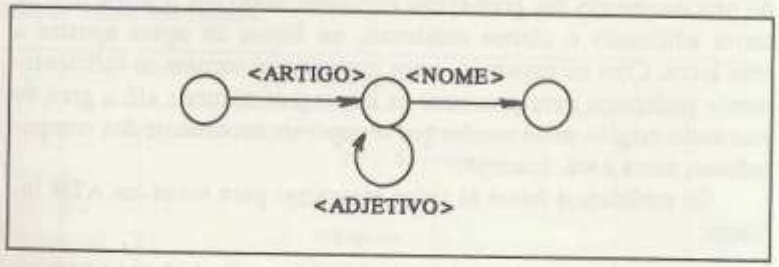

FIG. 6.2 Uma ATN recursiva simples

A implementação desses princípios depende fortemente do sistema computacional, sua capacidade de dados e sua velocidade, e também da linguagem em que ela é feita. Dialetos especializados da linguagem PROLOG foram usados com sucesso em máquinas de grande porte. As relações entre elementos da linguagem, sejam variáveis ou constantes, são expressas em forma de equações, e mostram a estrutura das respectivas redes. Por exemplo, a representação dos componentes semânticos de um pronome anafórico, na linguagem STUF (BOUMA, 88), pode ser feita como é mostrado na Fig. 6.3.

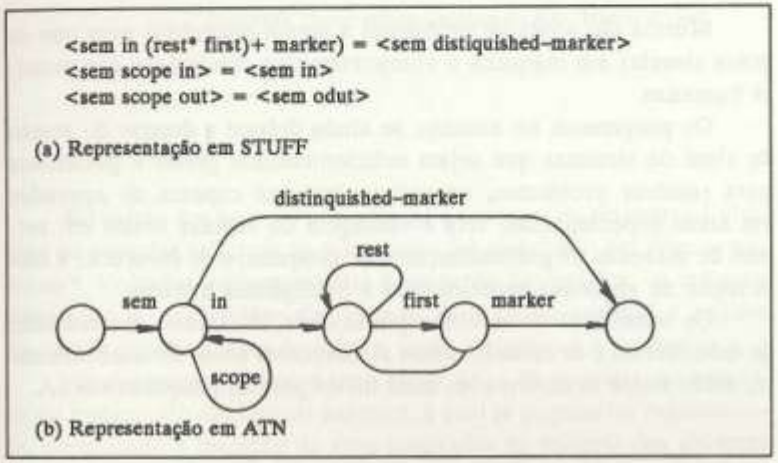

FIG. 6.3 Componente semântico da anáfora

Em estruturas desse tipo, a resolução anafórica fica expressa como uma relação de pertinência de um antecedente a uma rede vista como um conjunto de antecedentes e de condições correspondente à parte *(rest* first)+* da estrutura acima. Ali os valores dos atributos de *in* e de *out* servem para ligar informações contextuais, contidas na rede semântica, ao resto da árvore sintática. Essas ligações entre partes das estruturas são obtidas por herança de valores de nodos irmãos e também a partir de relações de hiponímia entre os itens lexicais. Os valores para *scope* são preenchidos a partir das regras de dominância na árvore sintática, o que permite computar o valor da anáfora.

Quando se usa uma linguagem declarativa como STUF ou PROLOG, procurar um membro de uma estrutura é o mesmo que prever

a existência de um membro com as mesmas características. Isso reflete uma importante propriedade desse tipo de formalismo, o princípio de *unificação* usado nas linguagens declarativas, em contraste com a *atribuição,* como usada nas linguagens procedurais de programação (BOUMA, 88).

6. CONCLUSÃO

Muitos são ainda os problemas a serem resolvidos para que se possa simular em máquina o comportamento inteligente das pessoas humanas.

Os progressos, no entanto, se ainda deixam a desejar do ponto de vista de sistemas que sejam suficientemente gerais e poderosos para resolver problemas, ou suficientemente capazes de aprender em áreas especializadas, têm a vantagem de ensinar muito em termos de métodos de generalização e de pesquisa, e de favorecer a elaboração de melhores modelos para a inteligência humana.

Os benefícios da interdisciplinaridade, do salutar intercâmbio de descobertas e de modelos entre as diferentes áreas do conhecimento, estão entre as motivações mais fortes para as pesquisas em IA.

BIBLIOGRAFIA

AHO, D. & ULLMAN, J. *The Theory of Parsing, Translating and Compiling.* Englewood Clifs, Prentice-Hall Inc., 1972.

BARR, A. & FEIGENBAUM, E.A. *The Handbook of Artificial Intelligence.* Los Alamos, William Kaufmann Inc., 1981.

BOUMA, KÖNIG & USZKOREIT. A flexible graph-unification formalism and its application to natural-language processing. *IBM Journal of Research and Development.* Armonk, (32)2:170-184, mar. 88.

BRATKO, I. *Prolog Programming for Artificial Intelligence.* Cambridge, 1986.

SCHANK, R.C. Conceptual Dependency: A theory of natural language understanding. *Cognitive Psychology* (1972)3:552-631.

WILKS, Y.A. Making preferences more active. *Artificial Intelligence* (1978)11:197-223.

WOODS, W.A. What's in a link: Foundations for semantic networks. In Bobrow & Collins, *Representation and understanding studies in cognitive science.* N. York, Academic Press, 1975:35-82.