1. Publicação nº	2. Versão	3. Data	5. Distribuição
INPE-4082-TDL/250		*Dez. 1.986*	☐ Interna ☒ Externa ☐ Restrita

4. Origem	Programa
DIN	*INTAL*

6. Palavras chaves - selecionadas pelo(s) autor(es)

ORGANIZAÇÃO *SISTEMAS ESPECIALISTAS*
ARQUITETURA *INTELIGÊNCIA ARTIFICIAL*

7. C.D.U.: *681.3.019*

8. Título *INPE-4082-TDL/250*

ARQUITETURA DE SISTEMAS ESPECIALISTAS

10. Páginas: *73*

11. Última página: *67*

12. Revisada por

Paulo Ouvera Simoni

9. Autoria *Edson Luiz França Senne*

Assinatura responsável

13. Autorizada por

Marco Antonio Raupp
Diretor Geral

14. Resumo/Notas

Este trabalho aborda a arquitetura de sistemas especialistas considerando-a de uma forma ampla, ou seja, levando em conta a estrutura (módulos componentes e suas comunicações) e a organização interna (estrutura de cada módulo) destes sistemas. Apontam-se as principais características dos sistemas especialistas e as diferenças entre eles e os programas tradicionais. Apresentam-se, como fundamento para sua construção, requisitos básicos como esquemas de representação de conhecimento e técnicas de resolução de problemas. Discutem-se as arquiteturas de alguns sistemas especialistas importantes já desenvolvidos e as prescrições organizacionais que devem orientar o desenvolvimento de um sistema especialista, em função da complexidade do problema e do conhecimento disponível para resolvê-lo. Conclui-se apresentando as tendências atuais com relação ao estabelecimento de novas arquiteturas, os estudos que vêm sendo feitos de forma a cobrir capacidades ainda não-modeladas pelos sistemas atuais e algumas características necessárias aos futuros sistemas especialistas.

15. Observações *Este trabalho é resultado do Exame Integrado em Computação Aplicada, feito pelo autor em 28/08/86 no Instituto de Pesquisas Espaciais.*

ABSTRACT

The architecture of expert systems is discussed considering it in a broad way, that is, taking into account the structure (constituent modules and their relationships) and the internal organization (structure of each module) of these systems. Expert systems' main characteristics and how they distinguish from traditional programs are noted. Basic requirements as knowledge representation schemes and problem solving techniques for building such systems are presented. Architectures of some important and already developed expert systems are discussed, as well as organizational prescriptions which must guide the development of expert systems depending on the problem complexity and available knowledge to solve it. Present trends concerning establishment of new architectures, studies that have been made in order to achieve some uncovered capabilities by current systems, and some presumed characteristics of future expert systems are presented as conclusion.

SUMÁRIO

LISTA DE FIGURAS

CAPÍTULO 1

INTRODUÇÃO

Inteligência Artificial é a área da Ciência da Computação que se preocupa em construir programas que exibem comportamento inteligente, isto é, exibem características que normalmente são associadas com inteligência no comportamento humano.

O paradigma atual para a construção destes programas é o de *sistemas baseados em conhecimento*. Assim, um sistema de visão computacional deve ter conhecimento sobre os tipos de objetos que ele "vê"; sistemas de entendimento de linguagem natural devem possuir conhecimentos sobre o universo de discurso para poder compreender as frases e sentenças que eles processam; sistemas de diagnóstico médico necessitam conhecer as características das doenças para poder diagnosticá-las e sugerir terapias apropriadas.

A construção destes sistemas fez emergir da inteligência artificial uma área separada de estudo - *representação de conhecimento*- que se preocupa com duas questões fundamentais: como armazenar e como manipular o conhecimento que tais sistemas necessitam para se comportarem inteligentemente.

Uma classe de sistemas baseados em conhecimento vem ganhando, nos últimos anos, grande destaque: a dos *sistemas especialistas*. Estes sistemas são caracterizados como "especialistas" porque os problemas que eles resolvem são difíceis e requerem conhecimento especializado, ou seja, requerem o conhecimento de pessoas com grande proficiência no domínio do problema.

Por esta caracterização de sistemas especialistas, amplamente aceita e difundida, pode-se ter a impressão que eles são os únicos programas que alcançam altos níveis de desempenho. Por certo, por definição eles *precisam* ter bom desempenho. Mas na verdade, muitos pro

gramas de aplicação utilizam conhecimento especializado e também alcançam altos níveis de desempenho (Nau, 1983).

A diferença principal entre tais programas é que nos sistemas especialistas o conhecimento específico para a resolução do problema é estabelecido como uma entidade separada *(base de conhecimento)* ao invés de aparecer implicitamente como parte da codificação do programa, misturado portanto com as instruções de controle. Nos sistemas especialistas, a base de conhecimento é manipulada por uma estrutura de controle também separada e claramente identificável.

Essa arquitetura básica dos sistemas especialistas, que estabelece uma separação muito nítida entre a base de conhecimento e os procedimentos encarregados de explorá-la, é conseqüência de uma série de razões (Hayes-Roth et alii, 1983), dentre as quais se pode destacar o fato de que para muitos problemas interessantes não se conhece uma solução algorítmica tratável porque se originam em contextos complexos para os quais não é possível uma descrição precisa e uma análise rigorosa. Para tais problemas é mais adequado um tratamento heurístico (Buchanan, 1985) utilizando o conhecimento disponível para fazer inferências plausíveis. Numa abordagem deste tipo, tal arquitetura é interessante uma vez que, flexível, permite que o conhecimento seja alterado e estendido facilmente. Esta característica é muito importante, especialmente na fase inicial da construção de uma base de conhecimento.

Essa, no entanto, não é a única diferença entre os sistemas especialistas e os programas tradicionais. Em geral, requer-se (Hayes-Roth, 1984a) que um sistema especialista:

- apoie seu processo de raciocínio na manipulação de símbolos;

- interaja de maneira apropriada com seus usuários;

- funcione com dados incorretos e conhecimento incerto;

- explique o por quê de suas perguntas;

- justifique suas conclusões.

Ou seja, os sistemas especialistas diferem dos programas convencionais na maneira em que são estruturados, incorporam conhecimento, são executados e na impressão que causam através de suas interações.

Muitos sistemas especialistas já foram desenvolvidos e atuam em domínios, tais como: diagnóstico médico, síntese química, exploração mineral, formação de conceitos matemáticos, análise de circuitos elétricos, diagnóstico de falhas em computadores, configuração de sistemas de computação, análise estrutural, ensino e mesmo concepção destes sistemas.

A Figura 1.1 ilustra a estrutura básica de um sistema especialista.

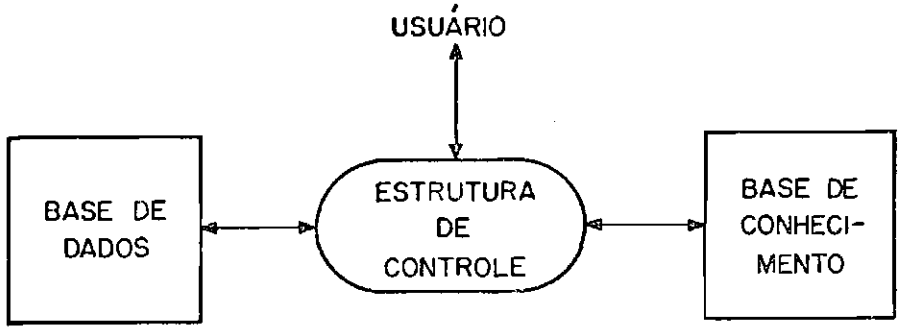

Fig. 1.1 - Estrutura básica de um sistema especialista.

As funções dos componentes dessa estrutura são as seguintes:

(a) *base de conhecimento:* armazenar fatos e heurísticas associadas ao domínio de aplicação;

(b) *base de dados global* (ou *memória de trabalho*): armazenar dados para um problema particular, resultados intermediários e histórico do processamento efetuado;

(c) *estrutura de controle* (ou *motor de inferência*): decidir sobre como utilizar o conhecimento disponível para resolver o problema.

O número e a diversidade de sistemas especialistas já desenvolvidos (Waterman, 1986) indicam que esta área está amadurecendo rapidamente apesar de ainda ser considerada experimental. Poucas técnicas de sistematização e princípios organizacionais existem atualmente para a construção de um sistema especialista, o que exige ainda muito trabalho criativo. Há entretanto uma grande preocupação em estabelecer prescrições de como organizar apropriadamente um sistema especialista, de modo a refletir o tipo e a complexidade do problema e a forma do conhecimento necessário para resolvê-lo; para isto uma nova área de estudo vem ganhando corpo: a *engenharia de conhecimento* (Hayes-Roth, 1984b).

Mesmo com bastante experiência no desenvolvimento de sistemas especialistas é difícil, segundo Nii e Aiello (1979), determinar se um método de resolução de problema é "especial" a um domínio particular ou se é fácil generalizá-lo para outros domínios. É também difícil formular um modelo conceitual que auxilie a determinar que fatos e heurísticas do domínio são relevantes e devem ser incluídos na base de conhecimento de um sistema especialista. Estas dificuldades realçam a importância do engenheiro de conhecimento (Haley and Williams, 1986).

Tipicamente o engenheiro de conhecimento analisa o problema a ser resolvido e adota uma arquitetura para o sistema especialista que reflete escolhas específicas sobre os tipos de conhecimento a representar, quais formalismos usar, os tipos de inferências a fazer e o tipo de flexibilidade a permitir. Adota também uma estratégia de solução que determina qual conhecimento aplicar e em que ordem. Ou se

ja, para determinar a arquitetura apropriada, o engenheiro de conhecimento levanta questões sobre a complexidade do problema a resolver (e em conseqüência, a complexidade envolvida na solução) e o tipo de conhecimento disponível; com base nas respostas destas questões, adota uma estrutura e organização gerais para o sistema. Melhorias e refinamentos são então introduzidos para moldar o sistema às características do domínio e para produzir o nível de desempenho desejado.

Alguns sistemas baseados em conhecimento já foram desenvolvidos para auxiliar em parte essas tarefas (Nii and Aiello, 1979; Reggia, 1980; Davis, 1982; Bennett, 1985).

A Figura 1.2 (Buchanan et alii, 1983) mostra os estágios de desenvolvimento de um sistema especialista.

De uma forma geral, para desenvolver um sistema especialista é preciso especificar o *conhecimento estrutural* (identificação de termos, conceitos, relacionamentos e associações entre conceitos e determinação dos esquemas de representação) e o *conhecimento estratégico* (identificação das técnicas básicas de como usar o conhecimento estrutural para resolver o problema) sobre o domínio de aplicação.

Evidentemente outras tarefas podem ser acrescentadas a essas duas como, por exemplo, a especificação das estruturas necessárias para facilitar o processo de aquisição de conhecimento pelo sistema, ou então das estruturas necessárias para que o sistema seja capaz de fornecer aos seus usuários boas justificativas para suas ações e explicações para suas conclusões.

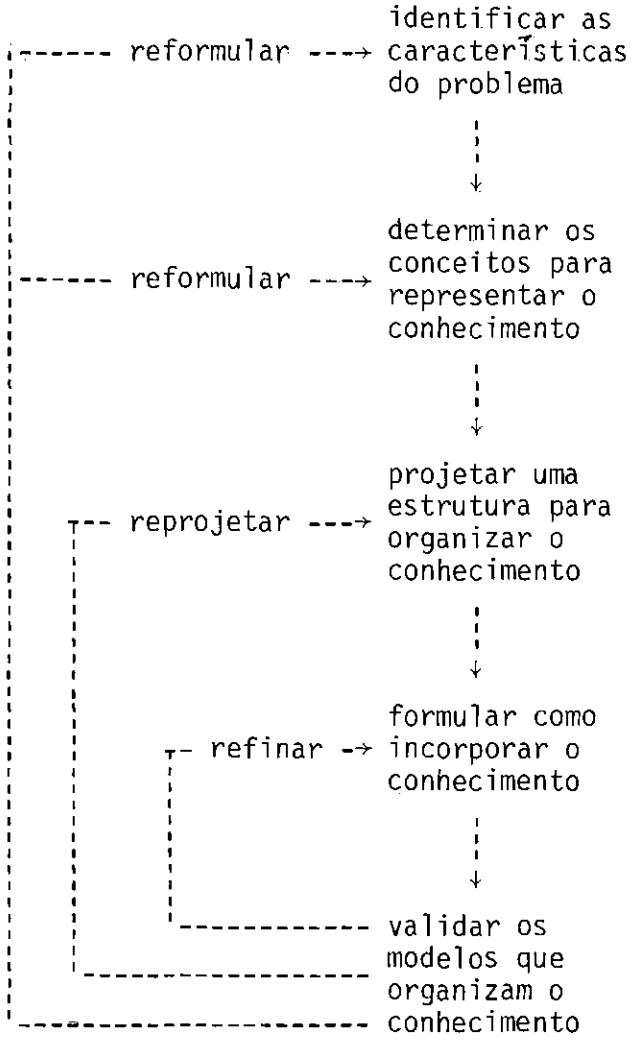

Fig. 1.2 - Estágios de desenvolvimento de um sistema especialista.

Apesar da importância das capacidades de aquisição de conhecimento e de explanação - alguns autores chegam mesmo a apontar esta última como a principal característica dos sistemas especialistas, segundo Brachman et alii (1983) - este trabalho aborda a arquitetura dos sistemas especialistas considerando-os na sua *estrutura básica* (Figura 1.1), ou seja, como constituídos essencialmente de uma *base de conhecimento* e uma *base de dados* (que armazenam o conhecimento estrutural) e de um *motor de inferência* (que incorpora o conhecimento estratégico).

Para isso, no Capítulo 2 discutem-se os fundamentos para a organização de um sistema especialista: abordam-se os principais esquemas de *representação de conhecimento* utilizados para a organização das bases de conhecimento e de dados, e técnicas de *resolução de problemas* que indicam maneiras de como organizar um motor de inferência. O Capítulo 3 mostra tipos importantes de arquiteturas de sistemas especialistas, considerando para discussão os sistemas MYCIN, PROSPECTOR e HEASAY-II. No Capítulo 4 apontam-se, em função das características de problemas, prescrições organizacionais a serem seguidas no desenvolvimento de um sistema especialista e, finalmente, no Capítulo 5 conclui-se apresentando as tendências atuais com relação ao estabelecimento de novas arquiteturas e os estudos que vêm sendo feitos de forma a cobrir algumas capacidades ainda não modeladas por sistemas especialistas.

CAPÍTULO 2

FUNDAMENTOS PARA ENGENHARIA DE CONHECIMENTO

Admite-se de forma unânime que um sistema especialista é um programa destinado a resolver uma certa categoria de problèmas (aqueles para os quais ainda não se dispõe de algoritmos de resolução) e que, para isto, utiliza uma grande quantidade de conhecimento específico ao domínio em questão. Admite-se também que num sistema especialista a base de conhecimento deve estar separada dos mecanismos que a utilizam.

Não existe uma maneira padrão de construir a base de conhecimento de um sistema especialista, ou seja, um padrão para representar os objetos de um domínio de conhecimento. Tampouco, mesmo fixando-se numa só representação, existe uma maneira única de explorar o conhecimento. Em geral, somente a aplicação poderá especificar que tipos de representação de conhecimento são adequados e que estratégias são mais apropriadas.

Neste capítulo discutem-se, sucintamente, os diferentes esquemas de representação de conhecimento e as várias técnicas de resolução de problemas, material básico para que um engenheiro de conhecimento, em vista de uma aplicação, possa escolher uma arquitetura adequada para o desenvolvimento de um sistema especialista eficiente.

2.1 - REPRESENTAÇÃO DE CONHECIMENTO

Os primeiros esforços no desenvolvimento de esquemas de representação de conhecimento surgiram no final dos anos 60 e início dos 70 devido à mudança de orientação que foi dada à pesquisa em inteligência artificial após o insucesso experimentado pelas estratégias de resolução de problemas de propósito geral (Newell and Simon, 1963), as quais se mostraram extremamente limitadas.

O problema básico da representação de conhecimento é o desenvolvimento de uma notação suficientemente precisa (estrutura de dados e procedimentos interpretativos), a qual possa ser usada num programa com o intuito de levá-lo a exibir um comportamento inteligente.

Uma grande variedade de maneiras de representar conhecimento já foi proposta e vem sendo explorada nos últimos anos (Brachman and Smith, 1980; Barr and Feigenbaum, 1981; Rich, 1983; Chouraqui et alii, 1985). Embora exista uma preocupação com a validade psicológica dos vários esquemas propostos - representação de conhecimento e psicologia cognitiva compartilham preocupações fundamentais (Norman and Rumelhart, 1975) - ainda não é possível mostrar que um esquema representa aspectos da memória humana melhor do que outros, ou seja, não existe ainda uma *teoria* de representação de conhecimento.

Os esquemas de representação mais importantes atualmente são: lógica de predicados, representações procedimentais, redes semânticas, sistemas de produção e "frames".

LÓGICA DE PREDICADOS. A característica mais importante da lógica de predicados de primeira ordem é que as deduções são garantidamente corretas (o conjunto de inferências ou conclusões que pode ser obtido a partir de sentenças lógicas é completamente especificado pelas regras de inferência); portanto, a base de conhecimentos de um sistema especialista pode ser mantida logicamente consistente. A utilidade da lógica como técnica de representação de conhecimento tornou-se evidente durante os anos 60, principalmente devido à pesquisa em prova automática de teoremas (Nilsson, 1971). Muitos trabalhos foram feitos investigando o uso do princípio da resolução (Robinson, 1965) como técnica de inferência e muitos programas foram desenvolvidos usando, com sucesso, este princípio.

REPRESENTAÇÕES PROCEDIMENTAIS. A idéia de representação procedimental surgiu da tentativa de codificar explicitamente o controle do processo

de prova de teoremas. Destas pesquisas surgiram, dentre outras, as linguagens de programação PLANNER e SAIL (Bobrow and Raphael, 1974) e, mais recentemente, a linguagem PROLOG (Kowalski, 1979). Muitas representações procedimentais foram também influenciadas pelo LISP (Winston and Horn, 1981), uma linguagem ainda muito utilizada para implementar sistemas inteligentes. Numa representação procedimental o conhecimento está contido em *procedimentos*, pequenos programas que "sabem" como proceder em situações bem especificadas. A dificuldade de verificar e alterar o conhecimento representado procedimentalmente (isto é, dificuldade de utilização) e, por outro lado, o fato de que qualquer sistema de computação utiliza uma representação desse tipo (em algum nível de sua operação), levantou no passado uma grande controvérsia (Winograd, 1975) sobre as vantagens e desvantagens da representação procedimental, em relação às demais representações (declarativas).

REDES SEMÂNTICAS. A rede semântica foi desenvolvida no final dos anos 60 como um modelo psicológico da memória associativa humana. Atualmente existem muitos tipos de redes semânticas (Findler, 1979) mas, basicamente, elas consistem em: (a) uma estrutura de nós (representando objetos, conceitos, ...) e de ligações entre nós representando seus relacionamentos) e, e(b) um conjunto de procedimentos especializados que operam sobre esta estrutura de dados para realizar inferências. Uma característica-chave da representação de conhecimento por rede semântica é que as associações importantes e os fatos relevantes acerca de um objeto ou conceito podem ser inferidos diretamente dos nós aos quais ele está ligado, sem que haja necessidade de uma grande busca. Esta característica é particularmente interessante - até mesmo do ponto de vista de economia de memória - no caso de ligações que estabelecem *herança de propriedades* na rede.

SISTEMAS DE PRODUÇÃO. Os sistemas de produção já foram utilizados como modelos de cognição humana (Newell and Simon, 1972) e têm sido muito empregados em programas de inteligência artificial, notadamente nos sistemas especialistas. A idéia básica dos sistemas de produção (Davis and King, 1977) como técnica de representação é constituir a base de

conhecimentos como um conjunto de *regras de produção* na forma de pares condição-ação, interpretadas como "se a condição ocorre, então execute a ação". A utilidade deste formalismo vem do fato de que as condições de aplicabilidade de cada regra estão estabelecidas de forma explícita e também porque as interações entre regras são mínimas, uma vez que uma regra não referencia (diretamente) uma outra regra. Estas características facilitam o entendimento e a modificação de sistemas com grandes bases de conhecimento, além de permitirem facilmente o desenvolvimento de capacidade de explanação, o que tem feito dos sistemas de produção a técnica mais comumente empregada na construção de sistemas especialistas.

"FRAMES". Um "frame" (Minsky, 1975) é, basicamente, uma estrutura de dados que inclui tanto informação declarativa como procedimental ligadas por relações predefinidas, sendo útil para representar conhecimento sobre conceitos ou situações padronizadas. Quando um conceito ou situação é reconhecida, os campos ("slots") do "frame" podem ser preenchidos com valores associados a esta situação particular, ou podem "herdar" valores preestabelecidos. Uma característica interessante deste esquema é que dentre as várias informações que podem ser representadas, pode existir informação sobre como usar o próprio "frame". Outras informações que podem estar presentes são, por exemplo, o que se pode esperar em seguida dado que um evento ocorreu; o que fazer se uma expectativa não for satisfeita; como proceder para preencher um campo.

Existem outras técnicas de representação de conhecimento, evidentemente. Uma delas é a *representação direta*. No domínio de visão computacional, por exemplo, uma cena pode ser representada por um conjunto de pequenos elementos ("pixels"), os quais representam uma característica (o nível médio de cinza, por exemplo) da área correspondente na cena. Esta representação direta é útil, em geral, para tarefas de baixo-nível como encontrar contornos de objetos de cena. Para tarefas de nível mais alto (como, por exemplo, identificar os objetos da cena) ela é inadequada (Ballard and Brown, 1982). Outras técnicas são:

- utilização de outros tipos de lógica, como lógica nebulosa - "fuzzy logic" (Zadeh, 1983) ou lógica não monotônica (Doyle, 1979);

- utilização de linguagens, como PROLOG (Dahl, 1983) - baseada em lógica, KRL (Bobrow and Winograd, 1977) - baseada em "frame", OPS-5 (Browston et alii, 1985) - baseada em sistemas de produção, ou ORIENT84/K (Tokoro and Ishikawa, 1984) - baseada em objetos.

Como não é possível estabelecer qual "o melhor" esquema de representação, é preciso estar bem familiarizado com as vantagens e desvantagens de cada um, para escolher o esquema mais apropriado ao sistema que se quer desenvolver. O quadro da Tabela 2.1 destaca esses pontos para os principais esquemas de representação.

TABELA 2.1

CARACTERÍSTICAS DE ESQUEMAS DE REPRESENTAÇÃO

ESQUEMA	VANTAGENS (+) E DESVANTAGENS (-)
LÓGICA	+ disponibilidade de regras de inferência em termos das quais se pode definir procedimentos de prova que podem ser usados para resolução de problemas; + semântica formal clara, bem compreendida e aceita; + simplicidade de notação, o que torna as bases de conhecimento mais inteligíveis;

(continua)

Tabela 2.1 - Continuação.

	- falta de princípios organizacionais para os elementos da base de conhecimentos; - dificuldade em representar conhecimento procedimental e heurístico.
REPRESENTAÇÕES PROCEDIMENTAIS	+ especificação de interações diretas entre elementos, eliminando a necessidade de busca; - dificuldade em entender e modificar a base de conhecimentos.
REDES SEMÂNTICAS	+ facilidade de recuperação de informação (utilização de associações para definir caminhos de acesso); - representação gráfica que aumenta sua legibilidade; + uso de associações primitivas para organizar a base de conhecimento; + economia de representação devido à hierarquia de generalização; - falta de uma semântica formal e uma terminologia padrão.
SISTEMAS DE PRODUÇÃO	+ facilidade em aumentar e modificar a base de conhecimentos; + facilidade em desenvolver capacidade explanatória;

(continua)

Tabela 2.1 - Conclusão.

	+ uniformidade para representar meta-conhecimento; - dificuldade em expressar o conhecimento devido ao formato muito rígido.
QUADROS	+ facilidade em representar objetos (e seus relacionamentos) que aparecem em situações padronizadas; + facilidade em associar informação procedimental; + economia de representação devido à hierarquia de generalização; - nem sempre é fácil instanciar um "frame" para uma situação particular.

Características desejáveis para as bases de conhecimento poderão também favorecer um ou outro esquema, ou tipo de esquema de representação. Por exemplo:

USO MÚLTIPLO DE FATOS. Os elementos numa base de conhecimento poderão ser usados, por exemplo, para inferência ou para acesso de informação. Esquemas lógicos têm claramente vantagens sobre os outros tipos de esquemas quando se considera a base de conhecimento do ponto de vista de facilidades de inferência. A característica associativa dos esquemas de rede semântica os torna fortes candidatos aos usos relacionados com o acesso.

META-CONHECIMENTO. Um elemento que descreve a forma de outros elementos da base é um tipo importante de meta-conhecimento. Dispor de tais elementos para auxiliar, por exemplo, o processo de *aquisição de conhecimento* dirige a escolha para os esquemas declarativos em detrimento de um esquema procedimental. Um bom exemplo deste uso de meta-conhecimento encontra-se no programa TEIRESIAS (Davis, 1982).

Uma outra questão importante para o engenheiro de conhecimento é se o conhecimento deve ser representado como factual ou como inferencial. Por exemplo, pode-se ter num sistema o conhecimento factual representado por triplas (objeto, atributo, valor) e o conhecimento inferencial representado por regras de produção.

A distinção entre factual e inferencial, no entanto, nem sempre é clara. Por exemplo, deve-se representar "o ferro é um metal" como um conhecimento factual (um tripla) ou inferencial (uma regra)? Armazenado como uma tripla (ferro, natureza, metal), este fato poderá permitir a avaliação da premissa de regras. Armazenado como uma regra "SE x é um objeto de ferro, ENTÃO x é um objeto de metal", permitirá, por exemplo, trocar um objetivo "prove que x é de metal" por um outro objetivo (talvez mais adequado) "prove que x é de ferro".

Portanto, a natureza do conhecimento está ligada à maneira pela qual ele será utilizado e, em conseqüência, à maneira escolhida para representá-lo. Somente a aplicação poderá orientar a escolha.

2.2 - RESOLUÇÃO DE PROBLEMAS

Já se comentou anteriormente que as aplicações de sistemas especialistas são de tal natureza que, tipicamente, não é conhecida uma seqüência precisa de passos necessários para resolver o problema. Conseqüentemente, é preciso, a todo instante, procurar num espaço de muitos caminhos alternativos, aqueles que levam a uma solução adequada. Por outro lado, a exploração do espaço de busca (em geral, su

jeito à explosão combinatória) deve ser feita eficientemente, isto é, com um volume razoável de recursos (tempo e memória).

O tipo de programa adequado para tratar tais situações é o de uma coleção de *módulos dirigidos por padrões* (Hayes-Roth and Waterman, 1977; Waterman and Hayes-Roth, 1978). Estes módulos não são invocados por outros módulos como é comum na programação tradicional. Ao invés disto, são ativados por uma estrutura de controle sempre que determinadas condições são satisfeitas pelo estado de uma base de dados.

Um módulo dirigido por padrões (conhecido também como *fonte de conhecimento*) pode resumir-se a uma única regra de produção, ou ter centenas de linhas de código (ver sistemas MYCIN e HEARSAY-II no Capítulo 3).

Existem várias razões para implementar o procedimento de resolução de problemas de um sistema especialista na forma de módulos dirigidos por padrões. Uma razão importante é que cada um destes módulos pode ser considerado como uma unidade separada e com significado independente no domínio em questão. Isto ajuda o desenvolvimento do procedimento heurístico de resolução de problema porque a revisão de um módulo não afeta os demais.

Um sistema especialista, na tentativa de resolver um problema, deve executar uma série de ações. Cada uma dessas ações aplica alguma fonte de conhecimento (ou seja, é disparada por dados ou por elementos de solução gerados previamente e já armazenados na base de dados) e, em geral, efetua uma *transição*, isto é, altera o estado de sua base de dados, gerando um novo elemento de solução (que representa, em alguma medida, uma solução parcial do problema) ou modificando algum elemento já existente.

A cada passo desse processo, muitas dessas ações são possíveis e o sistema deve decidir qual delas executar. Para isto (ideal

mente) o sistema deve determinar o seu próprio comportamento, ou seja, deve decidir que subproblemas resolver, que conhecimento usar, que métodos de resolução e estratégias aplicar, como avaliar soluções alternativas, como saber quando os subproblemas específicos estão resolvidos e sob que circunstâncias interromper sua atenção para selecionar um novo subproblema.

Um motor de inferência (ou procedimento de inferência) que necessita a cada passo decidir apenas qual ação executar e sobre qual objeto, segundo tipologia desenvolvida por Laurent (1984), é um motor do tipo T (transição), uma vez que não existe controle sobre o estado da base de dados. É o caso dos motores de inferência da maioria dos sistemas especialistas já desenvolvidos. Quando existe memória suficiente para armazenar o espaço de busca e as transições podem ser revertidas, a escolha do estado pode ser levada em conta. Neste caso tem-se um motor do tipo E-T (estado-transição). A linguagem PROLOG com seu mecanismo de "backtracking" implementa um motor deste tipo.

Para o caso dos motores do tipo T, é possível ainda outras classificações. Por exemplo, se para uma transição a escolha do *objeto* (elemento da base de dados sobre o qual a transição é possível) puder ser feita independentemente da *ação*, tem-se um motor do tipo O-A. No sistema TEIRESIAS, por exemplo, o objeto é escolhido previamente e a escolha da ação é feita pelo emprego de meta-regras.

De uma forma geral pode-se dizer que a decisão de utilizar em um sistema especialista um ou outro tipo de motor de inferência está intimamente associada à natureza dos elementos de suas bases de conhecimentos e de dados e às suas representações. Em qualquer caso existe ainda um segundo nível de decisão: a escolha da estratégia de controle adequada para a execução de problemas do domínio em pauta.

Como já se disse anteriormente, os problemas atacados pelos sistemas especialistas exigem, para sua resolução, exploração de

um grande espaço de busca. Segundo Gevarter (1983), as técnicas de controle usadas pelos sistemas especialistas são implementações de duas idéias básicas para evitar a explosão combinatória:

- encontrar maneiras de explorar eficientemente o espaço de soluções;

- encontrar maneiras para transformar um grande espaço de soluções em espaços menores que possam ser explorados eficientemente.

Ainda segundo Gevarter, pode-se pensar que a estrutura de controle dos sistemas especialistas é constituída de 3 componentes básicos: direção de busca, estratégia de controle e transformação do espaço de busca.

DIREÇÃO DE BUSCA.

O espaço de soluções pode ser explorado de uma forma progressiva ("forward"), partindo do estado inicial e efetuando transições até encontrar um estado final (ou objetivo). A todo momento o conjunto de transições possíveis é determinado pela base de dados do sistema, razão pela qual a busca progressiva denomina-se também *busca dirigida por dados* ("data-driven search").

Outra forma de exploração é a busca regressiva ("backward"). Neste caso, parte-se de um objetivo e tenta-se buscar evidências que o validem. Dado um objetivo O, tenta-se encontrar todos os estados que possam levar a O através de transições diretas e estabelecer, recursivamente, estes estados como subobjetivos. Por isto a busca regressiva é conhecida também como *busca dirigida por objetivos* ("goal-driven search").

A direção mais adequada depende da natureza do problema e muitos sistemas especialistas já foram desenvolvidos utilizando uma

ou outra direção. O sistema R1 (McDermott, 1982), por exemplo, utiliza busca progressiva. Já o sistema MYCIN (Shortliffe, 1976) utiliza busca regressiva.

No entanto, qualquer que seja a direção escolhida, um problema perdura: como tratar o não-determinismo comumente encontrado no processo de busca (em geral, a cada passo muitas transições são possíveis). Em sistemas de produção esta questão é conhecida como *resolução de conflito*. Algumas soluções já foram propostas para este problema como "backtracking" ou busca em largura (Nilsson, 1980). Tanto R1 como MYCIN "resolvem" este problema pelo emprego exaustivo de todas as transições possíveis.

Existem outros enfoques para a direção de busca, por exemplo:

BUSCA BIDIRECIONAL (Shapiro et alii, 1982) que combina os processos progressivo e regressivo de forma a limitar o espaço de busca. O sistema MELD (Thompson and Wojcik, 1984), por exemplo, utiliza busca bidirecional da seguinte maneira: o usuário fornece as informações iniciais (em geral poucas); o sistema começa a operar no modo progressivo e propaga as informações disponíveis até não ser mais possível avançar sem informação adicional; nesse ponto o sistema muda para o modo regressivo e estabelece subobjetivos para encontrar a informação necessária; após adquirir tal informação, o sistema retorna ao modo progressivo no ponto em que havia parado e assim sucessivamente. Este tipo de enfoque pode reduzir bastante o espaço de busca, pois somente elementos de solução derivãveis das informações iniciais do usuário serão consideradas.

BUSCA DIRIGIDA POR EVENTOS. ("event-driven search"). Este tipo de busca é usado, por exemplo, no sistema CRYSALIS (Engelmore and Terry, 1979), desenvolvido para inferir a estrutura de proteínas a partir de dados de cristalografia de raio-X. Este processo utiliza muitas fontes de conhecimento que devem contribuir de uma forma cooperativa para

a solução do problema. A decisão de ativar uma fonte de conhecimento não é preestabelecida, ela depende do estado de solução, isto é, das hipóteses (elementos de solução parcial) ativas da base de dados e de itens de uma *lista de eventos* (tipos de alterações que já foram feitas nas hipóteses). Basicamente, escolhe-se um evento desta lista; o tipo do evento escolhido determina qual fonte de conhecimento ativar.

Algumas arquiteturas já foram propostas para permitir a combinação desses e de outros enfoques para a direção de busca. Por exemplo, a arquitetura do sistema LES (Laffey et alii, 1984) permite a utilização de busca dirigida por objetivos, busca dirigida por dados e busca dirigida por eventos; a do sistema KAS permite tanto busca progressiva como regressiva; e a do sistema ROSIE permite busca dirigida por estado, por objetivo e por evento. A combinação de vários enfoques é especialmente útil em ferramentas de construção de sistemas especialistas (como é o caso dos 3 sistemas citados acima) porque, com isto, será mais fácil desenvolver sistemas especialistas para muitos domínio. O sistema LES, por exemplo, foi projetado para auxiliar a construção de sistemas que resolvem problemas de diagnóstico, monitoramento, planejamento, projeto e interpretação. Os sistemas KAS e ROSIE, e algumas outras ferramentas, são descritos e comparados por Waterman e Hayes-Roth (1983).

Uma arquitetura particularmente interessante é a do sistema HEARSAY-II (Erman et alii, 1980). Esta arquitetura, também utilizada no sistema CRYSALIS, permite que as fontes de conhecimento sejam dirigidas tanto por dados como por objetivos num processo de resolução de problema do tipo oportunístico (Hayes-Roth et alii, 1979). Esta arquitetura será abordada com mais detalhes no Capítulo 3.

ESTRATÉGIA DE CONTROLE

Uma estratégia de controle para resolução de problema é a *busca exaustiva*, já citada anteriormente. Esta técnica entretanto,

só é factível de ser usada para problemas de complexidade limitada, caracterizados por espaços de busca pequenos. Problemas intrinsicamente difíceis requerem estratégias de controle mais sofisticadas. Algumas destas estratégias são abordadas a seguir.

GERAÇÃO E TESTE. Estratégia de controle utilizada no sistema DENDRAL (Buchanan and Feigenbaum, 1978), desenvolvido para inferir a estrutura molecular de compostos orgânicos a partir de dados obtidos de espectrogramas de massa. O sistema usa um algoritmo que enumera sistematicamente estruturas moleculares que satisfazem certas restrições *(fase de geração)* e, para cada estrutura obtida, faz previsões a respeito de seu espectrograma de massa, comparando-as com os dados disponíveis *(fase de teste)*. Estas duas fases se sucedem até que se consiga uma estrutura aceitável. Existe ainda uma outra fase, na qual é utilizado um grande volume de conhecimento especializado para criar listas de subestruturas recomendadas e contra-indicadas *(fase de planejamento)* a serem usadas na fase de geração. Devido a isto, a estratégia que o sistema DENDRAL emprega é melhor entendida como *planejamento, geração e teste.* Ela é utilizada também pelo sistema AM (Lenat, 1982), desenvolvido para formação de conceitos matemáticos.

CONJECTURA. Esta estratégia (também denominada *raciocínio plausível)* consiste basicamente em fazer certas conjecturas sobre estados no espaço de busca (elementos de solução plausíveis), propagar as conseqüências dessas conjecturas e verificar se o resultado obtido é aceitável ou contraditório (caso em que a conjectura deve ser substituída por outra). A dificuldade no emprego desta estratégia reside em identificar conjecturas erradas e recuperar as suas conseqüências eficientemente. Esta estratégia de controle é utilizada, por exemplo, pelo sistema EL (Stallman and Sussman, 1977) para análise de circuitos de resistores, diodos e transistores. Dada a descrição de um circuito, o sistema analisa-o e determina os valores de parâmetros como voltagem e corrente elétrica em pontos estabelecidos. Durante a análise, o sistema EL faz conjecturas sobre o estado dos diodos (dentre os 2 estados possíveis:

"LIGADO" e "DESLIGADO") e sobre o estado dos transistores (dentre os 3 estados possíveis: "EM CORTE", "ATIVO", e "SATURADO"). Considerando um estado particular para cada componente, o sistema pode empregar expressões lineares fáceis de tratar, o que torna a análise mais simples. Uma vez feitas as conjecturas, o sistema verifica se os estados são consistentes com as voltagens e correntes previstas para os dispositivos, o que pode obrigar a refazer algumas suposições. Esta estratégia de controle requer, para que as suposições sejam refeitas, algumas características arquiteturas, as quais são discutidas no Capítulo 4.

COMPROMISSO MÍNIMO. Uma estratégia bastante empregada em resolução de problemas é admitir que os problemas podem ser decompostos de forma que, resolvendo subproblemas separadamente e combinando as subsoluções encontradas, obtém-se a solução para o problema original (o espaço de busca para problemas desse tipo é chamado *fatorável* porque pode ser dividido em subespaços menores e independentes). Entretanto, esta suposição nem sempre está correta, uma vez que os subproblemas podem eventualmente interagir entre si. Neste caso o que se obtém é uma "solução aproximada". Pode-se imaginar, por exemplo, num sistema baseado em regras, que a impropriedade da solução deve-se à não-utilização de uma regra porque, no instante em que ela foi invocada, suas condições não foram satisfeitas pelo estado da base de dados. Se esta regra fosse invocada posteriormente (após alguma outra regra ter modificado o estado da base de dados), poder-se-ia obter a solução correta. A idéia de estratégia de compromisso mínimo é adiar o máximo possível decisões desse tipo, o que requer uma série de habilidades, tais como:

- saber quando existe informação suficiente para tomar uma decisão;

- suspender a atividade num subproblema quando não se dispõe de informações, retomando-a quando a informação tornar-se disponível;

- combinar informações de vários subproblemas.

Este tipo de estratégia foi utilizado, por exemplo, no sistema NOAH (Sacerdoti, 1977), de planejamento de ações de um robô.

VÁRIAS LINHAS DE RACIOCÍNIO. No caso de problemas complexos, aos quais correspondem espaços de busca muito grandes, a utilização de uma única linha de raciocínio (como a utilizada pelo sistema MYCIN, por exemplo) pode não ser eficaz porque, contida pela limitação de recursos computacionais, a busca não pode ser completa. No caso de o espaço de busca ser *fatorável*, o emprego de várias linhas de raciocínio parece ser uma solução natural para aumentar a cobertura de uma busca incompleta. Alguns sistemas especialistas já foram desenvolvidos empregando a estratégia de várias linhas de raciocínio. Por exemplo: o sistema HEARSAY-II. Sua arquitetura a emprega associando a cada fonte de conhecimento um conjunto de *"demons"* que especificam sua condição de ativação. Neste sistema, um único evento (como o armazenamento de uma nova hipótese na base de dados) pode produzir vários *registros de ativação* de fontes de conhecimento, correspondentes a várias linhas de raciocínio. Outro uso dessa estratégia encontra-se no sistema SYN, desenvolvido para síntese de circuitos elétricos: dada a forma do circuito e algumas restrições quanto ao seu comportamento, o sistema determina os valores para seus componentes (a resistência de um resistor, por exemplo). Como o sistema EL, o SYN faz uso de propagação de conjecturas, mas "enxerga" o circuito segundo vários pontos de vista, cada um deles correspondendo a uma linha de raciocínio, (por exemplo, dois resistores em série podem ser vistos como um único resistor). Com essas "visões" (que correspondem a representações equivalentes do circuito) existirão vários caminhos redundantes para a informação, o que facilita a propagação de restrições e a síntese do circuito.

TRANSFORMAÇÃO DO ESPAÇO DE BUSCA

Uma transformação muito comum, comentada anteriormente, é a *redução* (ou *decomposição*) do problema em subproblemas. As decomposi

ções de um problema podem ser representadas num grafo E/OU. Neste caso, a solução é um subgrafo do grafo E/OU. Muitos sistemas especialistas utilizam esta transformação do espaço de busca. MYCIN é o exemplo clássico.

Alguns problemas, no entanto, não podem ser completamente decompostos. Em casos assim, é preciso dispor de métodos que permitam tratar cada subproblema separadamente, armazenar as interações potenciais entre os subproblemas e combiná-los apropriadamente. Problemas deste tipo são típicos em tarefas de planejamento, existindo alguns métodos para isto. Um deles, abordado anteriormente, é a estratégia de compromisso mínimo, utilizado no sistema NOAH. Outro é o método dos *espaços de abstração*, utilizado no sistema ABSTRIPS (Sacerdoti, 1977) de planejamento hierárquico. O sistema ABSTRIPS resolve o problema considerando inicialmente somente as condições de maior *valor de criticalidade*. Esta primeira solução corresponde ao nível mais alto de abstração, ou seja, é apenas um delineamento da solução do problema. É preciso então enriquecê-la com detalhes, o que é feito considerando sucessivamente as condições de nível mais baixo de criticalidade. Este tipo de transformação do espaço de busca é conhecido como *refinamento hierárquico*.

O objetivo central de uma transformação é *reduzir* o espaço de busca. Com o refinamento hierárquico, por exemplo, não se despendiça esforço em detalhar uma "solução parcial" que, na realidade, não é uma solução adequada para o problema. Uma outra forma de restringir a busca é pela aplicação de *conhecimento*, e o enfoque predominante é a utilização de *meta-regras*. O emprego de meta-conhecimento tem grande influência na arquitetura de um sistema especialista porque remove a informação de controle do mecanismo procedimental de inferência. Vários sistemas utilizam meta-regras, por exemplo, MYCIN, MOLGEN, CRYSALIS e MELD. Meta-conhecimento pode ser utilizado também para outros propósitos. Citou-se anteriormente, por exemplo, o sistema TEIRESIAS que o emprega para gerar expectativas que auxiliem o processo de aquisição de conhecimento.

A Figura 2.1 - parcialmente extraída da Gevarter (1983) - resume a estrutura de controle de alguns sistemas especialistas comentados nesta seção.

		DIREÇÃO DE BUSCA				TECNICA DE CONTROLE					TRANSFORMAÇÃO DO ESPAÇO DE BUSCA		
SISTEMA	FUNÇÃO	PROGRESSIVA	REGRESSIVA	BI-DIRECIONAL	DIRIGIDA POR EVENTOS	BUSCA EXAUSTIVA	GERAÇÃO E TESTE	CONJECTURA	COMPROMISSO MÍNIMO	VÁRIAS LINHAS RACIOCÍNIO	REDUÇÃO DE PROBLEMAS	REFINAMENTO HIERÁRQUICO	META-REGRAS
MYCIN	diagnóstico		X			X							X
MELO	diagnóstico			X									X
DENDRAL	interpretação	X					X						
CRYSALIS	interpretação				X		X						X
HEARSAY-II	interpretação			X					X	X		X	
CL	análise	X						X					
AM	formação de conceitos	X					X						
R1	projeto	X				X					X		
MOLGEN	projeto			X				X	X		X	X	X
SYN	projeto	X								X			
NOAH	planejamento		X						X		X		
ABSTRIPS	planejamento		X									X	

Fig. 2.1 - Estrutura de controle de alguns sistemas especialistas.

CAPÍTULO 3

ALGUMAS ARQUITETURAS DE SISTEMAS ESPECIALISTAS

Pretende-se neste capítulo ilustrar, através de alguns exemplos, algumas arquiteturas de sistemas especialistas. Entende-se por arquitetura de um sistema:

- sua *estrutura* ou seja, os *módulos* que o compõem e as *comunicações* entre eles;

- sua *organização interna*, ou seja, como os módulos se organizam.

A Figura 3.1 mostra uma estrutura geral (segundo Hayes-Roth et alii, 1983) de um sistema especialista.

Os módulos que compõem este sistema e suas funções são:

(a) um *processador de linguagem* para comunicação em linguagem natural entre o usuário e o sistema;

(b) uma *base de conhecimento* que armazena fatos, regras e heurísticas de resolução de problemas específicos ao domínio de conhecimento;

(c) uma *base de dados* que armazena hipóteses e decisões intermediárias, as quais o sistema manipula durante o processo de resolução. A base de dados contém idealmente 3 tipos de elementos:

- *elementos de planejamento* que descrevem a estratégia global que o sistema irá empregar, incluindo planos, objetivos e contextos;

- *elementos de agenda* que correspondem a ações que esperam por execução;

- *elementos de solução* que representam hipóteses (elementos de solução parcial), decisões feitas até o momento e relações de dependência com decisões anteriores.

(d) um *interpretador* que aplica o conhecimento disponível;

(e) um *escalonador* para controlar a ordem com que o conhecimento de resolução do problema deve ser aplicado;

(f) um *justificador* que explica o comportamento do sistema para que o usuário possa se convencer da validade de suas conclusões;

(g) um *mantenedor de consistência* que se preocupa em manter sempre verdadeiras as razões que fundamentam as conclusões do sistema.

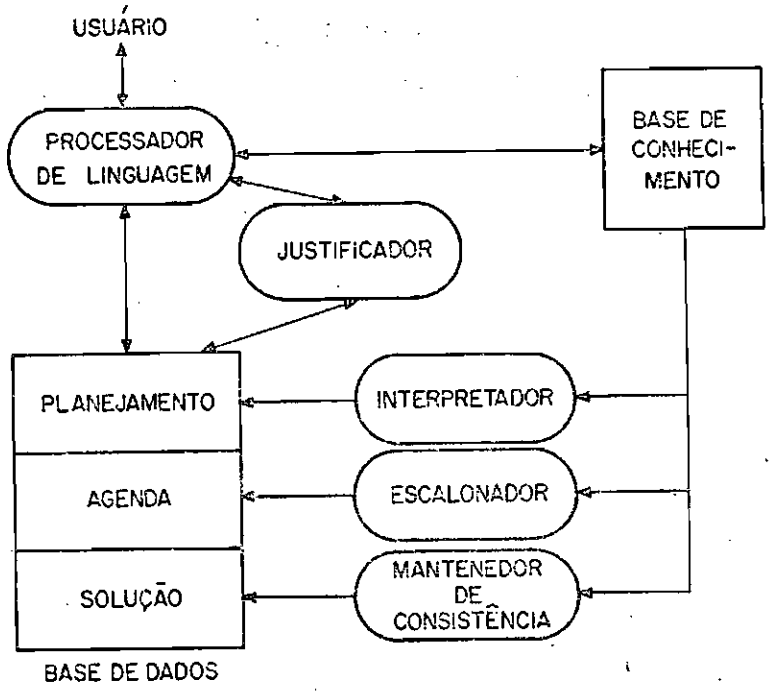

Fig. 3.1 - Uma estrutura geral de sistema especialista.

Pode-se imaginar a introdução de outros módulos nessa estrutura geral. Por exemplo, um *módulo de aprendizado* capaz de enriquecer e reestruturar os elementos da base de conhecimentos. Os sistemas especialistas já desenvolvidos implementam apenas em parte essa estrutura geral. A estrutura básica, como já se ressaltou no Capítulo 1 (Figura 1.1), isto é, a estrutura mínima comum a todos os sistemas especialistas, compreende apenas a base de conhecimento, a base de dados (armazenando, pelo menos, elementos de solução) e o interpretador.

Interessa neste capítulo abordar a *organização interna* de alguns sistemas especialistas, em especial, a organização das bases de conhecimento e de dados, e a organização do interpretador. Para isto serão abordados os sistemas MYCIN (Shortliffe, 1976), PROSPECTOR (Duda et alii, 1977) e HEARSAY-II (Erman et alii, 1980). Estes exemplos são significativos porque representam paradigmas importantes para a organização de sistemas especialistas. Todos eles deram origem a ferramentas - EMYCIN, KAS e HEARSAY-III, respectivamente - e, portanto, são uma espécie de "raiz" de várias famílias de sistemas especialistas.

3.1 - O SISTEMA MYCIN

O sistema MYCIN foi desenvolvido para auxiliar um médico no diagnóstico e na terapia de infecções bacterianas no sangue. A natureza desta tarefa apresenta certas características que influenciaram muito a escolha da arquitetura adotada para o sistema. Dentre elas:

- a decisão de quando começar o tratamento e que medicamentos usar é feita, em geral, sem dispor de conhecimento completo e preciso sobre o paciente;

- é muito importante poder aumentar e refinar o conhecimento disponível facilmente, assim como ter a capacidade de explicar de forma inteligível as decisões tomadas.

Para atender essas características, tem-se na estrutura (Figura 3.2) e na organização do sistema MYCIN:

(a) *Banco de conhecimento* - uma coleção de *regras de produção* (Davis et alii, 1977) do tipo SE <premissa> ENTÃO (x) <conclusão>, interpretadas como: se as asserções da premissa são verdadeiras, então as asserções da conclusão são verdadeiras com um grau de certeza x. As asserções (tanto da premissa como da conclusão) são compostas de cláusulas (OBJETO ATRIBUTO VALOR) e avaliadas por FUNÇÕES DE PREDICADO.

(b) *Base de dados* - armazena evidências da forma (OBJETO ATRIBUTO VALOR FATOR-DE-CERTEZA) que representam fatos deduzidos ou fornecidos ao sistema pelo usuário e utilizados para validar a premissa de regras da base de conhecimento. Por exemplo, "acredita-se com certeza de 0,78 que a identidade do organismo-1 é E.COLI" é representado por (ORGANISMO-1 IDENTIDADE E. COLI 0,78).

(c) *Módulo de aquisição de conhecimento* - interface de linguagem pseudonatural que permite aumentar ou modificar o banco de conhecimento.

(d) *Módulo de consulta* - interpretador de regras de produção.

(e) *Módulo de explanação* - permite examinar a linha de raciocínio empregada pelo sistema durante uma consulta, assim como informações relativas à base de dados e ao banco de conhecimentos do sistema. A utilização de regras de produção facilita esta tarefa: obtêm-se explicações razoáveis simplesmente exibindo as regras invocadas (pode-se obter melhores explicações incorporando um modelo do usuário - nível de conhecimento que possui e sua motivação para usar o sistema - e uma representação causal do conhecimento do domínio como propõem Willis e Shortliffe, 1982).

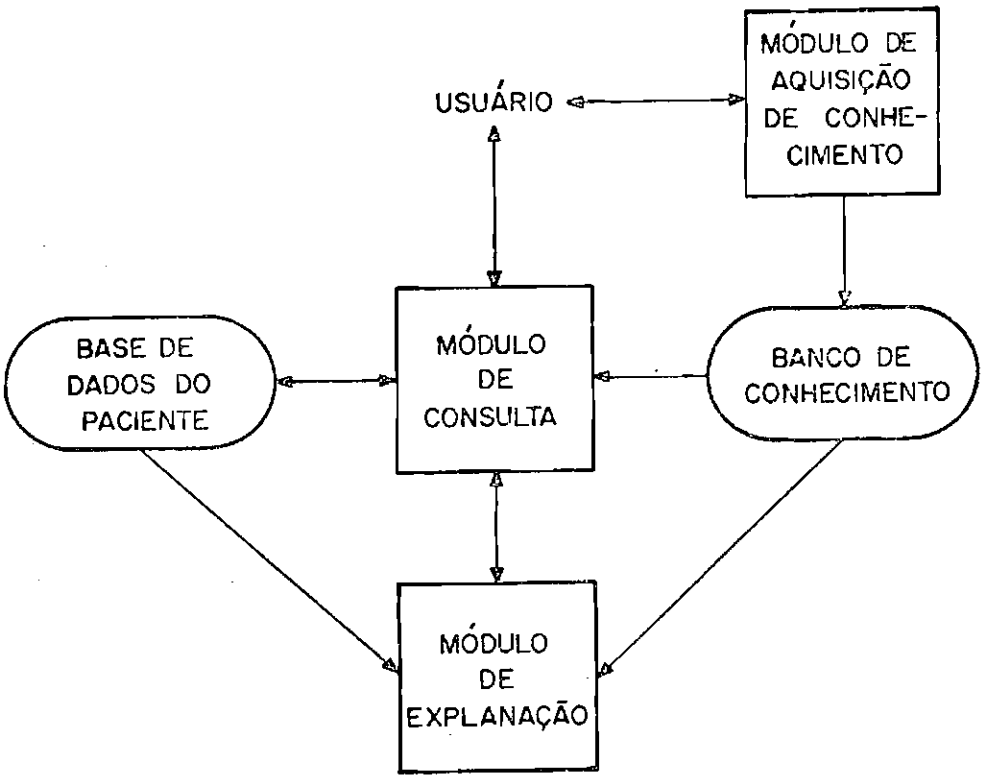

Fig. 3.2 - Estrutura do sistema MYCIN.

A estrutura de controle emprega o encadeamento regressivo de regras e incorpora métodos para combinar fatores de certeza, a fim de modelar o *raciocínio inexato*. Esta estrutura permite ainda que os objetos sejam instanciados várias vezes (pode-se, por exemplo, suspeitar de várias identidades para um dado organismo) e que as instanciações (*contextos*) sejam estruturadas numa hierarquia de *árvore*.

O encadeamento regressivo de regras produz uma busca em profundidade numa *árvore E/OU de objetivos:* dado um objetivo, recupera-se a lista de regras que o concluem; a premissa de cada regra é avaliada e as funções de predicado retornam valores entre -1 (claúsula definitivamente falsa) e 1 (cláusula definitivamente verdadeira); os valores de certeza são combinados - a certeza de uma conjunção (disjunção) de cláusulas é o mínimo (máximo) das certezas nas cláusulas individuais - e caso a premissa seja verdadeira (certeza maior que 0,2),

adota-se a conclusão com certeza dada por: <certeza na premissa> * <fator de certeza da regra>. Claúsulas de premissa que não podem ser avaliadas (não existe informação disponível para estabelecer sua veracidade ou falsidade) são estabelecidas como subobjetivos, e o processo se repete para cada um deles. Todas as regras relevantes são usadas porque no domínio do MYCIN é aconselhável considerar *exaustivamente* todas as possibilidades.

Parte da estrutura de controle pode ser implementada por *meta-regras* que estabelecem estratégias de como alcançar objetivos, ordenando ou mesmo descartando algumas regras (uma alternativa ao enfoque exaustivo).

As maiores fraquezas da arquitetura do MYCIN são:

- a representação de conhecimento que utiliza apenas regras de produção;

- a estrutura de controle restrita apenas ao encadeamento regressivo de regras; e

- a hierarquização de contextos (que asseguram as condições de aplicabilidade das regras) restrita a uma árvore, o que não é adequado para domínios nos quais os objetivos se relacionam de forma não-trivial.

3.2 - O SISTEMA PROSPECTOR

Sistema desenvolvido para auxiliar geólogos na busca de depósitos de minérios. A partir de dados de campo (características geológicas como tipos de rochas e minerais presentes ou suspeitos), o programa estima a chance de encontrar alguns tipos de depósitos de minério numa determinada região. Para isto compara os dados disponíveis com *modelos de depósitos* observando similaridades e diferenças, e, a

a partir daí, começa um diálogo para obter do usuário novas informações, a fim de determinar qual o potencial da área em estudo com relação a estes modelos.

Os modelos de depósitos de minérios do PROSPECTOR são estruturados como uma coleção de *regras de inferência* da forma:

SE : E1 & E2 & ... & En

ENTÃO : H (GS, GN)

que são interpretadas como: "as evidências E1, E2, ..., En sugerem (em algum grau) a hipótese H". Os GS e GN são, respectivamente, o *grau de suficiência* (em que medida a presença das evidências sugere a hipótese) e o *grau de necessidade* (em que medida a ausência das evidências desencoraja a hipótese).

A coleção de regras relativas a um modelo forma uma rede de inferência, como mostra a Figura 3.3, para as regras: E_1 & $E_2 \rightarrow H_2$ (GS_1, GN_1), $H_2 \rightarrow E_1$ (GS_2, GN_2) e $E_3 \rightarrow H_1$ (GS_3, GN_3).

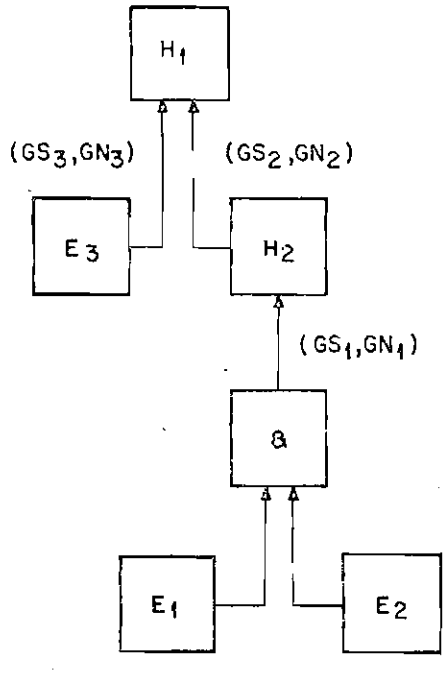

Fig. 3.3 - Rede de inferência do sistema PROSPECTOR.

Os detalhes, das regras, ou seja, o que está em cada uma dessas "caixas" - ou "espaços", segundo terminologia de Hendrix (1979) - formam uma rede semântica. Cada espaço que aparece no topo da rede de inferência (espaço-alvo) representa uma hipótese sobre a existência de um tipo de depósito mineral.

A Figura 3.4 mostra com mais detalhes uma parte de uma rede de inferência na qual existem as regras:

REGRA 1:

SE : existe pirita na forma de "filetes"

ENTÃO : H1 (GS1, GN1)

REGRA 2:

SE : existem sulfidos

ENTÃO : H2 (GS2, GN2).

A rede semântica fornece uma *taxonomia* de minerais (por exemplo, PIRITA é um SULFIDO) permitindo que a evidência observada para uma regra seja utilizada para uma outra (por exemplo, se foi observada a presença de pirita como requer a regra 1, então pode-se usar também a regra 2).

O sistema tenta avaliar a importância de uma área estimando a "confiança" que se deposita em cada um dos espaços-alvo. Para isto, as evidências geológicas fornecidas pelo usuário são armazenadas como probabilidades e associadas aos espaços da rede de inferência. Os espaços para os quais o usuário não forneceu evidência alguma assumem probabilidades "default", estabelecidas pelo geólogo ao construir o modelo.

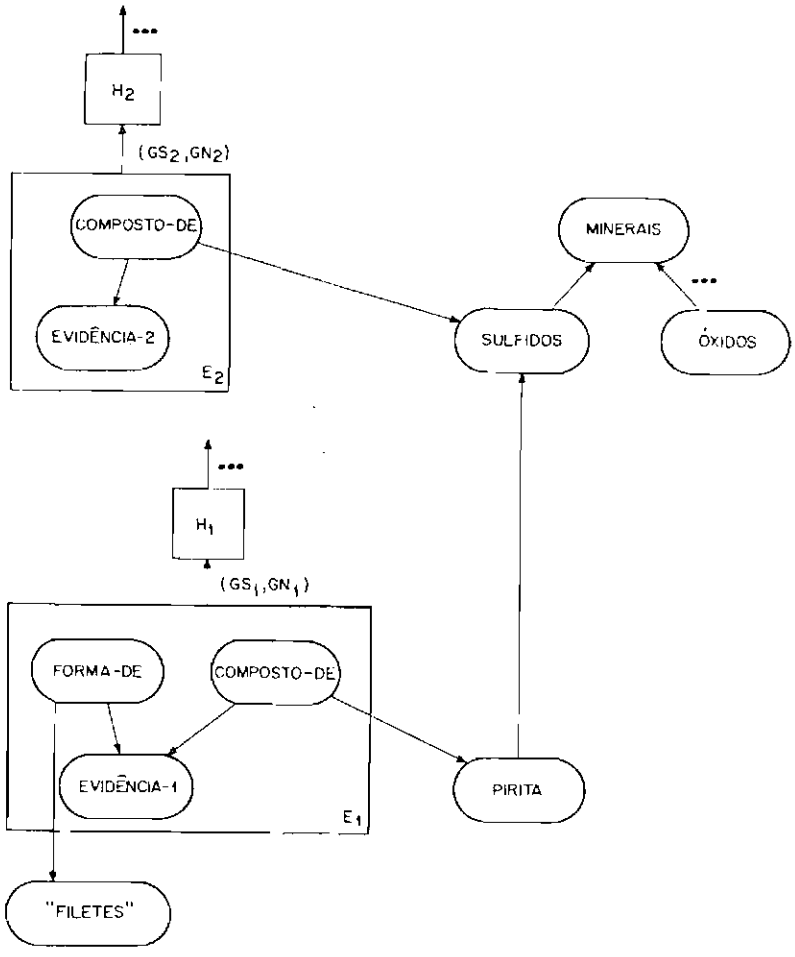

Fig. 3.4 - Rede semântica do sistema PROSPECTOR.

O sistema tenta avaliar a importância de uma área estimando a "confiança" que se deposita em cada um dos espaços-alvo. Para isto, as evidências geológicas fornecidas pelo usuário são armazenadas como probabilidades e associadas aos espaços da rede de inferência. Os espaços para os quais o usuário não forneceu evidência alguma assumem probabilidades "default", estabelecidas pelo geólogo ao construir o modelo.

A principal forma de raciocínio do PROSPECTOR é a *propagação de probabilidades* através da rede de inferência, o que é feito utilizando uma generalização do teorema de Bayes (ver NECESSIDADE DE

RACIOCÍNIO INEXATO, no Capítulo 4), desenvolvida para considerar o *grau de confiança* de a evidência estar realmente presente - que varia de -5 (definitivamente ausente) a 5 (definitivamente presente).

A estratégia de controle, responsável por decidir *que hipóteses são promissoras e que evidências buscar para confirmar hipóteses*, divide-se em 2 estágios:

(a) Com as informações disponíveis, busca-se no espaço de evidências quais são satisfeitas e propagam-se seus efeitos através da rede de inferência. Depois disto o sistema verifica as probabilidades das hipóteses-alvo e seleciona o modelo mais provável.

(b) Ordenam-se as regras correspondentes ao modelo selecionado com base em suas significâncias (uma medida que se relaciona com a probabilidade associada à hipótese, caso a evidência que a regra requer esteja presente). Seleciona-se o espaço mais significativo como novo *foco de atenção*, e o processo se repete.

Assim, no sistema PROSPECTOR as regras são usadas no *modo antecedente* (progressivamente) no primeiro estágio, para propagar as conseqüências das informações do usuário. No segundo estágio, o programa seleciona uma hipótese e caminha para trás pela rede de inferência, a fim de encontrar a evidência apropriada para perguntar ao usuário (*modo conseqüente*).

Portanto, o sistema PROSPECTOR apresenta algumas melhorias em relação ao MYCIN, como a utilização (além das regras) de redes semânticas para representar de forma estruturada o conhecimento do domínio e uma estratégia de controle que encadeia as regras de forma bidirecional. Uma fraqueza do sistema é a utilização subjetiva de probabilidades, o que pode gerar inconsistências.

3.3 - O SISTEMA HEARSAY-II

HEARSAY-II é um sistema para o entendimento de fala, cuja arquitetura (Figura 3.5) permite coordenar as contribuições de várias *fontes de conhecimento* que agem em níveis diferentes de abstração durante o processo de entendimento do sinal de voz.

Esta arquitetura possui as seguintes características:

- todos os elementos de solução gerados durante a resolução do problema são armazenados numa *base de dados* com estrutura *hierárquica* e acessível globalmente (*"blackboard"*);

- os elementos de solução são gerados e armazenados no "blackboard" pelas fontes de conhecimento, de uma forma independente e cooperativa;

- em cada ciclo de resolução um *mecanismo de escalonamento* escolhe qual fonte de conhecimento utilizar com base em uma *agenda* que contém todas as fontes utilizáveis naquele ciclo.

O "blackboard" é uma estrutura para armazenar as *hipóteses* sobre os elementos de vários níveis do sinal de fala (segmentos, sílabas, palavras, ...). Cada hipótese é basicamente: um par (ATRIBUTO VALOR), um conjunto de relacionamentos com outras hipóteses e uma *estimativa de sua validade*. As fontes de conhecimento consistem numa *condição* que estabelece sua aplicabilidade e num *programa de ação* que executa os processos necessários para produzir seus resultados, que são tipicamente: criação de hipóteses em alguns níveis do "blackboard" (processo de *previsão*), modificação de hipóteses já existentes ou junção de hipóteses de um nível a uma hipótese de um nível mais alto (processo de *reconhecimento*). Por exemplo, o ANALISADOR SINTÁTICO ("parser") é uma das fontes de conhecimento (a codificação de uma gramática) do HEARSAY-II, que toma seqüências de palavras, analisa-as e constrói fra

ses, ou seja, uma fonte que busca conhecimento no nível 5 do "blackboard" e produz hipóteses para o nível 6.

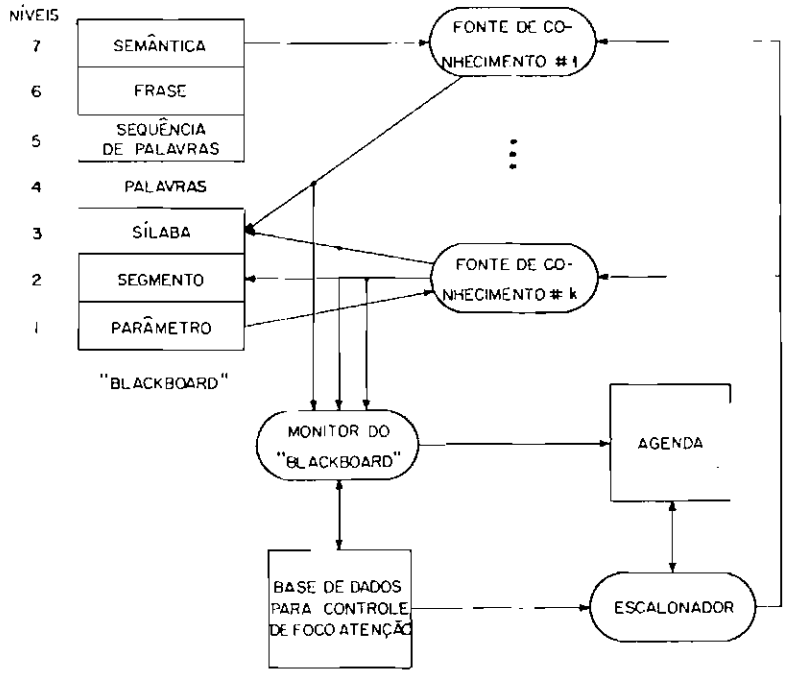

Fig. 3.5 - Estrutura do sistema HEARSAY-II.

A estratégia de controle permite que as fontes de conhecimento sejam tanto dirigidas por dados como dirigidas por objetivos, e o comportamento global do sistema pode ser visto como *oportunístico* (ou seja, guiado pelos elementos de solução mais recentes e promissores). As estimativas de validade fornecem heurísticas que indicam quais hipóteses devem ser expandidas e como isto deve ocorrer - de forma ascendente ("bottom-up") ou descendente ("top-down").

Os aspectos importantes da arquitetura do sistema HEARSAY-II são:

- o uso de quaisquer programas (incovados por padrões) como fontes de conhecimento, em contraste, por exemplo, com as regras

de produção do MYCIN, compostas por cláusulas que têm um formato bastante específico (as fontes de conhecimento do HEARSAY-II são codificadas como unidades procedimentais; uma tentativa de codificá-las na forma de regras de produção encontra-se em McCracken, 1978);

- a flexibilidade em utilizar o conhecimento, fornecida pelo mecanismo de escalonamento, em contraste com uma invocação estrita, por exemplo, dirigida por objetivo.

Para problemas complexos, que envolvem níveis de processamento muito diferentes, esses aspectos são vantajosos: pode-se empregar o enfoque mais apropriado a cada nível de processamento, ou seja, cada fonte de conhecimento pode ser um sistema especialista que atua de forma localizada.

Com efeito, além de HEARSAY-II, vários outros sistemas já foram desenvolvidos utilizando essa arquitetura para resolver problemas complicados como: interpretação de cenas (Hanson and Riseman, 1978), escalonamento de tarefas (Hayes-Roth and Hayes-Roth, 1979) ou determinação de estruturas de proteínas (Engelmore and Terry, 1979).

CAPÍTULO 4

CONCEPÇÃO DE SISTEMAS ESPECIALISTAS

Como comentado anteriormente, a organização de um sistema especialista deve refletir o tipo e a complexidade do problema a ser atacado, a disponibilidade do conhecimento necessário para resolvê-lo e a estratégia de controle que se pretende adotar para buscar a solução. Cada esquema de controle requer uma organização distinta da base de conhecimento e também um motor de inferência apropriado para aplicar o conhecimento disponível.

O Capítulo 2 mostrou que o leque de opções é grande e a escolha de princípios organizacionais nem sempre é uma tarefa fácil. Este capítulo discute, com base em Stefik et alii (1982) e Rayes-Roth (1984b), maneiras de organizar um sistema especialista para várias situações.

Segundo esses autores a função (diagnóstico; interpretação, previsão, ...) não é uma boa dimensão para classificar sistemas especialistas quando se pretende determinar padrões de estrutura e organização (pode-se apreender isto pela Figura 2.2). Segundo eles é preciso examinar os tipos de tarefas desempenhadas por especialistas para entender o que as torna difíceis, e, com base nessas dificuldades, definir como deve ser a arquitetura de um sistema especialista. A Tabela 4.1 resume suas conclusões.

TABELA 4.1

DIFICULDADES ENCONTRADAS EM TAREFAS TÍPICAS DE ESPECIALISTAS

TIPO DE TAREFA	DIFICULDADES
INTERPRETAÇÃO	- pode-se dispor apenas de informação parcial; - os dados podem ser contraditórios; - para a credibilidade da interpretação é preciso identificar onde a informação é incerta ou incompleta; - é preciso saber explicar como a interpretação foi obtida das evidências.
DIAGNÓSTICO	- sintomas de algumas falhas podem esconder outras falhas; - as falhas podem ser intermitentes; - os sintomas podem ser falsos; - nem sempre é possível dispor de todos os dados para um diagnóstico adequado; - pode-se não entender completamente a estrutura do sistema sob diagnóstico.
MONITORAMENTO	- condições de alarme dependem de contexto (as expectativas do sistema podem variar com o tempo).
PREVISÃO	- a previsão *requer* a integração de informação incompleta; - a quantidade de dados pode ser muito grande; - deve-se levar em conta vários "futuros" possíveis; - há sempre a possibilidade de eventos *imprevisíveis*.
PLANEJAMENTO	- os problemas, em geral, são grandes e complicados; - geralmente existem interações entre subobjetivos;

(Continua)

Tabela 4.1 - Conclusão.

	- deve-se poder trabalhar com incertezas; - pode-se requerer coordenação para a execução de planos.
PROJETO	- o número de possibilidades de projeto pode ser grande; - pode ser preciso compatibilizar requisitos provenientes de muitas fontes; - existem interações entre subproblemas; - é preciso justificar as decisões de projeto; - deve-se raciocinar sobre relacionamentos espaciais.

Em resumo, a dificuldade em resolver uma determinada tarefa deve-se principalmente a 4 fatores:

- pode-se não dispor de dados ou de conhecimentos precisos;

- os dados podem variar dinamicamente e, neste caso, pode ser preciso tomar decisões com base em expectativas (que podem não ser confirmadas);

- o número de possibilidades a avaliar (espaço de busca) pode ser grande;

- os procedimentos usados para descartar possibilidades podem ser muito complexos e onerosos.

A questão então é: "Quais devem ser as precrições organizacionais para superar estas dificuldades?". As respostas a esta questão serão abordadas a seguir, partindo de problemas de complexidade limitada (que permitem arquiteturas simples) e introduzindo dificuldades a cada nova situação, até chegar a problemas bastante complexos que exigem arquiteturas sofisticadas.

A ARQUITETURA MAIS SIMPLES.

Os problemas mais simples caracterizam-se por:

- pequeno espaço de busca;

- dados estáticos e confiáveis;

- conhecimento estático e confiável.

Em aplicações deste tipo, como o espaço de busca é pequeno não é preciso preocupação com limitações de recursos computacionais, sendo possível utilizar *busca exaustiva* e é suficiente desenvolver uma *única linha de raciocínio*. Além disto, devido à confiabilidade e à estabilidade tanto dos dados como do conhecimento, o *raciocínio* pode ser *monotônico*, isto é, as conclusões, uma vez inferidas, são sempre acrescentadas à base de dados do sistema. Em outras palavras, o sistema não precisa se preocupar com a veracidade dos fatos à medida que o tempo passa.

Poucos problemas reais satisfazem todos esses requisitos, sendo raro encontrar sistemas especialistas que empregam esta arquitetura.

NECESSIDADE DE RACIOCÍNIO INEXATO.

Uma pequena dificuldade pode ser introduzida, permitindo *dados e conhecimento não-confiáveis*. Esta dificuldade aparece tipicamente em tarefas de interpretação e diagnóstico (ver Figura 4.1). Muitos enfoques já foram propostos para tratar o problema de raciocínio com incerteza. Um deles é o *enfoque probabilístico* com o uso do *teorema de Bayes*. Este teorema fornece uma maneira de calcular a probabilidade de um evento dado um conjunto de observações, e foi utilizado, por exemplo, no sistema PROSPECTOR (para determinar se um certo local é um bom lugar para prospecção mineral, com base na evidência geológica disponível) na forma:

$$O(H/E) = GS * O(H),$$

$$O(H/-E) = GN * O(H),$$

onde:

O(H/E) é a "chance" de a hipótese H ser verdadeira, dado que a evidência E está presente;

O(H) é a "chance a priori" de H ser verdadeira;

GS e GN são os graus de suficiência e necessidade, respectivamente.

O grande problema deste enfoque é a determinação das probabilidades necessárias (a "chance" de uma hipótese H é a razão entre P(H) e P(-H)). Para contornar esta dificuldade, outros enfoques já foram propostos como, por exemplo, o de Zadeh (1979), que utiliza o conceito de *distribuição de possibilidades* e o de *fatores de certeza* utilizado no sistema MYCIN, no qual medidas de crença e descrença nas evidências são colhidas de forma subjetiva e combinadas para produzir a certeza numa hipótese. Este enfoque tem sido usado com razoável sucesso para modelar raciocínio inexato.

Modelos mais poderosos já foram propostos, com base na *teoria de Dempster-Shafer* (Barnett, 1981; Gordon and Shortliffe, 1985). Com esses modelos é possível atribuir certeza a conjuntos de hipóteses (ao invés de apenas hipóteses únicas como no caso do modelo de fatores de certeza), o que permite o raciocínio inexato em vários níveis de abstração. O que tem desencorajado o uso destes modelos em sistemas práticos é sua ineficiência computacional.

Prade (1982) discute a utilização de todos esses modelos em sistemas especialistas.

Enfim, modelos relacionados com probabilidades ou possibilidades usam medidas numéricas para combinar evidências no processo de raciocínio. Um enfoque radicalmente diferente é explorar a redundância dos dados para corrigir os erros que eles podem conter. Assim, utilizando *regras de correção* pode-se trabalhar com dados não-confiáveis sem comprometer a exatidão do raciocínio. Este enfoque é o empregado pelo sistema GA1 (Stefik, 1978) para a determinação de estruturas de ADN, a partir de dados de segmentação provenientes de "cortes" produzidos na molécula, por enzimas.

DADOS DINÂMICOS.

Outra dificuldade em relação à situação original aparece quando existem *dados que variam com o tempo*, o que é muito comum em tarefas de monitoramento. Uma solução é descrever o mundo como uma seqüência de instantes em cada qual existe um conjunto de fatos verdadeiros. Por exemplo, a representação por triplas (OBJETO ATRIBUTO VALOR) do sistema MYCIN pode ser facilmente ampliada para considerar também o *INSTANTE* (ou intervalo de tempo) para o qual um ATRIBUTO de um certo OBJETO assume um determinado VALOR. Este tipo de solução já foi adotado, por exemplo, para descrever um universo sujeito a ações de um robô para o qual basta que se tenha situações discretas. Problemas mais complexos, que envolvem o acompanhamento de dados ao longo do tempo, necessitam de soluções mais elaboradas. O sistema VM (Fagan et alii, 1979), por exemplo, necessita raciocinar sobre intervalos de tempo para interpretar a importância dos dados de um paciente que recebe respiração artificial numa unidade de tratamento intensivo. O sistema baseia seu raciocínio em expectativas que mantêm sobre os valores de algumas medidas fisiológicas, revendo decisões caso estas expectativas não sejam confirmadas. Ainda assim, o raciocínio do sistema VM é simples: limita-se a intervalos adjacentes de tempo. Quando se necessita raciocinar sobre eventos mais distantes, requerem-se representações mais elaboradas como, por exemplo, a proposta por Malik e Binford (1983), que converte especificações temporais em inigualdades e utiliza *programação linear* para realizar inferências.

Situações que se caracterizam por ter um pequeno espaço de busca permitem adotar soluções arquiteturais razoavelmente simples. As arquiteturas mais sofisticadas surgem quando esta situação se altera.

GRANDE ESPAÇO DE BUSCA.

Quando um problema se caracteriza por possuir um grande espaço de solução, a busca exaustiva torna-se inaplicável, a não ser que a situação a exija como em aplicações de alto risco, por exemplo, análise de substâncias perigosas ou diagnóstico médico. Em casos como estes, é preciso considerar todas as soluções possíveis, descartando somente aquelas que são inconsistentes com os dados do problema *(raciocínio por eliminação)*.

Para ganhar eficiência, é preciso descartar muitas alternativas do espaço de busca por meio de *regras de poda* que, em geral estão incorporadas num *gerador de soluções*. Com um gerador de soluções deste tipo, é possível utilizar geração e teste como estratégia de controle em grandes espaços de busca. A grande dificuldade é especificar as regras de poda, o que envolve muito conhecimento especializado. O interessante é dispor de regras que possam ser aplicadas o quanto antes no processo de geração, ou seja, regras que permitam descartar soluções ainda parcialmente especificadas *(geração e teste hierárquico* ou *raciocínio por eliminação de classes)*. Existem tarefas, no entanto, em que é difícil dispor de tais regras. Isto é típico em tarefas de projeto ou planejamento: em geral é difícil dizer se uma solução parcial (um projeto ou um plano de nível ainda muito abstrato) deve ser descartada ou não. Quando não é possível utilizar regras de poda, a solução é dividir o problema em subproblemas de forma que, pela combinação de soluções parciais, se consiga, com menos esforço, uma solução aceitável.

ORDENAÇÃO ESTÁTICA DE SUBPROBLEMAS.

No caso mais simples, existe um particionamento estático do espaço de solução adequado para todos os problemas do domínio. É o que ocorre, por exemplo, no sistema R1: a tarefa de determinar uma boa configuração para um sistema VAX é vista como uma seqüência ordenada de 6 subtarefas. O sistema resolve cada uma das subtarefas e vai combinando as soluções parciais adequadamente. Este enfoque foi utilizado também no sistema DIPMETER ADVISOR (Smith and Baker, 1983), que divide a tarefa de interpretar medidas geológicas de reservatórios de petróleo em 11 subtarefas sucessivas.

O particionamento em subproblemas corresponde à possibilidade de fazer abstrações no espaço de solução para salientar as características importantes de um problema.

ABSTRAÇÃO NO ESPAÇO DE BUSCA.

Quando não for possível determinar um conjunto fixo de subproblemas para todos os problemas do domínio, pode-se tentar uma partição "ad hoc". Um enfoque (utilizado no sistema ABSTRIPS, para planejamento) é o do *refinamento hierárquico*, em que se busca soluções em níveis cada vez mais específicos e, dentro de cada nível, os subproblemas são resolvidos numa ordem que independe do problema. No caso de os subproblemas interagirem, este enfoque pode ser combinado com a estratégia de compromisso mínimo para guiar o processo de raciocínio (o que é feito pelo sistema NOAH).

Uma dificuldade maior surge quando, para resolver um problema, é preciso *raciocinar de forma não-monotônica*, ou seja, quando é preciso fazer conjecturas durante o processo de resolução, o que, por vezes, requer que o sistema seja capaz de rever suas suposições à luz de novos conhecimentos recebidos ou descobertos, o que implica em características arquiteturais importantes.

CONJECTURAS E DEPENDÊNCIAS.

Para rever suposições que se mostraram inadequadas, um sistema deve raciocinar sobre as *dependências* que existem no seu *conjunto de crenças*. As crenças são mantidas *ativas* enquanto existem boas *justificativas* para elas. Quando surgem novas justificativas, ou quando não existir mais fundamento para alguma justificativa, é preciso rever o conjunto de crenças (este processo é conhecido como *revisão de crença* ou *manutenção da verdade*), o que pode ser feito por um mecanismo conhecido como *"backtracking" dirigido por dependências*, ou *"backtracking" não-cronológico* (Doyle, 1979). A utilização de registros de dependência tem implicações importantes na arquitetura de um sistema, até mesmo sobre sua capacidade de explicar ações e conclusões.

O emprego de raciocínio não-monotônico é importante porque para muitos problemas complexos é necessário fazer suposições sobre o problema em si, ou sobre a maneira de proceder para resolvê-lo, mas levanta alguns problemas sérios, em particular o de como organizar o sistema de forma que as dependências sejam armazenadas e utilizadas adequadamente (De Kleer et alii, 1979).

ARQUITETURAS MAIS SOFISTICADAS.

Quando a natureza do problema é muito complexa, a ponto de ser preciso combinar a contribuição de vários modelos ou a utilização de várias fontes de conhecimento, é preciso recorrer às arquiteturas mais sofisticadas como as que implementam várias linhas de raciocínio e são capazes de controlar a abrangência e a direção de busca. O sistema HEARSAY-II (ver ESTRATÉGIA DE CONTROLE, no Capítulo 2), por exemplo, foi o primeiro a empregar uma arquitetura deste tipo. Arquiteturas ainda mais sofisticadas já foram propostas como, por exemplo, com a utilização de 2 "blackboards": um para armazenar hipóteses sobre o problema e outro exclusivamente para escalonamento (Hayes-Roth, 1985). Este segundo "blackboard" permite "quebrar" o processo de escalonamento

(como é feito com o processo de resolução de problema) e utilizar um conjunto de fontes de conhecimento sobre o que é importante considerar para decidir como o sistema deve utilizar seus recursos.

CAPÍTULO 5

CONCLUSÃO

A arquitetura de um sistema - especialista ou não - consiste numa *estrutura* que identifica quais são seus componentes e de que forma eles se relacionam e numa *organização interna* que estabelece a essência de cada um de seus componentes.

Neste trabalho mostrou-se que em relação à estrutura, um sistema especialista é modular: fatos e conhecimentos de resolução de problemas sobre um domínio específico estão separados dos procedimentos de inferência que os utilizam e que, por sua vez, estão separados da base de dados que modela o "mundo" associado a um problema particular. Com relação à organização interna do banco de conhecimentos, da base de dados e do motor de inferência, mostrou-se que é preciso a combinação de várias técnicas de *representação de conhecimento* e de *resolução de problemas*.

Em resumo pode-se dizer que a arquitetura de um sistema especialista é basicamente função:

- das *características do domínio de aplicação*, tais como:

  - tamanho do espaço de busca;

  - forma dos dados disponíveis (estáticos, que variam com o tempo, incertos, inconsistentes, ...);

  - estrutura do problema a resolver (possibilidade de um particionamento estático em subproblemas, possibilidade de avaliar soluções parciais, interação entre subproblemas, ...);

- das *características do enfoque de resolução*, tais como:

  - representação de conhecimento;

  - tipo de busca (exaustiva, geração e teste, "backtracking" dirigido por dependências ...);

  - forma de controle (uma única linha de raciocínio, refinamento "top-down", escalonamento oportunístico, ...).

Em função dessas características têm-se escolhido desde arquiteturas muito simples como as dos sistemas de produção "puros", nos quais existe uma única maneira de representar conhecimento e um controle de inferência bem rígido, até arquiteturas complexas como as dos sistemas de "blackboard" com suas várias fontes de conhecimento que cooperam na resolução de problemas. Não existe ainda uma *teoria* que justifique tais escolhas. Novas arquiteturas surgem praticamente a cada nova aplicação, principalmente pela combinação de características que se mostraram relevantes em arquiteturas anteriores.

Existem atualmente, no entanto, algumas tendências:

(a) *representação de conhecimentos "profundos"* (Hart, 1982) sobre o domínio de aplicação como relações de causalidade, princípios físicos de funcionamento e estruturas para permitir analogias. O sistema MYCIN, por exemplo, não tem conhecimento sobre como as doenças aparecem ou sobre o que causa seus sintomas. A incorporação deste tipo de conhecimento já ocorre, por exemplo, no sistema CASNET (Weiss, 1978) para tratamento de glaucoma. O conhecimento "profundo", além de aumentar a capacidade de raciocínio do sistema, tem reflexos também sobre sua capacidade explanatória: o sistema pode explicar o "porquê" de suas decisões ao invés de meramente repetir associações empíricas como acontece com os sistemas "superficiais".

(b) *Incorporação de um sistema de banco de dados* convencional (como terceira memória) à estrutura básica dos sistemas. Isto já aparece, por exemplo, no sistema R1 para armazenar descrições de componentes do VAX e no sistema ACE (Vesonder et alii, 1983), um sistema especialista para manutenção de cabos telefônicos.

(c) *Extensão dos sistemas de produção* a fim de reduzir o processo de resolução de conflito e, em conseqüência, aumentar sua eficiência. Muitas modificações já foram propostas para aumentar a competência dos sistemas de produção (Rychener, 1977). Podem-se citar como novas arquiteturas:

- *sistema de produção controlado* (Georgeff, 1979) que restringe a invocação de produções pelo emprego de uma linguagem formal de controle: uma seqüência de produções (e seu efeito sobre a base de dados) é permitida somente se estiver incluída na linguagem de controle. Este tipo de sistema é útil em domínios onde o conhecimento é expresso em termos de *seqüências de ações*;

- *sistema de produção hierárquico*, comentado no item (d) a seguir;

- *sistema de produção procedimental* (Georgeff and Bonollo, 1983) que permite a representação de conhecimento procedimental por "regras", mantendo as características de modularidade, explanação e extensibilidade. O conhecimento procedimental (semelhante às fontes de conhecimento do HEARSAY-II) consiste numa *parte de invocação*, que estabelece as condições de sua aplicabilidade, e num *corpo*, um procedimento que estabelece seqüências de objetivos a alcançar e conclusões que podem ser tomadas caso sejam (ou não) atingidos estes objetivos.

Especialmente interessante é a arquitetura HAPS ("hierarquical augmentable production system architecture") que incorpora características que se imagina (Savers and Walsh, 1983) desejáveis para os futuros sistemas especialistas construídos no paradigma de sistemas de produção.

(d) *Emprego de estruturas hierárquicas* para estabelecer níveis de abstração convenientes. Podem-se citar alguns exemplos:

- Uma arquitetura de sistema de produção, proposta por Mizogushi e Kakusho (1979), que emprega duas estruturas hierárquicas de regras (uma para resolução de conflito e outra para resolução de problema) para ajudar o sistema invocar as regras apropriadas para cada caso, melhorando o seu desempenho.

- Um sistema para atualização automática de mapas florestais usando imagens Landsat, proposto por Braun e Goldberg (1985), que devido à série de tarefas (dependentes umas das outras) que precisam ser feitas sobre as imagens (registro, eliminação de regiões, detecção de objetos característicos, interpretação de alterações) compõe-se de vários subsistemas especialistas dedicados e organizados segundo uma hierarquia de árvore.

- Um sistema para interpretação de imagens multiespectrais, desenvolvido por Ferrante et alii (1984), o qual constitui-se de vários classificadores (especialistas) organizados hierarquicamente, cada um cuidando de um determinado nível de abstração: no primeiro nível o sistema aplica regras para identificar as classes mais básicas como, por exemplo, água, vegetação e construções; num nível mais alto, regiões classificadas, por exemplo, como vegetação são passadas para outro especialista que aplica regras para extrair da imagem, por exemplo, regiões de culturas ou florestas e assim sucessivamente.

(e) *Representação explícita do conhecimento de controle* para tornar mais flexível o comportamento do sistema (Thompson and Wojcik, 1984; Hayes-Roth, 1985) e aumentar sua capacidade explanatória (Clancey, 1983). A motivação para isto é o comportamento humano. Como observam Thompson e Wojcik (1984), as pessoas conseguem raciocinar sobre o próprio processo de controle. Por exemplo: são capazes de rejeitar linhas de pensamento que não parecem promissoras; são capazes de raciocinar ora de forma progressiva, ora regressivamente, dependendo da disponibilidade de dados; são capazes de utilizar o conhecimento adquirido ao resolver problemas semelhantes ou usar a intuição para controlar o raciocínio. Para um programa, raciocinar sobre o controle requer que o conhecimento de controle esteja representado explicitamente de forma que possa ser aplicado ao conhecimento de resolução de problema. Uma maneira simples de fazer isto, proposta por Davis (1977) e empregada em parte no sistema TEIRESIAS, utiliza meta-regras e divide o ciclo de controle em 2 fases:

- a primeira fase tem por objetivo decidir qual o critério de invocação utilizar (dirigido por objetivos ou dirigido por dados); faz-se isto utilizando meta-regras que especificam critérios de invocação;

- após selecionar um critério de invocação, recupera-se o conjunto apropriado de regras e utiliza-se uma segunda classe de meta-regras para selecionar e ordenar as regras mais adequadas para a resolução do problema.

Cada uma dessas fases não está limitada a apenas um nível de meta-regras: pode-se ter meta-regras que indiquem como usar meta-regras de nível mais baixo e assim sucessivamente.

Existem contudo, algumas capacidades de especialistas que ainda não são modelados pelos sistemas especialistas e que deverão dirigir a pesquisa futura, dentre elas:

- reestruturar e reorganizar o conhecimento;

- saber quando um determinado problema está fora de sua área de especialidade e quando recorrer a outros especialistas;

- tornar-se gradualmente menos proficientes nas fronteiras de sua especialidade, ao invés de subitamente incapazes.

Alguns trabalhos já vem sendo feitos com vistas nessas capacidades. Por exemplo, trabalhos sobre aprendizado estático (Cheng and Fu, 1984) e dinâmico (Politakis and Weiss, 1984); trabalhos sobre raciocínio de meta-nível (Lenat et alii, 1983) para fornecer aos sistemas um "sentimento" de suas limitações (uma forma de conhecimento de "bom senso"); e trabalhos sobre raciocínio qualificativo (Bobrow and Hayes, 1984), com representações "profundas" de conhecimento que poderão aumentar o poder de raciocínio dos sistemas especialistas.

Outras preocupações com a arquitetura de sistemas especialistas surgirão à medida que eles forem sendo aplicados para domínios maiores e mais complexos: em tais domínios será desejável a capacidade de tratar grandes bases de conhecimento e grandes memórias de trabalho. Para grandes bases, a utilização de representações uniformes e declarativas poderá comprometer de forma significativa a eficiência dos sistemas. Novos enfoques arquiteturais serão necessários como, por exemplo, de estruturas de dados especializadas ou a *compilação de conhecimento*, ou seja, a transformação de uma representação de conhecimento em outra mais eficiente. Imagina-se até mesmo que os futuros sistemas especialistas se preocupem com a *economia cognitiva*, termo proposto por Lenat, Hayes-Roth e Klahr (segundo Stefik et alii, 1982) para designar a capacidade de os sistemas melhorarem automaticamente seu

desempenho, mudando representações e mecanismos de acesso e compilando suas bases de conhecimento.

Outra possibilidade é a de sistemas especialistas que operam autonomamente em situações de tempo real, o que também exigirá o desenvolvimento de novas arquiteturas (o paradigma atual é de sistemas interativos): estes sistemas poderão ter capacidades de explanação extremamente limitadas, mas precisarão trabalhar com tempos de resposta muito curtos. Aparentemente, o domínio aeroespacial será uma área fértil ao desenvolvimento de tais sistemas, por exemplo, na automação de centros de controle ou de subsistemas de satélite. Uma experiência neste sentido é o sistema ECESIAS (Dickey and Toussaint, 1984), desenvolvido para gerenciar autonomamente o subsistema de controle ambiental de uma estação espacial tripulada.

Muita pesquisa ainda precisa ser feita para estabelecer metodologias de desenvolvimento de sistemas especialistas. O método atual é o *empírico:* constrói-se, testa-se e revisa-se. Segundo Stefik et alii (1982) a pesquisa sobre arquiteturas deverá seguir um ciclo semelhante, só que mais lento.

REFERÊNCIAS BIBLIOGRÁFICAS

BALLARD, D.H.; BROWN, C.M. *Computer vision*, Englewood Cliffs, NJ, Prentice-Hall, 1982.

BARNETT, J.A. Computational methods for a mathematical theory of evidence. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 7, Vancouver, BC, 1981. *Proceedings*. Los Altos, CA, Morgan Kaufmann, 1981, p. 868-875.

BARR, A.; FEIGENBAUM, E.A. *The handbook of artificial intelligence*, Los Altos, CA, William Kaufmann, 1981. V. 1.

BENNETT, J.S. ROGET: a knowledge-based system for acquiring the conceptual structure of a diagnostic expert system. *Journal of Automated Reasoning*, 1(1):49-74, 1985.

BOBROW, D.G.; HAYES, P.J., ed., Special volume on qualitative reasoning about physical systems. *Artificial Intelligence*, 24(1-3), Dec. 1984.

BOBROW, D.G.; RAPHAEL, B. New programming languages for artificial intelligence research. *Computing Surveys*, 6(3):153-174, Sept. 1974.

BOBROW, D.G.; WINOGRAD, T. An overview of KRL, a knowledge representation language. *Cognitive Science*, 1:3-46, 1977.

BRACHMAN, R.J.; AMAREL, S.; ENGELMAN, C.; ENGELMORE, R.S.; FEIGENBAUM, E.A.; WILKINS, D.E. What are expert systems? In: HAYES-ROTH, F.; WATERMAN, D.A.; LENAT, D.B., ed., *Building expert systems*, Reading, MA, Addison-Wesley, 1983. Cap. 2, p. 31-57.

BRACHMAN, R.J.; SMITH, B.C., ed., Special issue on knowledge representation. *SIGART Newsletter*, 70, Feb. 1980.

BRAUN, F.; GOLDBERG, M. Un système général pour la construction d'une hiérarchie d'experts en télédétection. In: CONGRÈS RECONNAISSANCE DES FORMES ET INTELLIGENCE ARTIFICIELLE, 5, Grenoble, 1985. *Proceedings*. Grenoble, INRIA, 1985. p. 931-941.

BROWNSTON, L.; FARREL, R.; KANT, E.; MARTIN, N. *Programming expert systems in OPS5 - an introduction to rule-based programming*, Reading, MA, Addison-Wesley, 1985.

BUCHANAN, B.G. Expert systems. *Journal of Automated Reasoning*, 1(1):28-35, 1985.

BUCHANAN, B.G.; BARSTOW, D.; BECHTEL, R.; BENNETT, J.; CLANCEY, W.; KULIKOWSKI, C.; MITCHELL, T.; WATERMAN, D.A. Constructing an expert system. In: HAYES-ROTH, F.; WATERMAN, D.A.; LENAT, D.B., ed., *Building expert systems*, Reading, MA, Addison-Wesley, 1983. Cap. 5, p. 127-167.

BUCHANAN, B.G.; FEIGENBAUM, E.A. DENDRAL and Meta-DENDRAL: their applications dimension. *Artificial Intelligence*, 11(1-2):5-24, Aug. 1978.

CHENG, Y.; FU, K.S. Conceptual clustering in knowledge organization. In: CONFERENCE ON ARTIFICIAL INTELLIGENCE APPLICATIONS, 1, Denver, CO, 1984. *Proceedings*. Los Angeles, CA, IEEE Computer Society, 1984. p. 274-279.

CHOURAQUI, E.; FARRENY, H.; KAYSER, D.; PRADE, H. Modélisation du raisonnement et de la connaissance. *Technique et Science Informatiques*, 4(4):391-399, 1985.

CLANCEY, W.J. The epistemology of a rule-based expert system - a framework for explanation. *Artificial Intelligence*. 20(3):215-251, May 1983.

DAHL, V. Logic programming as a representation of knowledge. *Computer*, 16(10):106-111, Oct. 1983.

DAVIS, R. Generalized procedure calling and content-directed invocation. *SIGART Newsletter*, 64:45-54, Aug. 1977.

DAVIS, R. Applications of meta level knowledge to the construction, maintenance and use of large knowledge bases. In: DAVIS, R.; LENAT, D.B., ed., *Knowledge-based systems in artificial intelligence*, New York, NY, McGraw-Hill, 1982. Cap. 2, p. 227-490.

DAVIS, R.; BUCHANAN, B.; SHORTLIFFE, E. Production rules as a representation for a knowledge-based consultation program. *Artificial Intelligence*, 8(1):15-45, Feb. 1977.

DAVIS, R.; KING, J. An overview of production systems. In: ELCOCK, E.W.; MICHIE, D., ed., *Machine Intelligence*, 8, New York, NY, Halsted Press, 1977. p. 300-332.

DE KLEER, J.; DOYLE, J.; STEELE, G.L.; SUSSMAN, G.J. Explicit control of reasoning. In: WINSTON, P.J.; BROWN, R.H., ed., *Artificial intelligence: an MIT perspective*, Cambridge, MA, MIT Press, 1979. p.93-116.

DICKEY, F.J.; TOUSSAINT, A.L. ECESIS: an application of expert systems to manned space stations. In: CONFERENCE ON ARTIFICIAL INTELLIGENCE APPLICATIONS, 1, Denver, CO, 1984. *Proceedings*. Los Angeles, CA, IEEE Computer Society, 1984. p. 483-489.

DOYLE, J. A truth maintenance system. *Artificial Intelligence*. 12(3):231-272, Nov. 1979.

DUDA, R.O.; HART, P.E.; NILSSON, N.J.; REBOH, R.; SLOCUM, J.; SUTHERLAND, G.L. *Development of a computer-based consultant for mineral exploration: annual report*. Menlo Park, CA, Stanford Research Institute, 1977.

ENGELMORE, R.; TERRY, A. Structure and function of the CRYSALIS system. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 6, Tokyo, 1979. *Proceedings*. Los Altos, CA, Morgan Kaufmann, 1979. p. 250-256.

ERMAN, L.D.; HAYES-ROTH, F.; LESSER, V.R.; REDDY, D.R. The HEARSAY-II speech-understanding system: integrating knowledge to resolve incertainly, *Computing Surveys*, 12(2):213-253, June, 1980.

FAGAN, L.M.; KUNZ, J.C.; FEIGENBAUM, E.A.; OSBORN, J.J. Representation of dynamic clinical knowledge: measurement interpretation in the intensive care unit. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 6, Tokyo, 1979. *Proceedings*. Los Altos, CA, Morgan Kaufmann, 1979. p. 260-262.

FERRANTE, R.D.; CARLOTTO, M.J.; POMAREDE, J.M.; BAIM, P.W. Multi-spectral image analysis system. In: CONFERENCE ON ARTIFICIAL INTELLIGENCE APPLICATIONS, 1, Denver, CO, 1984. *Proceedings*. Los Angeles, CA, IEEE Computer Society, 1984. p. 357-363.

FINDLER, N.V., ed. *Associative networks: representation and use of knowledge by computers*, New York, NY, Academic Press, 1979.

GEORGEFF, M.P. A framework for control in production systems. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 6, Tokyo, 1979. *Proceedings*. Los Altos, CA, Morgan Kaufmann, 1979. p. 328-334.

GEORGEFF, M.; BONOLLO, U. Procedural expert systems. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 8, Karlsruhe, 1983. *Proceedings*. Los Altos, CA, Morgan Kaufmann, 1983. p. 151-157.

GEVARTER, W.B. *An overview of artificial intelligence and robotics*, Washington, DC, NASA, 1983. (NASA Technical Memorandum 85838). V. 1.

GORDON, J.; SHORTLIFFE, E.H. A method for managing evidential reasoning in a hierarchical hypothesis space. *Artificial Intelligence*, 26(3):323-357, July, 1985.

HALLEY, P.; WILLIAMS, C. Expert system development requires knowledge engineering. *Computer design*, 25(4):83-88, Feb. 1986.

HANSON, A.R.; RISEMAN, E.M. VISIONS: a computer system for interpreting scenes. In: HANSON, A.R.; RISEMAN, E.M., ed., *Computer vision systems*, New York, NY, Academic Press, 1978. p. 303-333.

HART, P.E. Directions for AI in the eighties. *SIGART Newsletter*. 79:11-16, Jan. 1982.

HAYES-ROTH, B. A blackboard architecture for control. *Artificial Intelligence*, 26(3):251-321, July, 1985.

HAYES-ROTH, F. Knowledge-based expert systems. *Computer*, 17(10):263-273, Oct. 1984a.

HAYES-ROTH, F. The knowledge-based expert system: a tutorial. *Computer*, 17(9):11-28, Sept. 1984b.

HAYES-ROTH, B.; HAYES-ROTH, F. A cognitive model of planning. *Cognitive Science*, 3:275-310, 1979.

HAYES-ROTH, B.; HAYES-ROTH, F.; ROSENSCHEIN, S.; CAMMARATA, S. Modeling planning as an incremental opportunistic process. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 6, Tokyo, 1979. *Proceedings*. Los Altos, CA, Morgan Kaufmann, 1979. p. 375-382.

HAYES-ROTH, F.; WATERMAN, D.A., ed., Proceedings of the workshop on pattern-directed inference systems. *SIGART Newsletter*, 63, June, 1977.

HAYES-ROTH, F.; WATERMAN, D.A.; LENAT, D.B. An overview of expert systems. In: HAYES-ROTH, F.; WATERMAN, D.A.; LENAT, D.B., ed., *Building expert systems*, Reading, MA, Addison-Wesley, 1983. Cap. 1, p. 3-29.

HENDRIX, G.G. Encoding knowledge in partitioned networks. In: FINDLER, N.V., ed., *Associative networks: representation and use of knowledge by computers*, New York, NY, Academic Press, 1979. p. 51-92.

KOWALSKI, R. *Logic for problem solving*. Amsterdam, North-Holland, 1979.

LAFFEY, T.J.; PERKINS, W.A.; NGUYEN, T.A. Reasoning about fault diagnosis with LES. In: CONFERENCE ON ARTIFICIAL INTELLIGENCE APPLICATIONS, 1, Denver, CO, 1984. *Proceedings*. Los Angeles, CA, IEEE Computer Society, 1984, p. 267-273.

LAURENT, J.P. La structure de contrôle dans les systèmes experts, *Technique et Science Informatiques*, 3(3):161-177, Mai-Juin,1984.

LENAT, D.B. AM: an artificial intelligence approach to discovery in mathematics as heuristic search. In: DAVIS, R.; LENAT, D.B.;

ed., *Knowledge-based Systems in Artificial Intelligence*, New York, NY, McGraw-Hill, 1982. Cap. 1, p.1-225.

LENAT, D.; DAVIS, R.; DOYLE, J.; GENESERETH, M.; GOLDSTEIN, I.; SCHROBE, H. Reasoning about reasoning. In: HAYES-ROTH, F.; WATERMAN, D.A.; LENAT, D.B., ed., *Building expert systems*, Reading, MA, Addison Wesley, 1983. Cap. 7, p. 219-239.

MALIK, J.; BINFORD, T.O. Reasoning in time and space. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 8, Karlsruhe, 1983. *Proceedings*. Los Altos, CA, Morgan Kaufmann, 1983. p. 343-345.

McCRACKEN, D.L. *A production system version of the HEARSAY-II speech understanding system*, Ann Arbor, MI, UMI Research, 1978.

McDERMOTT, J. R1: a rule-based configurer of computer systems, *Artificial Intelligence*, 19(1):39-88, Sept. 1982.

MINSKY, M. A framework for representing knowledge. In: WINSTON, P. H., ed., *The psychology of computer vision*, New York, NY, McGraw-Hill, 1975. p. 211-277.

MIZOGUSHI, R.; KAKUSHO, D. Hierarchical production system. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 6, Tokyo, 1979. *Proceedings*. Los Altos, CA, Morgan Kaufmann, 1979. p. 586-588.

NAU, D.S. Expert computer systems. *Computer*, 16(2):63-85, Feb. 1983.

NEWELL, A.; SIMON, H.A. GPS: a program that simulates human thought. In: FEIGENBAUM, E.A.; FELDMAN, J.A., ed., *Computers and thought*, New York, NY, McGraw-Hill, 1963. p. 279-293.

NEWELL, A.; SIMON, H.A. *Human problem solving*, Englewood Cliffs, NJ, Prentice-Hall, 1972.

NII, H.P.; AIELLO, N. AGE (Attempt to GEneralize): a Knowledge-based program for building knowledge-based programs: In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 6, Tokyo, 1979. *Proceedings*. Los Altos, CA, Morgan Kaufmann, 1979. p. 645-655.

NILSSON, N.J. *Problem-solving methods in artificial intelligence.* New York, NY, McGraw-Hill, 1971.

NILSSON,N.J. *Principles of artificial intelligence*, Palo Alto, CA, Tioga, 1980.

NORMAN, D.A.; RUMELHART, D.E., ed., *Explorations in cognition*, San Francisco, CA, W.H. Freeman, 1975.

POLITAKIS, P.; WEISS, S.M. Using empirical analysis to refine expert system knowledge bases. *Artificial Intelligence*, 22(1):23-48, Jan. 1984.

PRADE, H. *Comment prendre en compte les aspects approximatifs du raisonnement humain dans les systèmes experts.* Notes de cours de l'Université Paul Sabatier, Toulouse, Oct. 1982.

REGGIA, J.A. A domain-independent system for developing knowledge bases. In: BIENNIAL CONFERENCE OF THE CANADIAN SOCIETY FOR COMPUTATIONAL STUDIES OF INTELLIGENCE, 3, Victoria, BC, 1980. *Proceedings*. Victoria, BC, University of Victoria, 1980. p. 289-295.

RICH, E. *Artificial Intelligence*, New York, NY, McGraw-Hill, 1983.

ROBINSON, A. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23-41, Jan. 1965.

RYCHENER, M.D. Control requirements for the design of production system architectures. Proceedings of the Symposium on Artificial Intelligence and Programming Languages. *SIGPLAN Notices*, 12(8):37-44, Aug. 1977.

SACERDOTTI, E.D. *A structure for plans and behavior*, New York, NY, Elsevier, 1977.

SAVERS, R.; WALSH, R. ON the requirements of future expert systems. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 8, Karlsruhe, 1983. *Proceedings*. Los Altos, CA, Morgan Kaufmann, 1983. p. 110-115.

SHAPIRO, S.; MARTINS, J.; McKAY, D. Bi-directional inference. In: ANNUAL CONFERENCE OF THE COGNITIVE SCIENCE SOCIETY, 4, Ann Arbor, MI, 1982. *Proceedings*. University of Michigan, 1982. p. 90-93.

SHORTLIFFE, E.H. *Computer-based medical consultation: MYCIN*, New York, NY, Elsevier, 1976.

SMITH, R.G.; BAKER, J.D. The dipmeter advisor system - a case study in commercial expert system development. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 8, Karlsruhe, 1983. *Proceedings*. Los Altos, CA, Morgan Kaufmann, 1983. p. 122-129.

STALLMAN, R.M.; SUSSMAN, G.J. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence*, 9(2):135-196, Oct. 1977.

STEFIK, M. Inferring DNA structures from segmentation data. *Artificial Intelligence*, 11(1-2):85-114, Aug. 1978.

STEFIK, M.; AIKINS, J.; BALZER, R.; BENOIT, J.; BIRNBAUM, L.; HAYES-ROTH, F.; SACERDOTI, E. The organization of expert systems, a tutorial. *Artificial Intelligence*, 18(2):135-173, Mar. 1982.

THOMPSON, T.F.; WOJCIK, R.M. MELD: an implementation of a meta-level architecture for process diagnosis. In: CONFERENCE ON ARTIFICIAL INTELLIGENCE APPLICATIONS, 1, Denver, CO, 1984. *Proceedings*. Los Angeles, CA, IEEE Computer Society, 1984. p. 321-330.

TOKORO, M.; ISHIKAWA, Y. An object-oriented approach to knowledge systems. In: INTERNATIONAL CONFERENCE ON FIFTH GENERATION COMPUTER SYSTEMS, Tokyo, 1984. *Proceedings*. Tokyo, ICOT, 1984. p. 623-631.

VESONDER, G.T.; STOLFO, S.J.; ZIELINSKI, J.E.; MILLER, F.D.; COPP, D. H. ACE: an expert system for telephone cable maintenance. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 8, Karlsruhe, 1983. *Proceedings*. Los Altos, CA, Morgan Kaufmann, 1983. p. 116-121.

WALLIS, J.W.; SHORTLIFFE, E.H. *Explanatory power for medical expert systems: studies in the representation of causal relationships for clinical consultations*. Stanford, CA, Stanford University, July, 1982. (report no. STAN-CS-82-923).

WATERMAN, D.A. *A guide to expert systems*, Reading, MA, Addison-Wesley, 1986.

WATERMAN, D.A.; HAYES-ROTH, F., ed., *Pattern-directed inference systems*, New York, NY, Academic Press, 1978.

WATERMAN, D.A.; HAYES-ROTH, F. An investigation of tools for building expert systems. In: HAYES-ROTH, F.; WATERMAN, D.A.; LENAT, D.B., ed., *Building expert systems*, Reading, MA, Addison-Wesley, 1983. Cap. 6, p. 169-215.

WEISS, S.M. A model-based method for computer aided medical decision-making. *Artificial Intelligence*, 11(1-2):145-172, Aug. 1978.

WINOGRAD, T. Frame representations and the procedural-declarative controversy. In: BOBROW, D.G.; COLLINS, A., ed., *Representation and understanding: studies in cognitive science*, New York, NY, Academic Press, 1975. p. 185-210.

WINSTON, P.H.; HORN, B. *LISP*. Reading, MA, Addison-Wesley, 1981.

ZADEH, L.A. A theory of approximate reasoning. In: HAYES, J.E.; MICHIE, D.; MIKULICH, L.I., ed., *Machine Intelligence*, 9, New York, NY, Halsted, 1979. Cap. 7, p. 149-194.

ZADEH, L.A. Common sense knowledge representation based on fuzzy logic. *Computer*, 16(10):61-65, Oct. 1983.