# Inteligência Computacional

Guilherme Bittencourt

**Histórico**

As correntes de pensamento que se cristalizaram em torno da IA já estavam em gestação desde os anos 30 [BF81]. No entanto, oficialmente, a IA nasceu em 1956 com uma conferência de verão em Dartmouth College, NH, USA. Na proposta dessa conferência, escrita por John McCarthy (Dartmouth), Marvin Minsky (Hardward), Nathaniel Rochester (IBM) e Claude Shannon (Bell Laboratories) e submetida à fundação Rockfeller, consta a intenção dos autores de realizar ``um estudo durante dois meses, por dez homens, sobre o tópico *inteligência artificial*". Ao que tudo indica, esta parece ser a primeira menção oficial à expressão ``Inteligência Artificial" [McC79]. Desde seus primórdios, a IA gerou polêmica, a começar pelo seu próprio nome, considerado presunçoso por alguns, até a definição de seus objetivos e metodologias. O desconhecimento dos princípios que fundamentam a inteligência, por um lado, e dos limites práticos da capacidade de processamento dos computadores, por outro, levou periodicamente a promessas exageradas e às correspondentes decepções.

Dada a impossibilidade de uma definição formal precisa para IA, visto que para tanto seria necessário definir, primeiramente, a própria inteligência, foram propostas algumas definições operacionais: ``uma máquina é inteligente se ela é capaz de solucionar uma classe de problemas que requerem inteligência para serem solucionados por seres humanos" [MH69]; ``Inteligência Artificial é a parte da ciência da computação que compreende o projeto de sistemas computacionais que exibam características associadas, quando presentes no comportamento humano, à inteligência" [BF81]; ou ainda ``Inteligência Artificial é o estudo das faculdades mentais através do uso de modelos computacionais" [CM85]. Outros se recusam a propor uma definição para o termo e preferem estabelecer os objetivos da IA: ``tornar os computadores mais úteis e compreender os princípios que tornam a inteligência possível" [Win84].

Existem duas linhas principais de pesquisa para a construção de sistemas inteligentes: a linha *conexionista* (ver seção 2) e a linha *simbólica* . A linha conexionista visa à modelagem da inteligência humana através da simulação dos componentes do cérebro, isto é, de seus neurônios, e de suas interligações. Esta proposta foi formalizada inicialmente em 1943, quando o neuropsicólogo McCulloch e o lógico Pitts propuseram um primeiro modelo matemático para um neurônio. Um primeiro modelo de *rede neuronal* , isto é, um conjunto de neurônios interligados, foi proposto por Rosenblatt. Este modelo, chamado *Perceptron* , teve suas limitações demonstradas por Minsky e Papert [MP69] em livro onde as propriedades matemáticas de redes artificiais de neurônios são analisadas. Durante um longo período essa linha de pesquisa não foi muito ativa, mas o advento dos microprocessadores, pequenos e baratos, tornou praticável a implementação de máquinas de conexão compostas de milhares de microprocessadores, o que, aliado à solução de alguns problemas teóricos importantes, deu um novo impulso às pesquisas na área. O modelo conexionista deu origem à área de redes neuronais artificiais.

A linha simbólica segue a tradição lógica e teve em McCarthy e Newell seus principais defensores. Os princípios dessa linha de pesquisa são apresentados no artigo *Physical symbol systems* de Newell [New80]. O sucesso dos *sistemas especialistas (SE)* (do inglês, ``expert system", ver capítulo 5), a partir da década de setenta, estabeleceu a manipulação simbólica de um grande número de fatos especializados sobre um domínio restrito como o paradigma corrente para a construção de sistemas inteligentes do tipo simbólico. Para facilitar a apresentação, vamos dividir a história da IA simbólica em ``épocas", conforme proposto em relatórios internos do MIT (Massachusetts Institute of Technology):

**Clássica (1956-1970)**

- Objetivo: simular a inteligência humana
- Métodos: solucionadores gerais de problemas e lógica
- Motivo do fracasso: subestimação da complexidade computacional dos problemas

**Romântica (1970-1980)**

- Objetivo: simular a inteligência humana em situações pré-determinadas.

- Métodos: formalismos de representação de conhecimento adaptados ao tipo de problema, mecanismos de ligação procedural visando maior eficiência computacional.
- Motivo do fracasso: subestimação da quantidade de conhecimento necessária para tratar mesmo o mais banal problema de senso comum.

**Moderna (1980-1990)**

- Objetivo: simular o comportamento de um especialista humano ao resolver problemas em um domínio específico.
- Métodos: Sistemas de regras, representação da incerteza, conexionismo.
- Motivo do fracasso: subestimação da complexidade do problema de aquisição de conhecimento.

**Clássica**

Inicialmente, a pesquisa em manipulação de símbolos se concentrou no desenvolvimento de formalismos gerais capazes de resolver qualquer tipo de problemas. O sistema GPS, *General Problem Solver*, projetado por Ernst e Newell [EN69], é um exemplo deste tipo de pesquisa. Estes esforços iniciais ajudaram a estabelecer os fundamentos teóricos dos sistemas de símbolos e forneceram à área da IA uma série de técnicas de programação voltadas à manipulação simbólica, por exemplo, as técnicas de busca heurística. Os sistemas gerais desenvolvidos nesta época obtiveram resultados interessantes, por vezes até impressionantes, mas apenas em domínios simplificados, onde o objetivo era principalmente a demonstração da técnica utilizada, e não a solução de um problema real. O problema com os sistemas gerais é que a sua extensão a domínios de problemas reais se mostrou inviável. Isto se deveu a duas razões, uma relacionada com características teóricas dos métodos utilizados, e outra associada à natureza do conhecimento do mundo real.

A razão teórica é conseqüência do uso, nos sistemas gerais, de modelos baseados em lógica de primeira ordem como formalismo básico. A utilização desses modelos leva à chamada *explosão combinatória* : a memória e o tempo necessários para resolver um determinado problema cresce exponencialmente com o tamanho do problema. Este problema, descrito

por Cook em seu artigo *The complexity of theorem proving procedures* [Coo71], é inerente aos métodos baseados em lógica, independentemente das técnicas de programação utilizadas. A segunda razão está associada ao fato de que, freqüentemente, o conhecimento disponível sobre o mundo real é incompleto e parcialmente incoerente, e que por vezes a única forma de solução conhecida para determinados problemas reais consiste em uma série de regras práticas não fundamentadas por nenhum tipo de teoria geral do domínio que pudesse ser usada para orientar a solução.

Esta situação levou a dois tipos diferentes de solução: (i) uso de métodos formais de inferência mais fracos do que a lógica de primeira ordem que garantissem uma certa eficiência aos programas, por exemplo, lógicas multivalores [PS85], [Bel77] e linguagens terminológicas [BW77]. (ii) Desenvolveram-se métodos heurísticos e lógicas não convencionais para permitir a representação de crenças, incoerências e incompletudes, por exemplo, lógica modal [HM85], lógica de exceções [Rei80] e lógica nebulosa [Zad79].

**Romântica**

Durante a década de setenta, a IA estava praticamente restrita ao ambiente acadêmico. Os objetivos da pesquisa eram, principalmente, a construção de teorias e o desenvolvimento de programas que verificassem estas teorias para alguns poucos exemplos. É interessante notar que o fato de que não havia interesse em construir programas de IA ``de verdade'', isto é, com aplicações práticas, não se deve a uma eventual incompetência em programação dos pesquisadores em IA. Pelo contrário, foi a inspiração desses ``hackers'' que levou a conceitos hoje integrados à ciência da computação, como: tempo compartilhado, processamento simbólico de listas, ambientes de desenvolvimento de ``software'', orientação objeto, etc., além da mudança da relação usuário-computador ao eliminar a intermediação de um operador e colocar cada usuário diante de sua estação de trabalho.

Uma mudança importante ocorreu ao longo da década de setenta em relação aos critérios acadêmicos de julgamento de trabalhos em IA: houve uma crescente exigência de formalização matemática. Se no início dos anos setenta, um programa, mesmo tratando de alguns poucos exemplos de um problema até então não tratado, já era considerado IA, isto não acontecia mais em 1980. O programa em si passou a ser a parte menos importante; a

análise formal da metodologia, incluindo decidibilidade, completude e complexidade, além de uma semântica bem fundada, passou a ser o ponto fundamental [Hay77], [McD78]. A década de setenta marcou também a passagem da IA para a ``vida adulta'': com o aparecimento dos primeiros SE's, a tecnologia de IA passou a permitir o desenvolvimento de sistemas com desempenho intelectual equivalente ao de um ser humano adulto, abrindo perspectivas de aplicações comerciais e industriais.

**Moderna**

A tecnologia de SE disseminou-se rapidamente e foi responsável por mais um dos episódios ligados a promessas não cumpridas pela IA: o sucesso dos primeiros SE's chamou a atenção dos empresários, que partiram em busca de um produto comercializável que utilizasse esta tecnologia. No entanto, um SE não era um produto: um produto, na visão dos empresários, não deveria ser um sistema específico para um dado problema, mas algo que fosse implementado uma única vez e vendido em 100.000 unidades, por exemplo, uma *ferramenta para a construção de sistemas especialistas (ASE)* . Com isso foram colocadas no mercado uma grande quantidade de ASE's que prometiam solucionar o problema de construção de SE's. A conseqüência foi uma grande insatisfação por parte dos usuários, pois, apesar de uma ferramenta de programação adequada ajudar muito a construir um sistema complexo, saber o que programar continua sendo o ponto mais importante.

Se os ASE's deveriam ser vendidos como produtos de IA, então em algum lugar deveria haver IA, e o lugar escolhido foi o *motor de inferência* (ver seção 5.3.4), que passou a ser considerado como sinônimo de IA. Isto levou à ilusão de que para construir um SE bastaria comprar um ASE, enquanto que a verdade é que a IA em um SE está basicamente na forma como é representado o conhecimento sobre o domínio, isto é, onde a IA sempre esteve: na tentativa de entender o comportamento inteligente a ser modelado, no caso o comportamento do especialista ao resolver um problema. Uma outra conseqüência desta visão distorcida dos ASE's foi a pouca ênfase dada inicialmente à *aquisição de conhecimento* , certamente a parte mais difícil do desenvolvimento de um SE. Se exageros existiram na publicidade dos ASE's, por certo houve também trabalhos descrevendo com

fidelidade o potencial e as limitações da nova tecnologia (por exemplo, [DK77], [HRWL83] e [Wat86]).

Atualmente, os ASE's são considerados como parte de uma tecnologia de desenvolvimento de ``software'' estabelecida, sendo objeto de diversas conferências internacionais e submetida a avaliações rigorosas de desempenho (por exemplo, [SMS92]). Entre os diversos benefícios associados ao desenvolvimento de SE's podem-se citar: distribuição de conhecimento especializado, memória institucional, flexibilidade no fornecimento de serviços (consultas médicas, jurídicas, técnicas, etc.), facilidade na operação de equipamentos, maior confiabilidade de operação, possibilidade de tratar situações a partir de conhecimentos incompletos ou incertos, treinamento, entre outros. Atualmente, existem milhares de SE's em operação nos mais variados domínios, e a influência da IA em outros campos da computação, como engenharia de ``software'', bancos de dados e processamento de imagens vem crescendo constantemente.

As principais áreas de pesquisa em IA simbólica são atualmente: sistemas especialistas, aprendizagem, representação de conhecimento, aquisição de conhecimento, tratamento de informação imperfeita, visão computacional, robótica, controle inteligente, inteligência artificial distribuída, modelagem cognitiva, arquiteturas para sistemas inteligentes, linguagem natural e interfaces inteligentes. Além das linhas conexionista e simbólica, observa-se hoje o crescimento de uma nova linha de pesquisa em IA, baseada na observação de mecanismos evolutivos encontrados na natureza, tais como a auto-organização e o comportamento adaptativo. Nesta linha, os modelos mais conhecidos são os algoritmos genéticos e os autômatos celulares [Bar75], [FTW83], [GH88], [Hol75] e [Hol86] (ver seção 3).

A gradativa mudança de metas da IA, desde o sonho de construir uma inteligência artificial de caráter geral comparável à do ser humano até os bem mais modestos objetivos atuais de tornar os computadores mais úteis através de ferramentas que auxiliam as atividades intelectuais de seres humanos, coloca a IA na perspectiva de uma atividade que praticamente caracteriza a espécie humana: a capacidade de utilizar representações externas, seja na forma de linguagem, seja através de outros meios [Hil89]. Deste ponto de

vista, a computação em geral e a IA em particular são o ponto culminante de um longo processo de criação de representações de conhecimento, iniciado com as primeiras pinturas rupestres. Esta nova perspectiva coloca os programas de IA como produtos intelectuais no mesmo nível dos demais, ressaltando questões cuja importância é central para os interesses atuais da IA, por exemplo, como expressar as características individuais e sociais da inteligência utilizando computadores de maneira a permitir uma maior produtividade, e como as propriedades das representações utilizadas auxiliam e moldam o desenvolvimento de produtos intelectuais?

**Conexionismo**

O *conexionismo* [1] é uma das duas grandes linhas de pesquisa da IA e tem por objetivo investigar a possibilidade de simulação de comportamentos inteligentes através de modelos baseados na estrutura e funcionamento do cérebro humano. Os primeiros trabalhos desenvolvidos na área datam de 1943, quando o neurofisiologista, filósofo e poeta americano Warren McCulloch, e o lógico Walter Pitts desenvolveram o primeiro modelo matemático de um neurônio.

As linhas conexionista e simbólica nasceram praticamente juntas: o livro publicado após o encontro em Darthmouth College, em 1956, já continha um artigo a respeito de redes neuronais. No entanto, por uma série de razões, as técnicas simbólicas de IA, baseadas na lógica, tiveram preferência na época. Entre estas razões pode-se citar a falta de computadores suficientemente potentes para tratar a complexidade inerente ao método, e a publicação do livro *Perceptrons* por Minsky e Papert [MP69], onde as propriedades matemáticas de redes artificiais de neurônios são analisadas e suas limitações são apontadas, como, por exemplo, a impossibilidade de simular o operador ``ou-exclusivo'' com Perceptrons de uma camada.

Na década de 80, houve um renascimento do interesse sobre o conexionismo. Este renascimento deve-se a diversos fatores, por exemplo, melhores conhecimentos da estrutura real do cérebro, melhores algoritmos de treinamento e disponibilidade de computadores poderosos, inclusive paralelos. Os sucessos obtidos em aplicações práticas levaram a uma mudança de ênfase quanto ao objetivo da pesquisa na área. Uma parte da pesquisa passa a

se dedicar ao estudo de redes neuronais vistas apenas como uma representação de funções matemáticas utilizando elementos computacionais aritméticos simples, sem maiores relações com a modelagem do sistema nervoso.

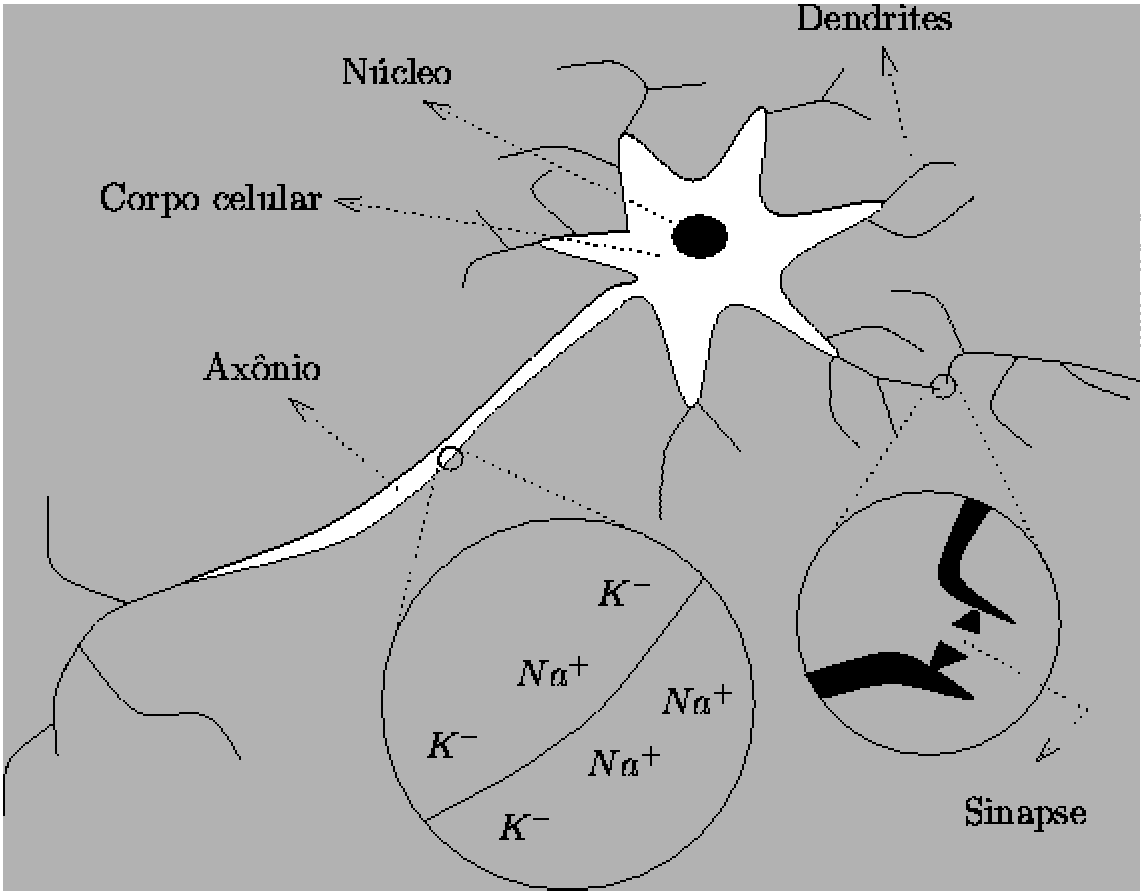
As características que tornam a metodologia de redes neuronais interessante do ponto de vista da solução de problemas são as seguintes:

- Capacidade de ``aprender" através de exemplos e de generalizar este aprendizado de maneira a reconhecer instâncias similares que nunca haviam sido apresentadas como exemplo.
- Bom desempenho em tarefas mal definidas, onde falta o conhecimento explícito sobre como encontrar uma solução.
- Não requer conhecimento a respeito de eventuais modelos matemáticos dos domínios de aplicação.
- Elevada imunidade ao ruído, isto é, o desempenho de uma rede neuronal não entra em colapso em presença de informações falsas ou ausentes, como é o caso nos programas convencionais, mas piora de maneira gradativa.
- Possibilidade de simulação de raciocínio ``a priori" e impreciso, através da associação com a lógica nebulosa (ver seção 4).

Alguns domínios onde são comuns aplicações da técnica de redes neuronais são: reconhecimento de padrões em geral (por exemplo, visão computacional, reconhecimento de voz, etc.), processamento de sinais, previsão desde variação de carga elétrica até cotações da bolsa de valores, diagnóstico de falhas e identificação e controle de processos.

**O modelo de McCulloch e Pitts**

**Figure:** Neurônio biológico




A estrutura do neurônio artificial proposto por McCulloch e Pitts é baseada no neurônio biológico. De maneira extremamente simplificada, um neurônio biológico é formado por um *corpo celular* ou *soma* que contém o *núcleo* da célula, diversos *dendrites* , através dos quais impulsos elétricos são recebidos, e um *axônio* , através do qual impulsos elétricos são enviados. A propagação de um impulso elétrico ao longo de um dendrite ou de um axônio se dá através da alteração da concentração dos íons $K^-$ e $Na^+$ em ambos os lados da membrana. A figura 1 mostra, esquematicamente, a estrutura de um neurônio. As interligações entre neurônios são efetuadas através de *sinapses* , pontos de contato entre dendrites e axônios controlados por impulsos elétricos e por reações químicas devidas a substâncias chamadas *neurotransmissores* .

A maior limitação do modelo de neurônio de McCulloch e Pitts é sua natureza binária. Esta limitação, conseqüência da crença por parte de McCulloch de que o funcionamento do

sistema nervoso central era baseado em uma representação do tipo ``tudo ou nada", também pode ser incluída como uma das razões para o pouco sucesso obtido inicialmente pelas redes neuronais. O funcionamento do modelo pode ser descrito intuitivamente da seguinte maneira: se a soma ponderada dos sinais de entrada de um neurônio ultrapassar um determinado limite de disparo, então a saída toma valor um; se não ultrapassar, toma valor zero. As entradas do neurônio também são binárias.

Formalmente, este funcionamento pode ser descrito da seguinte maneira. Considere o *i*-ésimo neurônio de uma rede neuronal com *n* neurônios. Este neurônio é caracterizado pelo valor $x_i$, chamado *atividade* do neurônio (que corresponde à taxa média de disparos dos *potenciais de ação* do neurônio biológico) e pelo valor $\sigma_i$, chamado *nível de ativação* do neurônio (que corresponde ao *potencial de membrana* do neurônio biológico). No modelo de McCulloch e Pitts, o nível de ativação é definido da seguinte maneira:

$$\sigma_i = \sum_{j=1}^{n} \omega_{ij} x_j$$

onde $\omega_{ij} \in \mathbb{R}$ é o peso atribuído àquela entrada do neurônio *i* cuja origem é a atividade do neurônio *j*. Este peso simula a sinapse entre dois neurônios e quando estes não estão conectados assume o valor zero. Assim como no caso biológico, também para as redes neuronais artificiais há dois tipos de sinapses: as excitadoras e as inibidoras. Uma sinapse excitadora tem o seu peso positivo, $\omega_{ij} > 0$, e uma sinapse inibidora tem o seu peso negativo, $\omega_{ij} < 0$. A atividade do neurônio *i* é dada por $x_i = f(\sigma_i)$. A função *f*, chamada *função de ativação* ou de *transferência* , adotada no modelo é a *função degrau* :

$$f(x) = \begin{cases} 0 & \text{se } x \leq \alpha \\ 1 & \text{se } x > \alpha \end{cases}$$

onde $\alpha$ é o limite de disparo.

Mesmo com este modelo rudimentar de neurônio, McCulloch e Pitts foram capazes de provar que uma rede neuronal é equivalente a uma máquina de Turing e, logo, capaz de calcular qualquer função computável [MP43]. Esta equivalência deve-se ao fato de que é possível simular os operadores ``ou", ``e" e ``não" utilizando redes neuronais e que, a partir destes operadores, pode-se construir um computador convencional.

**Modelo geral de neurônio**

O modelo atualmente utilizado para um neurônio artificial generaliza o modelo de McCulloch e Pitts nos seguintes aspectos:

- O nível de ativação passa a ser definido como uma função qualquer *g* das atividades dos neurônios da rede:

  $$\sigma_i = g(x_1, \ldots, x_n).$$

- A função de ativação *f*, que determina a atividade de um neurônio, é generalizada e passa a ser uma função limitada qualquer. É interessante ainda que esta função seja não linear, pois neste caso as restrições do modelo binário de McCulloch e Pitts desaparecem.
- É introduzido um valor de polarização $\theta \in \mathbb{R}$, de modo que a atividade de um neurônio passa a ser calculada por $x_i = f(\sigma_i + \theta)$.

Na maioria dos modelos, a função *g* é, da mesma maneira que no modelo de McCulloch e Pitts, simplesmente a soma ponderada, embora existam modelos onde é utilizado o produto, o mínimo ou o máximo. As funções *f* mais utilizadas, além da função degrau, são:

- *função semi-linear* :

$$f(x) = \begin{cases} 0 & \text{se } x < \alpha_{min} \\ mx + l & \text{se } \alpha_{min} \leq x \leq \alpha_{max} \\ f_{max} & \text{se } x > \alpha_{max} \end{cases}$$

- *função sigmoidal* :

$$f(x) = \frac{f_{max}}{1 + e^{-x}}$$

**Redes neuronais**

Uma vez definido um neurônio, é possível estudar as propriedades de redes de neurônios interconectados, as chamadas *redes neuronais* . Do ponto de vista do fluxo de informação, uma rede neuronal pode conter três tipos de neurônios: (i) de entrada - $u_k$ -, análogos aos neurônios sensoriais dos seres vivos, (ii) de saída - $y_k$ -, análogos aos neurônios motores e, finalmente, (iii) neurônios internos. Este último tipo tem um interesse biológico particular pelo fato de que, se admitirmos ciclos em sua estrutura de interligação, então é possível que eles desenvolvam atividades independentes de estímulos externos. Estas atividades podem ser interpretadas, metaforicamente, como ``pensamentos". Os neurônios internos são também importantes de um ponto de vista matemático, pois é possível provar que a classe de problemas conhecidos como *não linearmente separáveis* (do qual o ``ou-exclusivo" é

uma instância) só permite solução através de redes neuronais que possuam ao menos uma camada interna.

Do ponto de vista funcional, uma rede pode ser *homogênea* , se todos os neurônios se comportarem da mesma forma, ou, no caso contrário, *heterogêneas* . As redes artificiais são normalmente homogêneas, diferentemente das redes neuronais biológicas que são bastante heterogêneas.

Do ponto de vista da topologia da interligação entre neurônios, uma rede pode ser de *alimentação para frente* (do inglês, ``feed-forward") ou *recorrentes* . Nas redes de alimentação para frente, os neurônios são organizados em camadas e a informação se desloca em um único sentido, entre camadas adjacentes. Nas redes recorrentes, não existe direção privilegiada para a propagação da informação, podendo haver retroalimentação. Um caso limite de rede recorrente é a rede *totalmente conectada* onde cada neurônio está conectado a todos os outros e da qual toda estrutura de interligação é um caso particular.

**Table 1:** Modelos de redes neuronais

| | |
|---|---|
| Perceptrons [Ros62] [MP69] | |
| Aplicações | Reconhecimento de caracteres |
| Vantagem | Rede neuronal mais antiga |
| Desvantagens | Não reconhece padrões complexos, |
| | sensível a mudanças |
| Backpropagation [RHW86] | |
| Aplicações | Larga aplicação |

| | |
|---|---|
| Vantagens | Rede mais utilizada, |
| | simples e eficiente |
| Desvantagens | Treinamento supervisionado, |
| | exige muitos exemplos |
| Counterpropagation [HN90] | |
| Aplicações | Reconhecimento de padrões, |
| | análise estatística |
| Vantagem | Rapidez do treinamento |
| Desvantagem | Topologia complexa |
| Hopfield [Hop82] | |
| Aplicações | Recuperação de dados |
| | e fragmentos de imagens |
| Vantagem | Implementação em larga escala |
| Desvantagens | Sem aprendizado, pesos preestabelecidos |
| Bidirectional Associative Memories (BAM) [Kos92] | |
| Aplicações | Reconhecimento de padrões |
| Vantagem | Estável |
| Desvantagem | Pouco eficiente |
| Kohonen [Koh87] | |
| Aplicações | Reconhecimento de padrões |
| | não especificados |
| Vantagem | Auto-organização |

| | |
|---|---|
| Desvantagem | Pouco eficiente |

O primeiro modelo de rede neuronal - chamado *Perceptron* - foi proposto por Rosenblatt, em 1957. Este consiste em uma rede de duas camadas (uma de entrada, utilizada apenas para redistribuição da informação de entrada, e outra de saída, onde o processamento é realmente realizado) formadas por neurônios binários. Desde então diversos modelos foram propostos. Alguns dos modelos mais conhecidos, seus domínios de aplicação, vantagens e desvantagens, são apresentados no quadro 1.

Para servir de termo de comparação quanto ao tamanho de uma rede neuronal artificial, o sistema nervoso humano é constituído de cerca de $10^{11}$ neurônios, participando em aproximadamente $10^{15}$ interconexões sobre vias de transmissão que podem atingir um metro ou mais.

Independentemente da topologia de uma rede neuronal, seu comportamento ao longo do tempo pode ser formalizado matematicamente através de ferramentas formais da álgebra linear. A cada instante de tempo, o estado de uma rede neuronal, com $n$ neurônios, é representado por um vetor $n$-dimensional cujos componentes são os valores das atividades neuronais $x_i$. Um outro vetor é associado aos níveis de ativação $\sigma_i$ de cada neurônio. As interconexões entre neurônios, com seus respectivos pesos, podem ser então representadas na forma de uma matriz $n \times n$ (alguns pesos podem ter valor zero, significando ausência de conexão). Nesta representação, pode-se obter o vetor de níveis de ativação, em um determinado instante de tempo, a partir do vetor de atividades associado ao instante anterior através do seguinte produto matricial:

$$\vec{\sigma}(t+1) = [\omega_{ij}(t)]\ \vec{x}(t).$$

Uma rede neuronal é um sistema dinâmico onde o novo estado é função do estado anterior. O cálculo do novo estado da rede é chamado de *atualização* . A atualização pode ser *síncrona* , quando os novos níveis de ativação são determinados simultaneamente em função dos valores instantâneos das atividades, ou *assíncrona* , em que é atualizado o nível de ativação de uma única unidade por vez.

**Treinamento**

Treinar uma rede neuronal significa ajustar sua matriz de pesos de forma que o vetor de saída - $\vec{y}$ contido em $\vec{x}$ - coincida com um certo valor desejado para cada vetor de entrada - $\vec{u}$ também contido em $\vec{x}$. Existem também alguns algoritmos de treinamento que, além do ajuste de pesos, provocam também mudanças na própria arquitetura da rede, como a criação ou eliminação de neurônios. O treinamento pode ser de dois tipos: *supervisionado* ou *não supervisionado* . O treinamento supervisionado exige a disponibilidade de um conjunto de treinamento formado por pares de vetores de entrada e de saída, chamados *pares de treinamento* . Já no treinamento não supervisionado, o conjunto de treinamento consiste somente de vetores de entrada. Uma rede pode ser treinada com três objetivos diferentes:

- Auto-associação: após o treinamento com um conjunto de vetores, quando submetida a um vetor similar a um dos exemplos, mas deturpado, reconstituir o vetor original.
- Heteroassociação: após o treinamento com um conjunto de pares de vetores, quando submetida a um vetor similar ao primeiro elemento de um par, mas deturpado, reconstituir o segundo elemento do par.
- Detecção de regularidades: descobrir as regularidades inerentes aos vetores de treinamento e criar padrões para classificá-los de acordo com tais regularidades.

A maioria dos algoritmos de treinamento de rede neuronais é inspirada, direta ou indiretamente, na *lei de Hebb* , descoberta pelo biólogo que lhe empresta o nome:

A intensidade de uma ligação sináptica entre dois neurônios aumenta se ambos são excitados simultaneamente.

Esta lei pode ser formalizada da seguinte maneira. Seja:

$$\Delta\omega_{ij} = \eta x_i x_j$$

a variação no peso da conecção entre os neurônios *j* e *i*, onde $\eta$ é chamada *taxa de aprendizado* e $x_i$ e $x_j$ são as atividades dos neurônios. Uma propriedade importante da Lei de Hebb é sua *localidade*, isto é, a variação do peso de uma sinapse depende apenas de informações locais à sinapse.

A Lei de Hebb pode ser modificada para levar em conta o valor desejado para a atividade do neurônio - $d_i$-, dando origem à *regra delta* :

$$\Delta\omega_{ij} = \eta(d_i - x_i)x_j.$$

O algoritmo mais conhecido para treinamento de redes neuronais é a *retropropagação* (do inglês, ``backpropagation"). Este algoritmo pode ser considerado como uma generalização da regra delta para redes de alimentação para frente com mais de duas camadas. A retropropagação é um algoritmo de treinamento supervisionado. Seu funcionamento pode ser descrito da seguinte forma: (i) apresenta-se um exemplo à rede e obtém-se a saída correspondente, (ii) calcula-se o vetor de erro que consiste na diferença entre a saída obtida e a esperada, (iii) calcula-se o gradiente do vetor de erro e atualiza-se, utilizando a regra delta, os pesos da camada de saída e, finalmente, (iv) propaga-se para trás (origem do nome do algoritmo) os valores desejados de modo a atualizar os pesos das demais camadas.

Como o algoritmo de retropropagação requer o cálculo do gradiente do vetor de erro, é interessante que a função de ativação seja derivável em todos os pontos. Isto explica o sucesso da função sigmóide como função de ativação, pois ela apresenta esta propriedade.

Outros algoritmos de treinamento de redes neuronais são:

- contrapropagação (do inglês, ``counterpropagation");
- aprendizado competitivo, utilizado nas redes de Kohonen;
- algoritmos genéticos (ver seção 3).

**Computação evolutiva**

A *computação evolutiva (CE)* é um ramo da ciência da computação que propõe um paradigma alternativo ao processamento de dados convencional. Este novo paradigma, diferentemente do convencional, não exige, para resolver um problema, o conhecimento prévio de uma maneira de encontrar uma solução. A CE é baseada em mecanismos evolutivos encontrados na natureza, tais como a auto-organização e o comportamento adaptativo [FTW83], [GH88]. Estes mecanismos foram descobertos e formalizados por Darwin em sua *teoria da evolução natural* , segundo a qual, a vida na terra é o resultado de um processo de seleção, pelo meio ambiente, dos mais aptos e adaptados, e por isto mesmo com mais chances de reproduzir-se. A diversidade da vida, associada ao fato de que todos os seres vivos compartilham uma bagagem genética comum, pelo menos em termos de seus componentes básicos, é um exemplo eloqüente das possibilidades do mecanismo de evolução natural.

**Computação e computação evolutiva**

Segundo Heitkoetter [HB94], a CE pertence ao ramo da *computação natural* que inclui os tópicos de vida artificial, geometria fractal, sistemas complexos e inteligência computacional. Este último inclui redes neuronais artificiais (ver seção 2), sistemas nebulosos (do inglês ``fuzzy systems", ver seção 4) e a CE.

Historicamente, as primeiras iniciativas na área de CE foram de biólogos e geneticistas interessados em simular os processos vitais em computador, o que recebeu na época o nome de ``processos genéticos''. Alguns desses cientistas, citados em [Gol89], são: Barricelli, 1957, 1962; Fraser, 1960, 1962; Martin e Cockerham, 1960. Em sua tese de doutorado, o biólogo Rosenberg, em 1967, simulou uma população de seres unicelulares, estrutura genética clássica (um gene, uma enzima), com estrutura diplóide, com cromossomos de 20 genes e 16 alelos permitidos em cada um [Gol89].

Já na década de 60, Holland e outros começaram a estudar os chamados *sistemas adaptativos* , que foram modelados como sistemas de *aprendizagem de máquina* . Tais modelos, conhecidos como *algoritmos genéticos* , implementavam populações de indivíduos contendo um genótipo, formado por cromossomos (que neste modelo eram representados por cadeias de ``bits'') aos quais se aplicavam operadores de seleção, recombinação e mutação. Ainda que Holland tenha proposto um quarto operador (a inversão), este não chegou a ser largamente utilizado [Dav91].

Uma das primeiras aplicações propostas para os algoritmos genéticos (cuja primazia no uso do termo cabe a Bagley na sua dissertação de 1967), seguindo o enfoque de Holland, foram os sistemas classificadores, que são sistemas de produção e, na verdade, usam os algoritmos genéticos em uma parte do algoritmo global. Apesar do interesse que levantaram na época, os sistemas classificadores permanecem como um campo de estudo em grande parte ainda inexplorado.

Outro ramo descendente dos algoritmos genéticos é o da *programação genética (PG)* . Aqui, os indivíduos da população não são seqüências de ``bits'', mas sim programas de computador armazenados na forma de árvores sintáticas. Tais programas é que são os candidatos à solução do problema proposto. A programação genética não usa o operador mutação e a recombinação se dá pela troca de subárvores entre dois indivíduos candidatos à solução.

Outro ramo da CE é a *programação evolutiva (PE)* , que visa prever o comportamento de máquinas de estado finitas. Apenas dois operadores são usados: a seleção e a mutação. As idéias datam de 1966, e não foram muito consideradas na comunidade de CE por rejeitar o

papel fundamental da recombinação. Finalmente, um último ramo é a *estratégia evolutiva* , proposta nos anos 60, na Alemanha. A ênfase aqui é na auto-adaptação. O papel da recombinação é aceito, mas como operador secundário.

Embora tenham origens bastante diversas, todas essas abordagens têm em comum o modelo conceitual inicial - a evolução natural -, além dos operadores e, mais importante, o mesmo objetivo final: a solução de problemas complexos[BS93].

**Idéias básicas**

A CE está baseada em algumas idéias básicas que, quando implementadas, permitem simular em um computador o processo de passagem de gerações da evolução natural. As idéias que permitem esta simulação são as seguintes:

- A criação de uma população de soluções, possivelmente obtida na sua primeira geração de modo aleatório, e na qual os indivíduos tenham registrado de modo intrínseco os parâmetros que descrevem uma possível solução ao problema posto.
- A criação de uma entidade - chamada *função de avaliação* - capaz de julgar a aptidão de cada um dos indivíduos. Essa entidade não precisa deter conhecimento sobre como encontrar uma solução para o problema, mas apenas atribuir uma ``nota" ao desempenho de cada um dos indivíduos da população.
- E, finalmente, a criação de uma série de operadores que serão aplicados à população de uma dada geração para obter os indivíduos da próxima geração. Estes operadores são baseados nos fenômenos que ocorrem na evolução natural. Os principais operadores citados na literatura são: (i) *seleção*: permite escolher um indivíduo ou um par deles para gerar descendência. Note-se que este operador simula a reprodução assexuada (no primeiro caso) e a sexuada (no segundo) que ocorrem na natureza. Obviamente, a prioridade da escolha recai sobre indivíduos mais bem avaliados pela função de avaliação; (ii) *recombinação*: operador que simula a troca de material genético entre os ancestrais que, por sua vez, determina a carga genética dos descendentes; (iii) *mutação*: operador que realiza mudanças aleatórias no material genético.

O conceito chave na CE é o de *adaptação* que unifica a abordagem quanto ao método de solução: uma população inicial de soluções evolui, ao longo das gerações que são simuladas no processo, em direção a soluções mais adaptadas, isto é, com maior valor da função de avaliação, por meio de operadores de seleção, mutação e recombinação.

O conjunto de soluções iniciais pode ser aleatório ou pode ser obtido a partir de técnicas convencionais para resolver instâncias mais simples do problema que está sendo tratado. Por um lado, usando-se soluções inicialmente aleatórias, pode-se usar sempre o mesmo algoritmo e os mesmos operadores. Por outro lado, adaptando o conceito de CE a um problema específico e empregando soluções iniciais obtidas por métodos convencionais, é necessário adaptar os operadores usuais da CE para o problema específico. Em compensação, na pior das hipóteses, a solução encontrada é igual à melhor solução obtida anteriormente pelas técnicas convencionais.

Para definir a função de avaliação é necessário encontrar uma maneira de codificar as soluções para o problema que se quer resolver. O resultado dessa codificação corresponde aos cromossomos na evolução natural e é chamado de *genótipo*. A partir desses cromossomos, a função de avaliação deve ser capaz de determinar a qualidade de uma solução.

As novas soluções podem ser geradas a partir de uma única solução (assexuada, na natureza) ou a partir de duas soluções (na natureza, sexuada). Estabelecido um conjunto de novas soluções (os descendentes), estas sofrem a ação dos operadores evolutivos, mediante os quais, os descendentes passarão a ser diferentes dos ascendentes. Os melhores, de acordo com a função de avaliação, terão uma descendência maior do que a dos pouco aptos. Na reprodução sexuada, a troca de material genético - chamada de *recombinação* - leva um par de ascendentes a dar origem a um par de descendentes onde cada descendente herda partes aleatoriamente escolhidas de cada ascendente. A mutação leva à mudança, também aleatória, de uma parte da solução. No caso mais simples de cromossomos codificados em binário, a mutação é a simples inversão de um ``bit''. Tanto a recombinação quanto a mutação tendem a ocorrer segundo uma dada probabilidade (que são parâmetros da técnica).

Tal processo, repetido, simula a passagem das gerações. Como o processo está sendo simulado em um computador digital, o fator tempo pode ser comprimido sem perda de qualidade.

**Formalização**

Uma possível formalização algorítmica para a CE foi proposta por [BS93]. Seja *I* o conjunto de indivíduos e $\mathbb{R}$ o conjunto dos números reais. $\alpha(t) \in I$ denota um indivíduo na geração *t* e $\vec{x} \in \mathbb{R}^n$ indica as coordenadas de um ponto no espaço de soluções. $\mu \geq 1$ denota o tamanho da população de ancestrais, e $\lambda \geq 1$ é o tamanho da população de descendentes. Definem-se as seguintes funções:

Função objetivo a ser otimizada: $f : \mathbb{R}^n \rightarrow \mathbb{R}$.

Função de avaliação: $\Phi : I \rightarrow \mathbb{R}$.

A população na geração *t* - $P(t) = \{\alpha_1(t), \ldots, \alpha_\mu(t)\}$ - consiste de indivíduos $\alpha_i(t) \in I$, onde cada indivíduo codifica, com um grau de qualidade dado por $\Phi(\alpha_i(t))$, a função objetivo *f*. Os operadores são definidos da seguinte maneira:

Seleção: $s_{\Theta_s} : (I^\lambda \cup I^{\lambda+\mu}) \rightarrow I^\mu$.

Recombinação: $r_{\Theta_r} : I^\mu \rightarrow I^\lambda$.

Mutação: $m_{\Theta_m} : I^\lambda \rightarrow I^\lambda$.

Os operadores são controlados, respectivamente, pelos conjuntos de parâmetros $\Theta_r$, $\Theta_m$ e $\Theta_s$. Durante a etapa de avaliação, a função de avaliação $\Phi$ é calculada para todos os elementos da população. O critério de fim é dado por:

$$\iota : I^{\mu} \rightarrow \{V, F\}$$

onde *V* e *F* são valores verdade. Finalmente, *Q* é o conjunto de indivíduos que são tomados adicionalmente durante a etapa de seleção. Se *Q* é vazio, os indivíduos de uma geração não são usados como candidatos a ascendentes na próxima geração. Se $Q = P(t)$, todos os indivíduos de uma geração não são utilizados na próxima. De acordo com essa notação, pode-se descrever o algoritmo básico da CE da seguinte maneira:

1. $t \leftarrow 0$
2. População inicial: $P(0) \leftarrow \{\alpha_1(0), \ldots, \alpha_{\mu}(0)\} \in I^{\mu}$
3. Avaliação: $\{\Phi(\alpha_1(0)), \ldots, \Phi(\alpha_{\mu}(0))\}$
4. Enquanto $\iota(P(t)) \neq V$ faça:
   - Recombinação: $P'(t) \leftarrow r_{\Theta_r}(P(t))$
   - Mutação: $P'(t) \leftarrow m_{\Theta_m}(P'(t))$
   - Avaliação: $\{\Phi(\alpha_1(0)), \ldots, \Phi(\alpha_{\lambda}(0))\}$

- Seleção: $P(t+1) \leftarrow s_{\Theta_s}(P'(t) \cup Q)$
- $t \leftarrow t+1$

Dentro da CE existem diversas correntes ou abordagens. Todas compartilham a simulação do processo de seleção natural, mas se diferenciam pela origem e por suas estratégias de solução. Nas seções seguintes descrevem-se algumas destas abordagens.

**Estratégia evolutiva**

A *estratégia evolutiva (EE)* teve origem em 1964 na Universidade Técnica de Berlim, Alemanha. O problema original em estudo era o de encontrar formas otimizadas para objetos inseridos em fluxos de vento. Estratégias usando o método do gradiente não foram bem sucedidas. Dois estudantes, Ingo Rechenberg e Hans-Paul Schwefel tiveram a idéia de efetuar alterações randômicas nos parâmetros que definiam a forma do objeto, baseados na idéia da seleção natural, o que resultou na teoria da velocidade de convergência para o mecanismo denominado (1+1), um esquema simples de seleção-mutação trabalhando em um único indivíduo que gera um único descendente por geração através da mutação Gaussiana [BS93]. Mais tarde, esta teoria evoluiu para o chamado mecanismo $(\mu+1)$, no qual uma população de $\mu$ indivíduos se recombina de maneira randômica para formar um descendente, o qual, após sofrer mutação, substitui (se for o caso) o pior elemento da população. Ainda que este mecanismo nunca tenha sido largamente usado, ele permitiu a transição para os mecanismos chamados $(\mu+\lambda)$ e $(\mu,\lambda)$, já no final dos anos 70. Na primeira, os $\mu$ ancestrais e os $\lambda$ descendentes convivem, enquanto na segunda, os $\mu$ ancestrais ``morrem", deixando apenas os $\lambda$ descendentes ``vivos".

A abordagem EE é adequada a uma vasta gama de problemas de otimização, porque ela não necessita de muitas informações sobre o problema. Ela é capaz de resolver problemas multidimensionais, multimodais e não lineares sujeitos a restrições lineares ou não lineares [HB94]. O algoritmo básico conforme [Sch95] é o seguinte:

1. Uma dada população consiste de $\mu$ indivíduos. Cada um é caracterizado pelo seu genótipo, consistindo de *n* genes, que determina de modo não ambíguo a aptidão para a sobrevivência.
2. Cada indivíduo da população produz $\frac{\lambda}{\mu}$ descendentes, na média, de modo que um total de $\lambda$ indivíduos novos são gerados. O genótipo dos descendentes difere ligeiramente dos genótipos de seus ancestrais.
3. Apenas os $\mu$ melhores indivíduos dos $\lambda$ gerados permanecem vivos, tornando-se os ancestrais na próxima geração.

Note-se que até recentemente a EE era conhecida apenas na comunidade de engenharia e tida como alternativa a soluções padrão para problemas de otimização [HB94].

**Estratégia** 1+1

Na estratégia (1+1), uma solução gera outra a cada geração. Nessa operação é aplicada uma mutação normal (ou seja, pequenas alterações têm maior probabilidade de ocorrer do que grandes alterações, seguindo a distribuição normal), até que o descendente tenha um desempenho melhor que seu ascendente, quando então ele lhe toma o lugar.

Na estratégia evolucionária, um indivíduo corresponde a um ponto no espaço de soluções, e possui um genótipo formado por variáveis objetivo e variáveis estratégicas. As variáveis objetivo são aquelas que, sofrendo recombinação e mutação, permitem incrementar a aptidão dos indivíduos em direção ao máximo global de otimização. As variáveis estratégicas representam variâncias e co-variâncias, que devem ser operadas junto às variáveis de controle para produzir mutações.

Devido à relativa simplicidade deste esquema, é que a regra do 1/5 pode ser aplicada [HB94]. Esta regra diz que quando a taxa de sucesso (isto é, o descendente tem aptidão

maior do que o ascendente), após mutação é maior do que 0,2, a variância da distribuição normal deve ser aumentada, enquanto se ela for menor do que 0,2, a variância deve ser diminuída [Sch95]. Registre-se que cada indivíduo detém a sua própria variância e eventualmente co-variância (que são os parâmetros básicos da mutação), o que caracteriza o conceito de auto-adaptação.

**Estratégias soma e vírgula**

As estratégias $(\mu + \lambda)$ e $(\mu, \lambda)$, são chamadas *estratégia soma* e *estratégia vírgula* respectivamente. Na estratégia soma, os ascendentes são levados em conta durante a etapa de seleção, enquanto na estratégia vírgula, apenas os descendentes de uma dada geração são candidatos a serem selecionados para gerar a próxima. Esta característica dos ancestrais competirem junto com os descendentes frente ao operador seleção é chamada de elitismo. A escolha adequada de $\frac{\mu}{\lambda}$ determina a velocidade de convergência da EE [HB94].

**Programação evolutiva**

A *programação evolutiva (PE)* foi proposta por Fogel, Owens e Walsh em meados da década de 60. Ainda que a proposta original tratasse de predição de comportamento de máquinas de estado finitos, o enfoque da PE se adapta a qualquer estrutura de problema. Neste enfoque, cada indivíduo gera um único descendente através de mutação, e a seguir a (melhor) metade da população ascendente e a (melhor) metade da população descendente são reunidas para formar a nova geração. Usando a terminologia da EE, esta implementação poderia ser nomeada como $(\mu + \mu)$ [BS93].

Em 1992, na sua tese de doutorado, Fogel apresentou o conceito de programação metaevolucionária, na qual um vetor de variâncias substitui o valor padrão e exógeno da taxa de mutação, e aproxima este conceito da auto-adaptação descrita anteriormente para a EE.

O algoritmo básico segue os seguintes passos:

1. Escolhe-se uma população inicial de soluções de maneira randômica. O número de soluções é relevante para a velocidade da otimização.
2. Cada solução gera uma nova população, cujas soluções sofrem mutação de acordo com uma distribuição de taxas de mutação.
3. Cada solução tem sua aptidão calculada. Os mais aptos são retidos como população de soluções. Não se exige que a população permaneça constante, ou que cada ascendente gere apenas um descendente.

A mutação é o único operador que atua na programação evolucionária. Não há recombinação

**Algoritmos genéticos**

Os *algoritmos genéticos (AG)* são o ramo mais conhecido da CE, e tiveram origem no trabalho de Holland, também nos anos sessenta.[2] Ao contrário dos dois esquemas vistos acima - EE e PE -, os AG's conceitualmente apresentam um escopo mais amplo do que a simples otimização. Eles são apresentados como um modelo para a aprendizagem de máquina [HB94]. Há uma explicação para este fato: originalmente, os AG's estavam muito fortemente ligados a modelos de aprendizado automático, como o demonstra a ênfase dada por Holland em [Hol75] aos chamados sistemas classificadores, que são um modelo de máquina de aprendizado usando AG's. Só mais tarde, a partir da publicação do livro *Genetic algorithms in search, optimization, and machine learning* [Gol89], que a idéia de otimização passou a ocupar o lugar central na teoria dos AG's. No livro [Hol75], o autor introduz o assunto no âmbito da genética, economia, teoria de jogos, pesquisa, reconhecimento de padrões e inferência estatística, controle e otimização de funções e sistema nervoso central.

Um questão importante para a implementação dos AG's em particular e para a CE em geral é o papel dos operadores de mutação e de recombinação. Por exemplo, a estratégia PE estabelece que apenas a mutação deve ser usada. Quanto ao operador de recombinação, a

dúvida é sobre reprodução sexuada ou não. A reprodução sexuada, se comparada com a reprodução assexuada, tem um custo ``a priori" maior, pois espécies usando reprodução sexuada precisam ter mais de um tipo de indivíduo na espécie e estes tendem a gastar tempo e energia buscando seus parceiros antes da reprodução acontecer. Visto que a reprodução sexuada levou a melhores resultados na evolução natural, este custo adicional deve ser contrabalançado por outras vantagens.

Uma destas vantagens é que a reprodução sexuada permite a rápida combinação de novas características benéficas de uma maneira que não pode ser duplicada pela mutação. Para promover as boas soluções, a técnica básica recebeu o nome de ``roda da roleta". A idéia é que todos os indivíduos de uma população sejam avaliados, e o resultado da avaliação seja usado como abertura angular em uma roleta. Em outras palavras, indivíduos aptos teriam um grande ângulo nesta roleta, enquanto indivíduos menos aptos teriam ângulos cada vez menores. Jogada a bola (que em termos computacionais significa a geração de um número pseudo-aleatório), aqueles que tiverem maiores ângulos terão maior chance de serem escolhidos como ascendentes e é através deste mecanismo que a aptidão média da população vai sendo incrementada.

Com o passar das gerações, percebe-se que as soluções ``boas" começam a compartilhar partes comuns em seus cromossomos. Estas partes foram chamadas de esquemas, e o teorema fundamental dos AG's diz que esquemas que tiverem maior aptidão (o resultado da função de avaliação) do que a média da população tendem a crescer exponencialmente nas próximas gerações, enquanto que os esquemas que tiverem aptidões menores do que a média tendem a diminuir também exponencialmente, isto é, as soluções convergirão para um ponto de maior aptidão.

O uso direto do resultado da função de avaliação como aptidão diminui a robustez dos algoritmos, até porque torna a abordagem extremamente dependente da solução de avaliação. Assim, foram desenvolvidas algumas técnicas de transformação de avaliação em aptidão. Antes de se ver como são essas técnicas, vejam-se duas ocorrências comuns e ambas indesejáveis.

A primeira, chamada de *superindivíduo* , ocorre quando um indivíduo tem uma avaliação muitas vezes maior do que os outros. Se nenhum cuidado especial for tomado, este indivíduo acabará monopolizando as seleções e rapidamente ter-se-ão soluções praticamente idênticas (até porque descendem do mesmo ascendente). Este fenômeno também é chamado convergência precoce. Para corrigir este problema, uma técnica possível é a chamada normalização linear, que coloca os indivíduos em ordem decrescente de avaliação e atribui um valor de aptidão diminuído em taxa constante para todos. Neste caso, o valor da função de avaliação serviu apenas para determinar a ordem dos indivíduos, e o valor de aptidão por eles obtido não tem mais nada a ver com a avaliação original.

Outra ocorrência que não é boa é o surgimento de um grupo de indivíduos com avaliações muito baixas. Se nada for feito, tais indivíduos terão pequena probabilidade de serem escolhidos para gerar descendentes. No entanto, é importante que eles gerem descendentes, pois eles são a fonte da diversidade das soluções. Se exterminados do processo logo no começo, a população também rapidamente tenderá para uma convergência precoce. Se a função objetivo for ``bem comportada", por exemplo, tendo um único máximo global, a convergência precoce em si não é um mal. Mas no mundo real, poucas funções são ``bem comportadas". Elas em geral são multimodais e, neste caso, a convergência precoce pode-se dar em um máximo local. Utilizando-se soluções com a maior diversidade possível, aumenta-se a possibilidade de - quando se chegar a um ótimo - este ser global. A correção para este segundo problema pode ser a chamada técnica de janelamento, pela qual é determinado um valor mínimo de aptidão que é atribuído a todos aqueles que ficarem abaixo de um certo piso na avaliação.

Finalmente, como os AG's são estocásticos por excelência, boas soluções (que portanto tenham boas aptidões), podem deixar de ser escolhidas para gerar descendência. Uma maneira de evitar este fato é usar o conceito de *elitismo* . Por esta técnica, as *n* melhores soluções de uma geração passarão automaticamente para a próxima.

O algoritmo genético básico é o seguinte:

1.

    Inicializar a população de cromossomos (soluções).

2. Avaliar cada cromossomo da população.
3. Criar novos cromossomos a partir da população atual, aplicar mutação e recombinação, substituindo os ascendentes pelos descendentes.
4. Se o critério de fim foi alcançado, deve-se terminar. Caso contrário, retorna-se ao passo 1.

**Hibridização**

A técnica de hibridização resulta na integração de uma boa maneira convencional de resolver um problema aos conceitos usuais de AG's. O resultado costuma ser melhor que o obtido com qualquer uma das duas técnicas isoladamente [Dav91]. A hibridização agrega a representação usual de dados no domínio original, bem como as técnicas de otimização já existentes. Isto permite a incorporação de heurísticas otimizadoras ao conjunto de operadores genéticos (recombinação e mutação) que passam portanto a ser dependentes do domínio. Nesse sentido, o algoritmo genético passa a ser muito mais uma filosofia de otimização do que um método pronto para utilização.

Um exemplo de hibridização possível é quando o problema exige codificação com base em números reais e não em números binários. Alguns conceitos teriam que ser adaptados: por exemplo, a mutação não seria mais a troca simples de um ``bit'', mas a geração de um novo real, possivelmente dentro de um intervalo dado. Já a recombinação de dois reais poderia ser qualquer número compreendido entre eles, ou talvez a sua média.

Outra possibilidade de hibridização é quando o problema envolve algoritmos de ordenamento como, por exemplo, os problemas do caixeiro viajante e da coloração de um grafo sem que nenhum par de nodos conectados tenham cor igual.

Representação de incerteza

A imperfeição da informação é geralmente conhecida na literatura de sistemas baseados em conhecimento como *incerteza* . No entanto, este termo é muito restritivo; o que se convenciona chamar tratamento de incerteza pode, na verdade, estar endereçando outras imperfeições da informação, como imprecisão, conflito, ignorância parcial, etc.

Suponhamos, por exemplo, que queiramos descobrir a que horas começa um determinado filme. Algumas das respostas que podemos obter são:

- Informação perfeita: O filme começa às 8h 15min.
- Informação imprecisa: O filme começa entre 8h e 9h.
- Informação incerta: Eu acho que o filme começa às 8h (mas não tenho certeza).
- Informação vaga: O filme começa lá pelas 8h.
- Informação probabilista: É provável que o filme comece às 8h.
- Informação possibilista: É possível que o filme comece às 8h.
- Informação inconsistente: Maria disse que o filme começa às 8h, mas João disse que ele começa às 10h.
- Informação incompleta: Eu não sei a que horas começa o filme, mas usualmente os filmes neste cinema começam às 8h.
- Ignorância total: Eu não faço a menor idéia do horário do filme.

As informações que podemos obter podem, portanto, variar de perfeitas, quando descobrimos exatamente o que queremos saber, a completamente imperfeitas, seja pela total ausência de informações ou por informações completamente conflitantes. O mais interessante aqui é que, mesmo lidando diariamente com o tipo de informações acima, conseguimos tomar decisões razoáveis. Para tanto, nós, de alguma forma, encontramos um modelo adequado para representar a informação que obtivemos e a tratamos segundo o modelo escolhido. O mesmo deve (ou pelo menos deveria) ocorrer com sistemas baseados em conhecimento, em face de informações imperfeitas.

Por exemplo, para cada um dos tipos de informação descritos acima, existe um modelo formal conhecido de tratamento. Além disso, encontramos na literatura implementações,

formais ou ``ad hoc", para cada um desses modelos. A informação de conotação probabilista pode ser tratada pela teoria de probabilidades ou pela teoria da evidência (também conhecida como Dempster-Shafer) [Sha76]; enquanto que a informação imprecisa e/ou vaga pode ser tratada pela teoria dos conjuntos nebulosos ([Zad65], [DP80]), pela teoria dos *conjuntos de aproximação* (do inglês, ``rough sets") [Paw92], ou pela manipulação de classes de referência [KJ83]; e a informação possibilista pode ser tratada pela teoria de possibilidades ([Zad78], [DP88]). A informação incerta pode ser tratada tanto pelas teorias de probabilidades, possibilidades ou evidência, ou por modelos ``ad hoc". As informações inconsistentes e aquelas que chamamos, um pouco impropriamente, incompletas, podem ser tratadas por lógicas não clássicas; como a paraconsistente e a de quatro valores no primeiro caso [Bel77], e as lógicas não monotônica no segundo caso, como a lógica de ``default" [Rei80] e a circunscrição [McC80].[3]

Nesta seção, estamos interessados em discutir alguns dos modelos numéricos de representação da informação imperfeita. Embora reconhecidamente importantes, não abordaremos modelos de propagação local de informação imperfeita ([Pea87], [San93]), nem modelos para agregação de opiniões de especialistas ([SDP95]).

A seguir descrevem-se os modelos numéricos mais conhecidos para a representação da informação imperfeita: os modelos probabilista, possibilista, nebuloso e da evidência.

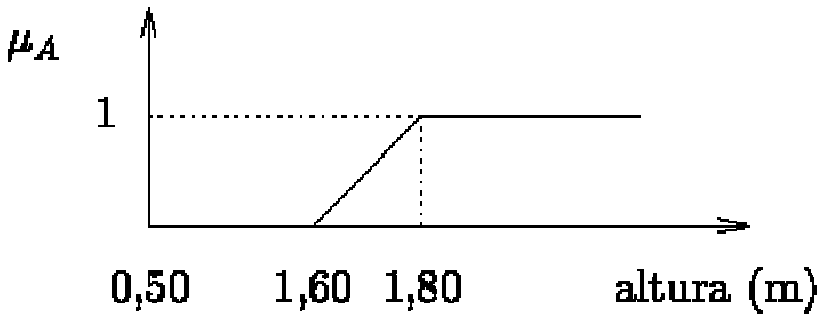
Modelo nebuloso

A *teoria dos conjuntos nebulosos* é o modelo mais tradicional para o tratamento da informação imprecisa e vaga. Este modelo, introduzido em [Zad65], tem por objetivo permitir graduações na pertinência de um elemento a uma dada classe, ou seja de possibilitar a um elemento de pertencer com maior ou menor intensidade àquela classe. Basicamente, isso se faz quando o grau de pertinência de um elemento ao conjunto, que na teoria dos conjuntos ``clássica" assume apenas os valores 0 ou 1, passa a ser dado por um valor no intervalo dos números reais [0,1]. Para uma descrição detalhada desta teoria, ver [DP80] e [DP88], e para uma descrição mais sucinta, ver [BM93b]).

Dado um universo de discurso *X*, um subconjunto nebuloso *A* de *X* é definido por uma função de pertinência que associa a cada elemento *x* de *X* o grau $\mu_A(x)$, compreendido entre 0 e 1, com o qual *x* pertence a *A* [Zad65]:

$$\mu_A : X \rightarrow [0,1]$$

**Figure:** Conjunto nebuloso A $\equiv$ ``alto"




**Exemplo:** Suponhamos que queiramos modelar o conceito ``alto". Usualmente, podemos dizer com certeza que uma pessoa que mede mais de 1,75 *m* é alta, e que ela não é alta se tiver menos de 1,60m. Já uma pessoa que mede entre 1,60 *m* e 1,75 *m* será considerada tanto mais alta quanto mais sua altura esteja próxima de 1,75 *m*. Então podemos modelar o conceito ``alto" pelo conjunto nebuloso *A*, definido no intervalo de 0,5 *m* a 2,5 *m*, dado por:

$$\mu_A(x) = \begin{cases} 1 & x > 1,75m \\ 0 & x < 1,60m \\ \frac{x-1,6}{0,15} & 1,60m \leq x \leq 1,75m \end{cases}$$

A figura 2 ilustra o conjunto nebuloso ``alto".

Na teoria dos conjuntos nebulosos, a negação $n : [0,1] \to [0,1]$ é implementada por uma família de operadores, sendo que o mais comumente utilizado é dado por $n(x) = 1 - x$. Ou seja, se $\overline{A}$ representa a negação de $A$ no universo $Y$, então utilizando a negação $n$ temos $\mu_{\overline{A}}(x) = 1 - \mu_A(x)$.

Nesta teoria, a intersecção é implementada por uma família de operações - chamadas *T-normas* - e a união é implementada por outra família de operações - chamadas *T-conormas* [DP88]. O conjunto das T-normas e T-conormas formam as *normas triangulares* . Cada norma triangular $\nu : [0,1] \times [0,1] \to [0,1]$ verifica, para todos os $x$, $y$, $z$ em [0, 1], propriedades abaixo.

(i)

Comutatividade: $\nu(x,y) = \nu(y,x)$

(ii)

Associatividade: $\nu(x,\nu(y,z)) = \nu(\nu(x,y),z)$

(iii)

Monotonicidade: se $x \leq z$ e $y \leq t$, então $\nu(x,y) \leq \nu(z,t)$

Além disso, cada T-norma $\top$ verifica a propriedade:

(iv)

$\top(x,1) = x$ (elemento neutro 1),

e cada T-conorma $\perp$ verifica a propriedade:

(v)

$\perp(x, 0) = x$ (elemento neutro 0).

Uma T-norma $\top$ e uma T-conorma $\perp$ são duais em relação a uma operação de negação *n* se elas satisfazem as relações de De Morgan, isto é, se $n(\top(A, B)) = \perp(n(A), n(B))$ e $n(\perp(A, B)) = \top(n(A), n(B))$. As T-normas e T-conormas duais, em relação à operação de negação 1 - *x*, mais utilizadas são apresentadas no quadro 2.

**Table 2:** Principais T-normas e T-conormas duais

| T-norma | T-conorma | Nome |
|---|---|---|
| *min*(*x*, *y*) | *max*(*x*, *y*) | Zadeh |
| $x \cdot y$ | *x* + *y* - *xy* | Probabilista |
| *max*(*x* + *y* - 1, 0) | *min*(*x* + *y*, 1) | Lukasiewicz |
| $\frac{xy}{\gamma+(1-\gamma)(x+y-xy)}$ | $\frac{x+y-xy-(1-\gamma)xy}{1-(1-\gamma)xy}$ | Hamacher ( $\gamma > 0$ ) |
| *x*, se *y* = 1 | *x*, se *y* = 0 | Weber |
| *y*, se *x* = 1 | *y*, se *x* = 0 | |
| 0 se não | 1 se não | |

Os *operadores de implicação* são usados para modelar regras de inferência do tipo ``Se ⟨premissa⟩ então ⟨conclusão⟩''. Seja $\alpha$ o grau de compatibilidade entre as condições estabelecidas na premissa e os valores encontrados na realidade e *C*, definido em *Z*, o conjunto nebuloso presente na conclusão da regra. Então, para verificarmos o grau com que a premissa implica a conclusão, dados os valores encontrados na realidade, verificaremos o quanto $\alpha$ implica $\mu_C(z)$, verificando $I(\alpha, \mu_C(z))$ para todo $z \in Z$. O quadro 3 traz as principais operações de implicação encontradas na literatura.

**Table:** Principais operadores de implicação

| Implicação | Nome |
|---|---|
| $max(1 - x, y)$ | Kleene |
| $min(1 - x + y, 1)$ | Lukasiewicz |
| 1, se $x \leq y$ | Gödel |
| $y$, se não | |
| $min(x, y)$ | Mandani |
| $x \cdot y$ | Larsen |

A utilização mais significativa da teoria dos conjuntos nebulosos em sistemas baseados em conhecimento são os controladores nebulosos [Lee90], [DHR93]. Um controlador nebuloso pode ser visto como um sistema especialista simplificado, onde a conseqüência de uma

regra não é aplicada como antecedente de outra. Isto porque as ações de controle são baseadas em um único nível de inferência. As regras de controle nebuloso são usualmente do tipo

$R_j$ : Se $x_1$ é $A_{1j}$ e ⋯ e $x_n$ é $A_{nj}$ então $y$ é $B_j$.

onde os $A_{ij}$ e $B_j$ são conjuntos nebulosos, e a implicação é implementada como uma função de implicação nebulosa. Em cada ciclo do processo, os valores das variáveis xij são medidos e então comparados aos conjuntos nebulosos Aij nas regras, gerando uma medida da adequação dos valores medidos à premissa da regra (utilizando uma T-norma para implementar o conectivo ``e'' na premissa das regras). A implicação então utiliza esta medida de adequação e o conjunto nebuloso $B_j$ na conclusão para obter um valor $B_j'$ para a variável de controle, em relação àquela regra. Os valores $B_j'$ são então agregados em uma única ação de controle $C$, utilizando uma T-norma ou T-conorma. Em alguns controladores os $B_j'$ e $C$ são nebulosos, e então é necessário se determinar um valor preciso para a variável de controle, a partir de $C$.

Modelo probabilista

A *teoria de probabilidades* é o modelo mais tradicional para o tratamento da informação incerta. Neste modelo, dado um evento $A \subseteq X$, onde $\subseteq$ denota a inclusão não estrita, a probabilidade da ocorrência de $A$ é descrita por uma medida de probabilidade $\mathcal{P} : 2^X \rightarrow [0,1]$, que satisfaz os seguintes axiomas:

$$\forall A \subseteq X, \mathcal{P}(A) \geq 0, \mathcal{P}(A) \leq \mathcal{P}(X)$$

$$\mathcal{P}(X) = 1$$

$$\forall A, B \subseteq X, A \cap B = \emptyset, \mathcal{P}(A \cup B) = \mathcal{P}(A) + \mathcal{P}(B)$$

onde $\emptyset$ denota o conjunto vazio. O primeiro axioma estabelece que todas as probabilidades sobre os eventos $A \subseteq X$ são não negativas e que *X* é o evento mais provável. O segundo axioma estabelece que a probabilidade máxima é dada a *X*, que é interpretado como o evento certo. O axioma da aditividade estabelece que, se dois eventos são mutuamente exclusivos, então a probabilidade de que ao menos um dentre eles se realize é a soma de suas probabilidades individuais. Uma conseqüência importante deste axioma é que, se a probabilidade de um evento *A* é conhecida, então pode-se determinar exatamente a probabilidade do evento contrário, $\overline{A}$. A partir desses axiomas pode-se estabelecer as seguintes propriedades:

$$\mathcal{P}(\emptyset) = 0,$$

$$\forall A \subseteq X, \mathcal{P}(A) = 1 - \mathcal{P}(\overline{A}),$$

$$\forall A, B \subseteq X, \mathcal{P}(A \cup B) = \mathcal{P}(A) + \mathcal{P}(B) - \mathcal{P}(A \cap B).$$

Podemos, assim, caracterizar a probabilidade dos eventos sobre um domínio discreto *X*, utilizando a distribuição de probabilidade $p : X \to [0,1]$, que é tal que $\sum_{x \in X} p(x) = 1$. A função *p* pode ser definida a partir de P por $p(x) = \mathcal{P}(\{x\})$. A partir do axioma de aditividade deduz-se que:

$$\forall A \subseteq X, \mathcal{P}(A) = \sum_{x \in A} p(x).$$

No caso contínuo, *p* é chamada de função densidade de probabilidade que é tal que $\int_X p(x)dx = 1$, e $\mathcal{P}(A) = \int_A p(x)dx$.

A medida de probabilidade tem uma interpretação freqüentista e uma interpretação subjetiva. Na interpretação freqüentista, que é a base da utilização de informações estatísticas, P(A) representa o limite da freqüência da ocorrência do evento A em uma seqüência infinita de experiências independentes. Na interpretação subjetiva, mais freqüentemente usada em sistemas baseados em conhecimento, $\mathcal{P}(A)$ representa a crença de um determinado indivíduo na ocorrência de *A*. Neste caso, supõe-se que o indivíduo que fornece as probabilidades aos eventos em *X* seja capaz de exprimi-las de forma a que elas obedeçam os axiomas acima.

O *teorema de Bayes* provê a base para o tratamento da imperfeição da informação em diversos sistemas baseados em conhecimento [Ric83]. Este teorema computa a probabilidade de um dado evento, dado um conjunto de observações. Seja:

$\mathcal{P}(H_i \mid E)$ a probabilidade que a hipótese $H_i$ seja verdadeira dada a evidência *E*.

$\mathcal{P}(E \mid H_i)$ a probabilidade que a evidência *E* será observada se a hipótese $H_i$ for verdadeira.

$\mathcal{P}(H_i)$ a probabilidade ``a priori" que a hipótese $H_i$ é verdadeira na ausência de qualquer evidência específica.

$k$ o número de hipóteses possíveis.

O teorema de Bayes é formulado como:

$$\mathcal{P}(H_i \mid E) = \frac{\mathcal{P}(E \mid H_i) \cdot \mathcal{P}(H_i)}{\sum_{j=1}^{k} \mathcal{P}(E \mid H_j) \cdot \mathcal{P}(H_j)}.$$

Em sistemas baseados em conhecimento, uma modificação desta regra é muitas vezes utilizada [BS84]:

$$\mathcal{P}(H_i \mid E_n \cup E_a) = \frac{\mathcal{P}(E_n \mid H_i \wedge Ea) \cdot \mathcal{P}(H_i \mid E_a)}{\sum_{j=1}^{k} \mathcal{P}(E_n \mid H_j \wedge E_a) \cdot \mathcal{P}(H_j \mid E_a)}$$

onde $E_n$ e $E_a$ representam respectivamente novas evidências e evidências anteriores, em relação a um dado momento do processo.

**Exemplo:** Suponhamos que no meio da noite dispare o alarme contra ladrões de nossa casa [Pea87]. Queremos então saber quais são as chances de que esteja havendo uma tentativa de roubo. Suponhamos que existam 95% de chances de que o alarme dispare quando uma tentativa de roubo ocorre, que em 1% das vezes o alarme dispara por outros motivos, e que em nosso bairro existe uma chance em 10.000 de uma dada casa ser assaltada em um dado dia. Temos então: $P(alarme \mid roubo) = 0,95$, $P(alarme \mid \overline{roubo}) = 0,01$ e $P(roubo) = 10^{-4}$.

Então $P(roubo \mid alarme) = 0{,}00941$. Este valor pode parecer estranho, mas ele pode ser intuitivamente entendido quando verificamos que as chances de haver um roubo e do alarme tocar (0,000095) são muito pequenas em relação às chances de haver um alarme falso (0,01).

Muitos dos sistemas baseados em conhecimento mais famosos têm o enfoque bayesiano como base para o tratamento da informação imperfeita, como, por exemplo, o MYCIN [BS84] e o PROSPECTOR [RJP79]. No sistema MYCIN, tanto regras como fatos têm associados um par [*MB MD*], onde *MB* mede a crença na hipótese (o próprio fato, ou conclusão de uma regra quando a premissa é completamente satisfeita), e *MD* mede a crença na negação dessa hipótese. A máquina de inferência então não somente cria novos fatos, mas também um par [*MB MD*] para este novo fato, utilizando uma variação formalmente imperfeita, porém eficaz, da regra de Bayes.

Modelo possibilista

A *teoria de possibilidades* é um modelo que alia um grande poder de expressividade com uma grande flexibilidade para o tratamento da informação incerta. Neste modelo, a informação fornecida por uma fonte de conhecimento sobre o verdadeiro valor de uma variável *x* em um universo de discurso *X* é codificado sob a forma de uma distribuição de possibilidades $\pi : X \rightarrow [0,1]$ [Zad78], [DP88]. Para qualquer valor de $x_i \in X$, $\pi(x_i)$ reflete até que ponto é possível que $x = x_i$ (partindo do pressuposto que se tem somente um valor verdadeiro).

Uma distribuição de possibilidades $\pi$ relativa à variável *x* pode portanto ser vista como a função de pertinência de um conjunto nebuloso dos valores possíveis de *x* [Zad78]. Estes valores são supostamente mutuamente exclusivos, pois *x* toma somente um valor (seu valor verdadeiro), que pertence a um conjunto universo *X* dado. Normalmente, assume-se também que existe ao menos um valor considerado como completamente possível de ser o verdadeiro valor de *x*. Isto se traduz pela condição de normalização: $\exists x \in X, \pi(x) = 1$.

É importante notar que nada impede que valores distintos de *X* possam ser considerados completamente possíveis simultaneamente.

Conhecendo-se a distribuição de possibilidades, a verossimilhança dos eventos pode ser descrita por duas funções de conjunto: a medida de possibilidade e a medida de necessidade, denotadas respectivamente por $\Pi$ e *N*. Quando é uma função de pertinência de um conjunto ``crisp'' (estritamente clássico) *A*, um evento *B* é dito *possível* se e somente se $A \cap B \neq \emptyset$, e *necessário* se $A \subseteq B$; por definição fazemos $\Pi(B) = 1$ e $N(B) = 1$ nas situações respectivas. No caso geral onde $\pi$ é a função de pertinência de um conjunto nebuloso, as medidas de possibilidade e necessidade são definidas da seguinte maneira:

$$\Pi(A) = sup_{x \in A} \pi(x),$$

$$N(A) = inf_{x \in A} 1 - \pi(x)$$

onde *N*(*A*) quantifica o quanto a evidência disponível suporta a hipótese de que *A* contém o verdadeiro valor de *x*, e $\Pi(A)$ quantifica o quanto a evidência não contradiz esta hipótese. A partir das expressões acima obtemos:

$$N(A) = 1 - \Pi(\overline{A})$$

onde $\overline{A}$ é o complemento de *A* em relação a *X*. A expressão dual exprime que *A* é tanto mais certo quanto mais $\overline{A}$ é impossível.

**Exemplo:** Suponhamos que não se saiba a altura de Carlos, mas que se saiba que ele é alto. Então a distribuição de possibilidade da altura de Carlos pode ser aproximada por aquela do conjunto nebuloso $A$ = ``alto'', ou seja, fazemos $\pi(x) = \mu_A(x)$. Dada a distribuição de possibilidade da altura de Carlos, temos:

$$\Pi([0,5,1,60]) = 0, \quad N([0,5,1,6]) = 0,$$

$$\Pi([1,60,1,75]) = 1, \quad N([1,60,1,75]) = 0,$$

$$\Pi([1,75,2,5]) = 1, \quad N([1,75,2,5]) = 0,34,$$

$$\Pi([1,80,2,5]) = 1, \quad N([1,80,2,5]) = 0,$$

$$\Pi([1,60,2,5]) = 1, \quad N([1,60,2,5]) = 1.$$

Transformações entre os modelos possibilista e probabilista têm sido exploradas na literatura (por exemplo, [Kli90] e [DPS93]), e se tornam a cada dia mais úteis.

A teoria de possibilidades é a base da lógica possibilista *PL1*, que tem sido usada em sistemas baseados em conhecimento [SBM93]. Na lógica *PL1*, todos os elementos de conhecimento têm o formato $(\alpha[K])$, onde $\alpha$ é uma asserção lógica - isto é, uma fórmula bem-formada do cálculo de predicados - e a valuação *K* é um limite inferior da necessidade de $\alpha$ (isto é, $N(\alpha) \geq K$). Novos elementos de conhecimentos são inferidos utilizando um esquema lógico para a dedução de novas asserções lógicas e os axiomas de teoria de possibilidades para a geração das valuações correspondentes. Uma nova extensão desta lógica permite o uso formal de constantes nebulosas e a modelagem de regras graduais [DPS96].

Modelo da evidência

A *teoria de evidências* [Sha76] é um dos modelos mais conhecidos para a representação da incerteza em sistemas baseados em conhecimento. Neste modelo, a informação fornecida por uma fonte de conhecimento a respeito do valor real de uma variável *x*, definida em um universo de discurso *X*, é codificada sob a forma de um *corpo de evidência* sobre *X*. Um corpo de evidência é caracterizado por um par (*F*, *m*), onde *F* é uma família de subconjuntos de *X* (isto é, $F \subseteq 2^X$), e a função de *alocação de massa m* é uma aplicação de $2^X$ no intervalo [0,1], tal que $m(A) > 0$ se e somente se $A \in F$ e . Cada elemento $A \in F$ é chamado *elemento focal* , e $m(A)$ se refere à evidência relativa a *A* unicamente (e não aos subconjuntos de *A*). Nesta teoria, a incerteza ligada ao evento $A \in 2^X$ é medida através de duas funções $Bel : 2^X \rightarrow [0,1]$ e $Pl : 2^X \rightarrow [0,1]$, definidas por:

$$Bel(A) = \sum_{B \subseteq A} m(B),$$

$$Pl(A) = \sum_{B \cap A \neq \emptyset} m(B).$$

A função de *credibilidade Bel* mede a que ponto as informações fornecidas por uma fonte sustentam A. A função de *plausibilidade Pl* mede a que ponto as informações dadas por uma fonte não contradizem A. As funções de credibilidade e plausibilidade estão ligadas pela relação:

*Bel*(*A*) = 1 - *Pl*(*A*).

Isto equivale a dizer que quanto mais se aumenta a evidência sobre uma hipótese, menor se torna a plausibilidade da hipótese contrária.

**Exemplo:** Para ilustrar o uso dessa teoria, vamos estudar o problema de tomada de decisão em relação à perfuração ou não em um determinado local, em busca de petróleo. Suponhamos que possamos aplicar um teste cujos resultados possíveis são *vermelho (V), amarelo (A)* ou *verde (R)*. O valor vermelho indica que o local é seco ( $\{s\}$ ), o valor amarelo indica que o local é seco ou molhado ( $\{s, m\}$ ), e o valor verde indica que o local é ou molhado ou ensopado ( $\{m, e\}$ ). Usando resultados de outras perfurações na mesma área, foram conseguidas as seguintes probabilidades ``a priori'':

$$p(V) = 0,5 \quad p(A) = 0,3 \quad p(R) = 0,2.$$

O corpo de evidência sobre os possíveis estados do local $X = \{s, m, e\}$ é então dado por:

$$m(\{s\}) = 0,5 \quad m(\{s,m\}) = 0,3 \quad m(\{m,e\}) = 0,2.$$

Temos então:

$$Pl(\{s\}) = 0,8, \quad Bel(\{s\}) = 0,5,$$

$$Pl(\{m\}) = 0,8, \quad Bel(\{m\}) = 0,$$

$$Pl(\{e\}) = 0,2, \quad Bel(\{s\}) = 0,$$

$$Pl(\{s,m\}) = 1, \quad Bel(\{s,m\}) = 0,8,$$

$$Pl(\{m,e\}) = 0,5, \quad Bel(\{m,e\}) = 0,5.$$

Dependendo dos custos do teste e da perfuração, e da expectativa de ganho para cada estado do local, pode ou não valer a pena fazer o teste e posteriormente a perfuração.

As teorias de probabilidade e de possibilidade podem ser consideradas como casos particulares da teoria da evidência; as distribuições de probabilidade e possibilidade podem ser caracterizadas por tipos especiais de corpos de evidência.

Na teoria da evidência, a regra universalmente aceita para a combinação de dois corpos de evidência independentes é a chamada *regra de Dempster* [4] [Sha76]. Esta regra é composta de uma regra de combinação conjuntiva e de um passo de normalização:

Regra de Dempster:

$$\mu(A) = \sum_{B \cap C = A} m_1(B) \cdot m_2(C)$$

$$m(A) = \frac{\mu(A)}{(1 - \mu(\emptyset))}$$

onde $\mu(\emptyset) = \sum_{B \cap C = \emptyset} m_1(B) \cdot m_2(C)$, representa o conflito entre os corpos de evidência $(F_1, m_1)$ e $(F_2, m_2)$.

Outras regras podem ser encontradas na literatura (ver [JSS95] para uma discussão mais completa). As seguintes regras, por exemplo, só introduzem uma mudança na normalização da função $\mu$ dada acima:

Regra de Yager:

$$m(X) = \mu(X) + \mu(\emptyset), \quad \text{se} \quad A = X$$

$$m(A) = \mu(A), \quad \text{caso contrário}$$

Regra de Hau e Kashyap:

$$m(A) = \mu(A) + \sum_{B \cap C = \emptyset \wedge B \cup C = A} m_1(B) \cdot m_2(C)$$

Regra de Joshi, Sahasrabudhe e Shankar:

$$m(A) = \mu(A)(1 + \mu(\emptyset)), \quad \text{se} \quad A \neq X$$

$$m(X) = 1 - \sum_{A \neq X} m(A), \quad \text{caso contrário}$$

A regra seguinte modifica a regra de combinação em si e não necessita do passo de normalização [DP86]:

Regra de Dubois e Prade:

$$m(A) = \sum_{B \cup C = A} m_1(B) \cdot m_2(C)$$

A regra de Dempster e a de Dubois e Prade são associativas, e as demais obedecem somente a uma propriedade mais fraca chamada *quasi-associatividade* . Nenhuma das regras é idempotente, o que faz com que a independência dos corpos de evidência seja particularmente importante.

**Exemplo:** Para $X = \{a, b\}$, com o corpo de evidência ($F_1$, $m_1$) dado através de $m_1(a)$ = 0,9, $m_1(b)$ = 0,1, e com o corpo de evidência ($F_2$, $m_2$) dado através de $m_2(a)$ = 0,9, $m_2(ab)$ = 0,1, tem-se:[5]

Regra de Dempster:

$m(a)$ = 0,967, $m(b)$ = 0,0322

Regra de Yager e Regra de Hau e Kashyap:

$m(a)$ = 0,9, $m(b)$ = 0,03, $m(ab)$ = 0,07

Regra de Joshi, Sahasrabudhe e Shankar:

$m(a) = 0{,}963$, $m(b) = 0{,}0321$, $m(ab) = 0{,}0049$

Regra de Dubois e Prade:

$m(a) = 0{,}63$, $m(ab) = 0{,}27$

O *Fril* [Bal87] é um dos sistemas baseados em conhecimento que utilizam a teoria da evidência como base para o tratamento da incerteza. Neste sistema, a cada regra ou fato está associado um par [*S U*], onde *S* e *U* podem ser vistos respectivamente como as funções *Bel* e *Pl*. A máquina de inferência do *Fril* usa a regra de Dempster, além de alguns mecanismos ``ad hoc'', para determinar os pares [*S U*] referentes aos novos fatos gerados. Apesar de não seguir estritamente os conceitos e restrições da teoria da evidência, os resultados obtidos com este sistema em várias aplicações foram plenamente satisfatórios.

Sistemas especialistas

*Sistemas de produção* é um nome genérico para todos os sistemas baseados em *regras de produção* , isto é, pares de expressões consistindo em uma condição e uma ação. A idéia inicial dos sistemas de produção foi introduzida por Post, em 1936, quando ele propôs os hoje chamados *sistemas de Post* [Pos43]. Um sistema de Post consiste em um conjunto de regras para a especificação sintática de transformações sobre cadeias de caracteres, e representa, como demonstrou Post, um método geral para o processamento de dados.

Na sua forma mais simples, um modelo de sistema de produções apresenta dois componentes passivos - o *conjunto de regras* e a *memória de trabalho* - definidos da seguinte forma:

- Regras: conjunto ordenado de pares (LHS, RHS), onde LHS e RHS são seqüências de caracteres.
- Memória de trabalho: uma seqüência de caracteres.

Apresenta, também, um componente ativo - o *interpretador* - que realiza o seguinte procedimento:

> Para cada regra (LHS,RHS), se a seqüência de caracteres LHS está contida na memória de trabalho, então substituir os caracteres LHS na memória de trabalho pelos caracteres de RHS; se não continuar na próxima regra.

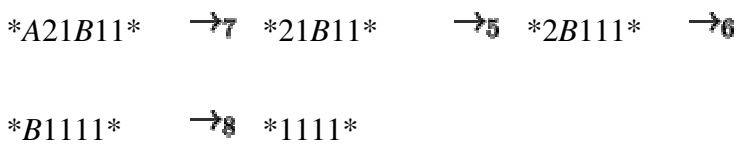
**Exemplo:** Um sistema de produções, correspondendo à definição acima e capaz de multiplicar dois números em notação *unária* (na qual o número *n* é representado por uma seqüência de *n* algarismos 1), pode ser especificado através do seguinte conjunto de regras:[6]

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | $\times 1 * \rightarrow B *$ | 2 | $1 \times 11 \rightarrow 1A \times 1$ | 3 | $1A \rightarrow A21$ | 4 | $2A \rightarrow A2$ |
| 5 | $1B \rightarrow B1$ | 6 | $2B \rightarrow B1$ | 7 | $*A \rightarrow *$ | 8 | $*B \rightarrow *$ |

Inicialmente a memória de trabalho contém o problema a ser resolvido, por exemplo $*11 \times 11*$, onde os caracteres * tem a função de delimitadores. A execução do procedimento interpretador definido acima resultaria na seguinte seqüência de valores como conteúdo da memória de trabalho:

$$*11 \times 11* \rightarrow_2 *11A \times 1* \rightarrow_3$$

$$*1A21B* \rightarrow_3 *A2121B* \rightarrow_5 *A212B1* \rightarrow_6$$

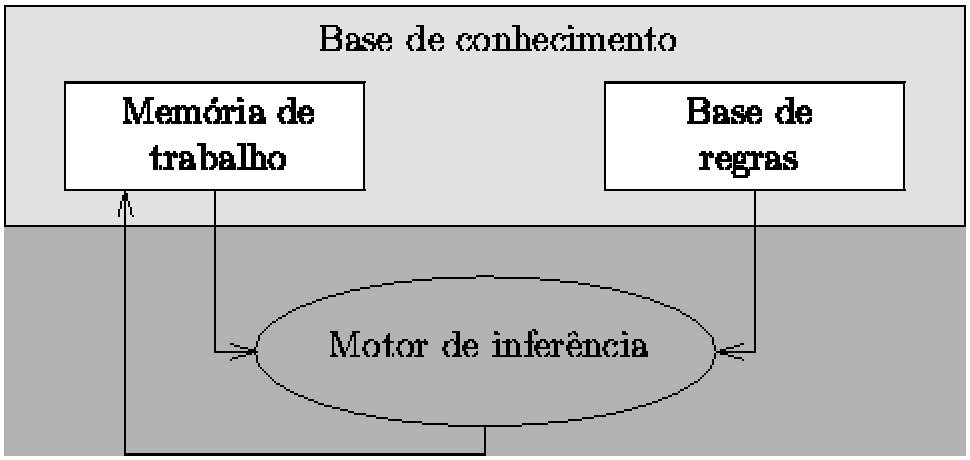



Os sistemas de produção foram redescobertos durante os anos setenta como uma ferramenta para a modelagem da psicologia humana. O formato condição-ação se adapta à modelagem de todos os comportamentos baseados em pares estímulo-resposta. Dois sistemas que utilizaram o modelo de sistemas de produção para a modelagem do comportamento humano foram PAS II [Wat73] e VIS [Mor73].

Outro tipo de sistemas que utilizam o formato de regras de produção como método de representação de conhecimento são os *sistemas especialistas (SE)* . O objetivo dos SE's é, ao mesmo tempo, mais restrito e mais ambicioso do que o objetivo dos modelos psicológicos: os SE's são concebidos para reproduzir o comportamento de especialistas humanos na resolução de problemas do mundo real, mas o domínio destes problemas é altamente restrito. Os primeiros SE's que obtiveram sucesso em seu objetivo foram o sistema DENDRAL [FBL71] e MYCIN [Sho76]. O sistema DENDRAL é capaz de inferir a estrutura molecular de compostos desconhecidos a partir de dados espectrais de massa e de resposta magnética nuclear. O sistema MYCIN auxilia médicos na escolha de uma terapia de antibióticos para pacientes com bacteremia, meningite e cistite infecciosa, em ambiente hospitalar.

Desde então, muitos SE's foram desenvolvidos para resolver problemas em muitos domínios diferentes, incluindo: agricultura, química, sistemas de computadores, eletrônica, engenharia, geologia, gerenciamento de informações, direito, matemática, medicina, aplicações militares, física, controle de processos e tecnologia espacial. Alguns destes sistemas, como o sistema XCON/R1 [McD82] para a configuração de computadores VAX da empresa DEC, e PROSPECTOR [HDE78] para a análise das possibilidades de se encontrar tipos específicos de depósitos minerais em uma dada área, revelaram-se altamente lucrativos do ponto de vista comercial.

**Figure 3:** Arquitetura de um SE




Visando a uma maior facilidade de uso, além de eficiência e expressividade, a arquitetura simples dos sistemas de produção de Post foi generalizada. Um SE atual, conforme mostrado na figura 3, apresenta, em geral, uma arquitetura com três módulos: uma *base de regras* , uma *memória de trabalho* e um *motor de inferência* . A base de regras e memória de trabalho formam a chamada *base de conhecimento* do SE, onde está representado o conhecimento sobre o domínio. O motor de inferência é o mecanismo de controle do sistema que avalia e aplica as regras de acordo com as informações da memória de trabalho.[7]

A memória de trabalho, apenas uma seqüência de caracteres no modelo de Post, no modelo generalizado pode conter qualquer tipo de estrutura de dados. Mais do que estruturas de dados, as memórias de trabalho de SE's devem respeitar um *método de representação de conhecimento* (ver capítulo 2), isto é, uma linguagem formal e uma descrição matemática de seu significado. A lógica de primeira ordem é um exemplo típico de formalismo de representação de conhecimento.

A base de regras passa a conter condições que representam ``perguntas'' à representação de conhecimento da memória de trabalho. Estas perguntas, limitadas à comparação de caracteres no modelo de Post, podem ser de diferentes tipos, mas em geral envolvem

variáveis a serem instanciadas e eventualmente algum tipo de inferência. A sintaxe das regras varia de acordo com o sistema e pode ser bastante flexível e próxima da linguagem natural, como nos sistemas ROSIE [FHRSW82] e G2 [Gen92], ou bastante formais, como na família de sistemas OPS [FM77].

Outra característica comum nos SE's atuais é a existência de um mecanismo de raciocínio incerto que permita representar a incerteza a respeito do conhecimento do domínio (ver seção 4).

O motor de inferência controla a atividade do sistema. Esta atividade ocorre em ciclos, cada ciclo consistindo em três fases:

1. Correspondência de dados, onde as regras que satisfazem a descrição da situação atual são selecionadas.
2. Resolução de conflitos, onde as regras que serão realmente executadas são escolhidas dentre as regras que foram selecionadas na primeira fase, e ordenadas.
3. Ação, a execução propriamente dita das regras.

As principais vantagens dos sistemas de produção como método de representação de conhecimento são [Wat86]: modularidade, uniformidade e naturalidade. As principais desvantagens são: ineficiência em tempo de execução e complexidade do fluxo de controle que leva à solução dos problemas. Estas vantagens e desvantagens determinam as características que devem ter os domínios que se adaptam ao desenvolvimento de SE's baseados em sistemas de produção: (i) ser descrito por um conhecimento consistindo em um conjunto muito grande de fatos parcialmente independentes, (ii) dispor de métodos de solução consistindo de ações independentes, e (iii) apresentar uma nítida separação entre conhecimento e ação.

A chave para a desempenho de um SE está no conhecimento armazenado em suas regras e em sua memória de trabalho. Este conhecimento deve ser obtido junto a um especialista

humano do domínio e representado de acordo com regras formais definidas para a codificação de regras no SE em questão. Isto divide um SE em duas partes: a ferramenta de programação que define o formato do conhecimento da memória de trabalho e das regras, além dos aspectos operacionais de sua utilização, e o conhecimento do domínio propriamente dito.

Devido a esta separação, atualmente, os SE's são desenvolvidos em geral a partir de *arcabouços de sistemas especialistas (ASE)* : ferramentas que suportam todas as funcionalidades de um SE [Gev87], restando ao programador apenas codificar o conhecimento especializado de acordo com a linguagem de representação de conhecimento disponível. A existência de ASE's facilitou bastante a implementação de SE's e foi um dos fatores responsáveis por sua disseminação.

**Exemplo:** Diversos exemplos apresentados neste capítulo são baseados no sistema *FASE (Ferramenta para a construção de Arcabouços de Sistemas Especialistas)* [BM93a], [Bit95]. Este sistema foi implementado na linguagem de programação Common Lisp [SJ84]; os programas fonte são de domínio público e podem ser facilmente obtidos via Internet. O sistema consiste em um conjunto de bibliotecas de funções com as seguintes funcionalidades: representação de conhecimento, estratégias de controle, resolução de conflito e tratamento de incerteza. Existe ainda um módulo único de manipulação de bases de regras ao qual podem ser integradas as funcionalidades adequadas, de acordo com as características do problema a ser resolvido. A modularidade do ambiente é decorrência dos conceitos de *padrão* e *instância* . Um padrão (do inglês, ``pattern'') é uma representação genérica de um conjunto de fragmentos de conhecimento, geralmente envolvendo variáveis. Para cada tipo de representação de conhecimento disponível no ambiente, existe um tipo de padrão. Uma instância é uma representação específica de um fragmento de conhecimento. Uma instância contém uma substituição de variáveis, o tempo de criação do fragmento de conhecimento que gerou a instância, um ou mais números representando medidas de incerteza, informação sobre a origem do fragmento de conhecimento e, eventualmente, outros atributos que sejam necessários a representações de conhecimento específicas (por exemplo, definições de domínio de discurso no caso de variáveis nebulosas).

Intuitivamente um padrão pode ser interpretado como uma ``pergunta" à memória de trabalho, e uma instância, como uma ``resposta" a esta pergunta. Além disso, a combinação entre um padrão e uma instância é interpretada como um novo fragmento de conhecimento, onde as variáveis presentes no padrão são substituídas de acordo com a substituição presente na instância.

As regras utilizadas no ambiente têm a seguinte sintaxe:

<rule>::= (<name><alpha>

({<pattern>})  ({<pattern>}))

onde <name> é um símbolo correspondente ao nome da regra e <alpha>

Aquisição de conhecimento

A parte mais sensível no desenvolvimento de um SE é, certamente, a *aquisição de conhecimento* . Esta não pode limitar-se à adição de novos elementos de conhecimento à base de conhecimentos; é necessário integrar o novo conhecimento ao conhecimento já disponível, através da definição de relações entre os elementos que constituem o novo conhecimento e os elementos já armazenados na base. Dois tipos de mecanismos para a definição de tais relações foram propostos: ligar os elementos de conhecimento diretamente através de ponteiros, ou reunir diversos elementos relacionados em grupos (em inglês ``clustering").

Outro ponto importante na aquisição de conhecimento é o tratamento de incoerências. Dependendo da forma como o novo conhecimento é adquirido, pode haver erros de aquisição. Estes erros podem resultar da própria natureza do conhecimento, como em dados obtidos através de sensores sujeitos a ruído, ou podem ser gerados pela interface humana existente entre o mundo real e o sistema de representação. Técnicas foram desenvolvidas para evitar erros de aquisição, como, por exemplo, a especificação de regras de aquisição

em que o tipo de conhecimento esperado é definido. Estas técnicas são comuns aos sistemas de representação de conhecimento e aos sistemas de gerenciamento de bancos de dados. Por outro lado, uma base de conhecimento pode ser examinada periodicamente com a finalidade de detectar incoerências eventualmente introduzidas no processo de aquisição. Este método é limitado pelo fato de que linguagens de representação razoavelmente expressivas não contam com procedimentos completos de verificação conhecidos. Finalmente, deve-se observar que a adequação do formalismo de representação ao tipo de conhecimento do mundo real a ser representado é fundamental para a eficiência do processo de aquisição.

Métodos de representação de conhecimento

A parte mais importante no projeto de um SE é a escolha do método de representação de conhecimento. A linguagem associada ao método escolhido deve ser suficientemente expressiva (mas não *mais* do que o suficiente) para permitir a representação do conhecimento a respeito do domínio escolhido de maneira completa e eficiente. Em tese, uma representação geral como a lógica seria suficientemente expressiva para representar qualquer tipo de conhecimento. No entanto, problemas de eficiência, facilidade de uso e a necessidade de expressar conhecimento incerto e incompleto levaram ao desenvolvimento de diversos tipos de formalismos de representação de conhecimento. A seguir, apresentam-se alguns dos formalismos de representação de conhecimento mais utilizados.

**Lógica**

A *lógica* é a base para a maioria dos formalismos de representação de conhecimento, seja de forma explícita, como nos SE's baseados na linguagem Prolog, seja disfarçada na forma de representações específicas que podem facilmente ser interpretadas como proposições ou predicados lógicos, por exemplo, as listas da forma:

**(<atributo>, <objeto>, <valor>, <coeficiente de certeza>)**

utilizadas como padrão para a representação de conhecimento no sistema MYCIN (ver seção 5.4.1). Mesmo os formalismos não lógicos têm, em geral, seu significado formal descrito através de uma especificação lógica de seu comportamento.

**Exemplo:** O sistema FASE dispõe de um módulo de representação de conhecimento baseado na lógica de primeira ordem. Todos os formalismos implementados no sistema apresentam as mesmas primitivas de acesso: *store*, para armazenar um fragmento de conhecimento na base, *query*, para obter as instâncias que satisfazem às restrições especificadas através das variáveis que ocorrem em seu argumento, e *list*, que permite examinar o conteúdo da base. O acesso a uma base de conhecimento pode ser feito através de um padrão de uma regra ou, diretamente, através de comandos. A linguagem de acesso obedece à seguinte definição:

<command>::=

(*logic-query* <query>) |

(*logic-store* <store>) |

(*logic-list*)

<pattern>::=

(*logic*{<query>}) |

(*logic*{<store>})

<query>::=

<store>::=

(<predicate> {<closed term>}) |

(*off* (<predicate> {<closed term>}))

<closed term>::=

<constant>|

(<function> {<closed term>})

<term>::=

<constant>|

<variable>|

(<function> {<term>})

Por exemplo, uma base de conhecimento pode ser armazenada no sistema FASE, utilizando o formalismo lógico, através da seguinte seqüência de comandos:

> (logic-store '(progenitor boris jane))
t

```
> (logic-store '(progenitor boris marcia))
t
> (logic-store '(progenitor adelia jane))
t
> (logic-store '(progenitor adelia marcia))
t
> (logic-store '(progenitor jane tiago))
t
```

Uma vez armazenadas as informações, a base de conhecimento pode ser consultada através do comando *query*:

```
> (logic-query '(progenitor x tiago))
([ Substituicao  : ((x . jane)) ])
> (logic-query '(progenitor boris y))
([ Substituicao  : ((y . marcia)) ]
 [ Substituicao  : ((y . jane)) ])
```

Em uma regra, as instâncias associadas a padrões diferentes são combinadas, o que permite filtrar as instâncias desejadas, por exemplo, o padrão (logic (progenitor x y)(progenitor y tiago)) teria como resposta a seguinte lista de instâncias:

```
([ Substituicao  : ((x . boris)(y . jane)) ]
 [ Substituicao  : ((x . adelia)(y . jane)) ])
```

O conteúdo total da base de conhecimento pode ser listado através da primitiva *list*:

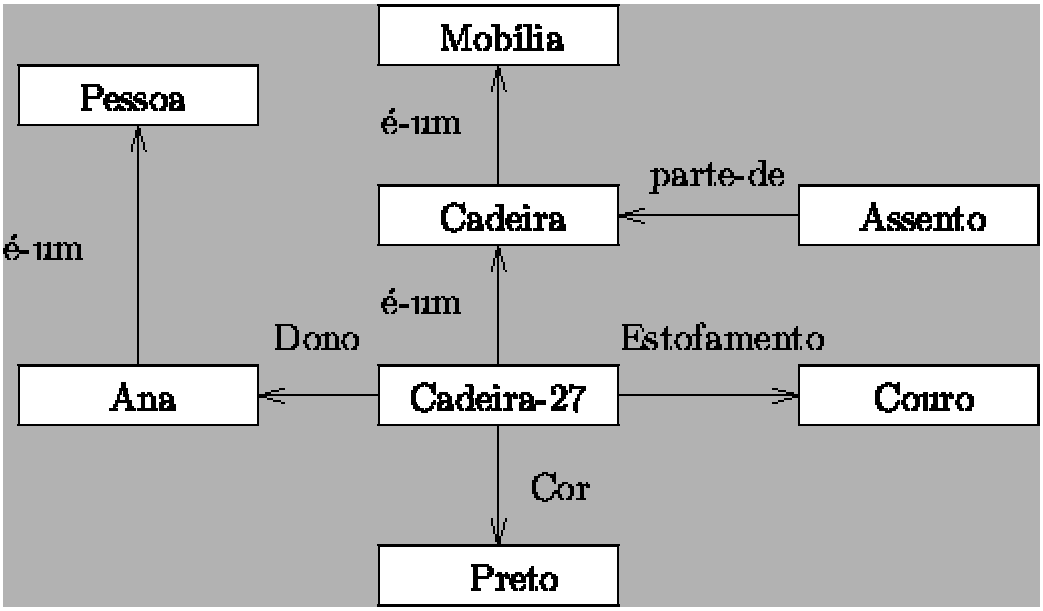
```
> (logic-list)
(progenitor jane tiago)
(progenitor adelia marcia)
(progenitor adelia jane)
(progenitor boris marcia)
(progenitor boris jane)
```

**Redes semânticas**

*Rede semântica* é um nome utilizado para definir um conjunto heterogêneo de sistemas. Em última análise, a única característica comum a todos estes sistemas é a notação utilizada: uma rede semântica consiste em um conjunto de *nodos* conectados por um conjunto de *arcos* . Os nodos em geral representam objetos e os arcos, relações binárias entre esses objetos. Mas os nodos podem também ser utilizados para representar predicados, classes, palavras de uma linguagem, entre outras possíveis interpretações, dependendo do sistema de redes semânticas em questão.

A utilização do formalismo de nodos e arcos para a representação de conhecimento foi proposta por Quillian [Qui68]. No seu artigo, Quillian propõe um modelo computacional da memória humana chamado *memória semântica* . Este modelo, onde conceitos são representados por nodos, e relações entre conceitos, por arcos, explica diversos resultados experimentais sobre o comportamento da memória humana, como, por exemplo, o fato de que o reconhecimento de objetos que pertencem a classes mais numerosas toma mais tempo do que o reconhecimento dos pertencentes a classes menos numerosas. Muitas características dos sistemas de redes semânticas desenvolvidos ulteriormente já estavam presentes na proposta de Quillian.

**Figure:** Rede semântica




**Exemplo:** Considere a rede semântica da figura 4, que representa conceitos relacionados com mobiliário [Ric83]. Os arcos *é-um* e *é-parte* são bastante comuns em sistemas de redes semânticas. Este tipo de arco é utilizado para determinar a herança de propriedades. Os demais arcos (*dono, cor, estofamento*) são específicos do domínio e representam propriedades de conceitos. Esses arcos são chamados *traços*.

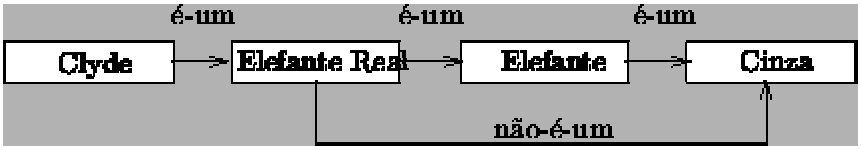
Durante os anos setenta, o formalismo de redes semânticas foi utilizado na implementação de diversos sistemas para a compreensão de linguagem natural. O sistema HAM [AB73] e o sistema Active Structural Network [RN75] são dois exemplos deste tipo de sistemas.

Dois artigos publicados em 1975 tiveram uma grande influência na pesquisa relacionada às redes semânticas: o artigo de Minsky [Min75], propondo o formalismo de quadros, e o artigo de Woods [Woo75], analisando o significado dos arcos nas redes semânticas. O artigo de Minsky introduziu a noção de nodos com estrutura interna, criando uma nova forma de representação de conhecimento, derivada das redes semânticas, chamada quadros. O artigo de Woods chamou a atenção para a necessidade de uma semântica formal que fundamentasse os sistemas baseados em redes semânticas. Este artigo foi seguido de uma série de outros, onde as redes semânticas eram associadas ao formalismo lógico. Alguns destes artigos utilizavam as redes semânticas apenas como uma notação sintática alternativa

para fórmulas lógicas; outros apresentavam as redes semânticas como um método independente de representação de conhecimento, utilizando o formalismo lógico apenas como ferramenta para a definição de uma semântica para nodos e arcos.

A *herança de propriedades* através de caminhos formados por arcos é uma das características mais importantes do método de representação por redes semânticas. Esta característica permite que propriedades de um nodo sejam especificadas apenas uma vez, no mais alto nível de uma hierarquia de conceitos, sendo herdadas por todos os conceitos derivados, implicando uma economia substancial de memória. Os algoritmos de herança utilizados em redes semânticas na forma de árvores são bastante simples e muito eficientes, mas se heranças múltiplas forem permitidas, especialmente na presença de arcos que definam exceções, o problema da determinação de caminhos de herança se torna bastante complexo. Mais grave ainda é o fato de que na presença de herança múltipla e exceções, a intuição sobre o que é uma política de herança coerente passa a ser discutível, levando diferentes sistemas a implementarem diferentes políticas de herança [THT87].

**Figure:** O problema das exceções em redes semânticas




**Exemplo:** O problema da ``cor de Clyde'' é um exemplo clássico dos problemas decorrentes do uso de múltipla herança e arcos de exceção. De acordo com a rede semântica da figura 5, Clyde deve ser considerado Cinza por ser um Elefante e não Cinza por ser um Elefante Real. A estratégia de herança, usualmente aceita, determina que propriedades ligadas a conceitos mais específicos devem ter prioridade, ou seja, no caso Clyde deve ser considerado não Cinza, pois Elefante Real é mais específico que Elefante. No entanto, exemplos mais complexos podem não ser tão intuitivos.

Além da herança de propriedades, um outro mecanismo de inferência utilizado em redes semânticas é a correspondência de um fragmento de rede em relação a uma rede dada. A especificação da semântica deste mecanismo é ainda mais complexa que a do mecanismo de herança, por depender da escolha do significado dos arcos da rede. Estes arcos receberam uma variedade de significados; uma revisão das diferentes interpretações pode ser encontrada em [BFL83].

**Exemplo:** Analogamente à representação lógica, o formalismo de redes semânticas conta com comandos de armazenamento e consulta, e pode ser acessado a partir das regras através de padrões. A especificação da linguagem de acesso é a seguinte:

<command>::=

(*snet-query* <query>) |

(*snet-store* <store>) |

(*snet-list*)

<pattern>::=

(*snet*{<query>}) | (*snet*{<store>})

<store>::=

(<node>) |

(<node><node> [<edge>]) |

(<node><node> [(*not*. <edge>)]) |

(*off* (<node>)) |

(*off* (<node><node> [<edge>])) |

(*off* (<node><node> [(*not*. <edge>)]))

<query>::=

(<snet> (<node><node> [<edge>]))

<snet>::=

<variable>|<symbol>

<node>::=

<variable>|<symbol>

<edge>::=

<variable>|<symbol>| (*not* . <variable>) | (*not* . <symbol>)

A rede semântica associada ao exemplo da cor de Clyde pode ser definida através do seguinte comando:

> (snet-store '(color (clyde real e-um)
(real elefante e-um)

```
                (elefante cinza e-um)
                (clyde elefante e-um)
                (real cinza (not . e-um))))
t
```

Observe como foi introduzido o arco negado, com a construção de um ``dotted-pair'' onde o primeiro elemento é o símbolo ``not''. A visualização do conteúdo da base pode ser feita através do comando *list*:

```
> (snet-list)
==================
graph : color
------------------
node : clyde
  | * edge : e-um
  |---|> node : elefante
  |   | * edge : e-um
  |   |---|> node : cinza
  | * edge : e-um
  |---|> node : real
      | * edge : (not . e-um)
      |---|> node : cinza
      | * edge : e-um
      |---|> node : elefante
```

Pode-se agora determinar toda a teoria hierárquica definida através da rede ``color'', utilizando as expressões de consulta adequadas:

```
> (snet-query '(color (x y e-um)))
([ Substituicao  : ((x . elefante) (y . cinza)) ]
 [ Substituicao  : ((x . real) (y . elefante)) ]
 [ Substituicao  : ((x . clyde) (y . elefante)) ]
 [ Substituicao  : ((x . clyde) (y . real)) ])
```

> (snet-query '(color (x y (not . e-um))))
([ Substituicao : ((x . real) (y . cinza)) ]
[ Substituicao : ((x . clyde) (y . cinza)) ])

**Quadros**

Os *quadros* (do inglês, ``frames"), e sua variação, os *roteiros* (do inglês, ``scripts"), foram introduzidos para permitir a expressão das estruturas internas dos objetos, mantendo a possibilidade de representar herança de propriedades como as redes semânticas.

As idéias fundamentais destes métodos foram introduzidas por Marvin Minsky [Min75] em seu artigo *A framework to represent knowledge*. As aplicações propostas por Minsky para o novo método foram análise de cenas, modelagem da percepção visual e compreensão de linguagem natural. No entanto, o artigo não propõe metodologia de implementação, nem definição formal do método. Desde 1975, diversos sistemas foram implementados baseados na idéia de quadros, e diversas definições formais foram propostas. O método de quadros também está na origem das idéias que levaram às linguagens de programação orientadas a objetos. Os roteiros foram propostos por Schank e Abelson [SA75], [SA77], e consistem em sistemas de quadros especializados na descrição de seqüências de eventos.

**Figure 6:** Quadros

| Frame : Cômodo | Super-Frame : Lugar-coberto | | |
|---|---|---|---|
| Atributos | Default | Tipo | Se-necessário |
| Número de paredes | 4 | número | |
| Formato | retangular | símbolo | |
| Altura | 3 | número ($m$ ) | |
| Área | | número ($m^2$ ) | |
| Volume | | número ($m^3$ ) | (Área*Altura) |

é-um

| Frame : Sala | Super-Frame : Comodo | |
|---|---|---|
| Atributos | Default | Tipo |
| Mobiliário | (sofa, mesa, cadeiras) | lista de símbolos |
| Finalidade | convivência | símbolo |

Em geral, um quadro consiste em um conjunto de *atributos* que, através de seus valores, descrevem as características do objeto representado pelo quadro. Os valores atribuídos a estes atributos podem ser outros quadros, criando uma rede de dependências entre os quadros. Os quadros são também organizados em uma hierarquia de especialização, criando uma outra dimensão de dependência entre eles. Os atributos também apresentam propriedades, que dizem respeito ao tipo de valores e às restrições de número que podem ser associados a cada atributo. Essas propriedades são chamadas *facetas* .

**Exemplo:** Quadros descrevendo um comodo e uma sala são mostrados na figura 6. Por esta descrição, pode-se concluir que uma sala é um tipo de cômodo, normalmente com quatro paredes e de formato retangular, com um mobiliário específico. As facetas dos atributos especificam os tipos de valores esperados e, se for o caso, procedimentos adequados para calcular o valor do atributo.

Da mesma maneira que as redes semânticas, os sistemas baseados no método de quadros não são um conjunto homogêneo; no entanto, algumas idéias fundamentais são compartilhadas por estes sistemas. Uma dessas idéias é o conceito de herança de propriedades, o que permite a especificação de propriedades de uma classe de objetos através da declaração de que esta classe é uma subclasse de outra que goza da propriedade em questão. A herança pode ser um mecanismo de inferência muito eficiente em domínios que apresentem uma taxonomia natural de conceitos, como a biologia ou a paleontologia.

Outra idéia comum aos sistemas baseados em quadros é o *raciocínio guiado por expectativas* . Um quadro contém atributos, e estes atributos podem ter valores típicos ou, ``a priori", os chamados *valores de exceção* (do inglês, ``default values"). Ao tentar instanciar um quadro para que corresponda a uma situação dada, o processo de raciocínio deve tentar preencher os valores dos atributos do quadro com as informações disponíveis na descrição da situação. Para o processo de raciocínio, saber o que procurar para completar a informação necessária (e caso esta não esteja disponível, que valores tentativos atribuir aos atributos não preenchidos) pode ser um fator fundamental para a eficiência do processo de reconhecimento de uma situação complexa.

Uma terceira idéia é a *ligação procedimental* . Além dos valores por ``default", um atributo pode ser associado a um procedimento que deve ser executado quando certas condições forem satisfeitas, por exemplo: ao ser criado o atributo, ao ser lido o valor do atributo, ao ser modificado o valor do atributo, ou ao ser destruído o valor do atributo.

**Exemplo:** A linguagem de acesso do formalismo de quadros é definida pela seguinte gramática:

<command>::=

(*frame-query* <query>) |

(*frame-store* <store>) |

(*frame-list*)

<pattern>::=

(*frame*{<query>}) |

(*frame*{<store>})

<query>::=

(<frame> [<antecedent>]{<slot query>}))

<frame>::=

<variable>|<symbol>

<antecedent>::=

<variable>|<symbol>

<slot query>::=

(<slot><value>)

<slot>::=

<variable>|<symbol>

<value>::=

qualquer S-expressão

<store>::=

(<frame><antecedent> {<slot definition>}) |

(*off* (<frame>)) |

(*off* (<frame> (<slot>)$^{+}$))

<slot definition>::=

(<slot><value>) |

(<slot><function definition><demon type>) |

<demon type>::=

*if-create* |

*if-modify* |

*if-necessary* |

*if-delete*

O tipo de *saci* (do inglês, ``demon'') indica em que ocasião o procedimento associado ao *atributo* (do inglês, ``slot'') do quadro deverá ser executado: (i) *if-modify*, ao ser alterado o valor do atributo, (ii) *if-necessary*, ao ser consultado o valor do atributo, (iii) *if-delete*, ao ser eliminado o atributo. Todos os sacis devem ser funções Lisp com os dois primeiros

parâmetros associados ao nome do quadro e nome do atributo. No caso de um saci do tipo *if-modify*, um terceiro parâmetro é associado ao novo valor do atributo.

O exemplo de quadro associado a uma sala pode ser armazenado no sistema FASE através dos seguintes comandos:

```
> (frame-store '(lugar-coberto root))
t
> (frame-store '(comodo lugar-coberto
                    (numero-de-paredes 4)
                    (formato retangular)
                    (altura 3)
                    (area nil)
                    (volume (lambda (f s)
                            (* (get-slot f (quote altura))
                               (get-slot f (quote area))))
                          if-necessary)))
t
> (frame-store '(sala comodo
                  (mobiliario (sofa mesa cadeiras))
                  (finalidade convivencia)))
t
```

Observe que o quadro ``root'' é pré-existente e não precisa ser definido. O saci do tipo *if-necessary* foi definido utilizando uma forma *lambda*. Alternativamente, poderíamos ter definido uma função externa e colocado seu nome na base de conhecimento. O saci será executado caso o valor do atributo ``volume'' seja consultado. Para que a execução ocorra sem erros, é necessário antes atribuir um valor para o atributo ``area''. Na função saci foi utilizada a primitiva ``get-slot'' que permite acessar diretamente o valor de um atributo, sem utilizar a função ``frame-query''. Os comandos abaixo exemplificam a utilização da função saci e da função *list*:

```
> (frame-store '(comodo (area 10)))
```

```
t
> (frame-query '(comodo (volume x)))
([ Substituicao  : ((x . 30)) ])
> (frame-list)

root <== root

   lugar-coberto <== root

      comodo <== lugar-coberto
        * volume = 30
        * se-necessario: (lambda (f s)
                          (* (get-slot f (quote altura))
                             (get-slot f (quote area))))
        * area = 10
        * altura = 3
        * formato = retangular
        * numero-de-paredes = 4

          sala <== comodo
            * finalidade = convivencia
            * mobiliario = (sofa mesa cadeiras)
```



Para descrever as principais características encontradas nos ASE's disponíveis atualmente, vamos adotar as dimensões de análise propostas em um artigo de Stylianou et al. [SMS92]. Neste artigo, é descrita uma pesquisa[8] desenvolvida pelos autores visando detectar as características técnicas mais importantes dos ASE's, segundo os usuários e construtores. As dimensões escolhidas foram as seguintes: interface com o usuário, interface de desenvolvimento, interface com o sistema operacional, motor de inferência e métodos de representação de conhecimento. Os aspectos comentados abaixo foram considerados na

pesquisa como importantes. Para cada dimensão da análise são ressaltados os aspectos considerados críticos.

**Interface com o usuário**

Em qualquer tipo de ``software", a interface com o usuário final é fundamental para o seu sucesso. Em particular, um SE, além de apresentar uma interface ergonomicamente bem projetada, deve ainda levar em conta o grau de familiarização do usuário com o domínio de trabalho do sistema e com os computadores em geral.

Algumas técnicas tornam a interface com o usuário mais amigável, por exemplo, o uso de janelas, menus, gráficos de alta resolução, animação, cores, etc. De todo modo, as telas a serem apresentadas ao usuário devem ser de fácil compreensão, e as explicações necessárias devem ser claras e diretas. Mais algumas características interessantes para interfaces com o usuário são: (i) disponibilidade de diversos tipos de interfaces, adaptadas ao tipo de usuário (iniciante, especialista); (ii) possibilidade de interromper a execução do sistema em um determinado ponto e poder retomá-la sem necessidade de reprocessamento; (iii) mensagens de erro claras e informativas; (iv) possibilidade de alterar certas entradas ao sistema e comparar os resultados obtidos; (v) capacidade de capturar e armazenar telas de execução. Os aspectos considerados críticos foram: facilidade para explicação e documentação.

**Interface de desenvolvimento**

A interface a ser utilizada pela equipe de desenvolvimento de um SE representa, em geral, a maior parte do código de um ASE, além de ser determinante para o bom andamento dos projetos desenvolvidos com o arcabouço em questão.

A possibilidade de customizar o ambiente pela introdução de novas capacidades, ou pela modificação de capacidades já existentes, através da escrita de código na linguagem de implementação do ASE, é uma característica bastante útil. Caso os programas fonte do arcabouço estejam disponíveis, isto é facilitado.

Correção de erros, manutenção de uma lista de comandos da seção, índice cruzado de símbolos e regras, visualização gráfica dos encadeamentos de regras utilizadas na solução, editor de regras, facilidades de explicação e uma boa documentação são outras características positivas para a interface de desenvolvimento.

Aspectos considerados críticos: facilidade para explicação, documentação, facilidade para customizar explicações e prototipagem rápida.

**Interface com o sistema operacional**

Existem atualmente SE's rodando em plataformas que variam de microcomputadores do tipo PC até ``mainframes", passando por estações de trabalho e máquinas especializadas em processamento simbólico, as chamadas ``Lisp Machines". No entanto, setenta por cento dos SE's pesquisados por Stylianou et al. [SMS92] foram desenvolvidos para microcomputadores. Diversas ferramentas disponíveis no mercado rodam ainda em várias plataformas diferentes.

Quanto à linguagem de programação, existem ASE's escritos em linguagem de montagem (do inglês, ``assembler"), C, Fortran, Basic, Forth, Pascal e Cobol, além, é claro, daqueles escritos em linguagens para IA como Lisp, Prolog e Smalltalk. Diversos ASE's originalmente escritos em Lisp foram traduzidos para linguagens como C ou C++ para aumentar sua eficiência, portabilidade e compatibilidade com outros ``softwares".

A facilidade de comunicação com aplicações convencionais (como bancos de dados, planilhas e sistemas de rede) é considerada uma importante característica em um ASE. O fato dessa facilidade ter sido relegada a um segundo plano nos primeiros ASE's representou uma grande dificuldade para sua disseminação em ambiente comercial.

Uma última característica ligada ao sistema operacional é o grau de segurança oferecido por um ASE. É possível que um SE envolva informações confidenciais ou únicas que não devem ser acessadas e modificadas por qualquer usuário.

Aspectos considerados críticos: facilidade para integração com sistemas já existentes.

**Motor de inferência**

As principais características do motor de inferência disponível em um ASE dizem respeito às seguintes funcionalidades: método de raciocínio, estratégia de busca, resolução de conflito e representação de incerteza. Dentre todos os aspectos discutidos nesta seção, foi considerada crítica a existência de raciocínio do tipo encadeamento regressivo.

**Modo de raciocínio**

Existem basicamente dois modos de raciocínio aplicáveis a regras de produção: *encadeamento progressivo* ou *encadeamento a frente* (do inglês, ``forward chaining"), e *encadeamento regressivo* ou *encadeamento para trás* (do inglês, ``backward chaining"). No encadeamento progressivo, também chamado encadeamento dirigido por dados, a parte esquerda da regra é comparada com a descrição da situação atual, contida na memória de trabalho. As regras que satisfazem a esta descrição têm sua parte direita executada, o que, em geral, significa a introdução de novos fatos na memória de trabalho.

No encadeamento regressivo, também chamado encadeamento dirigido por objetivos, o comportamento do sistema é controlado por uma lista de objetivos. Um objetivo pode ser satisfeito diretamente por um elemento da memória de trabalho, ou podem existir regras que permitam inferir algum dos objetivos correntes, isto é, que contenham uma descrição deste objetivo em suas partes direitas. As regras que satisfazem esta condição têm as instâncias correspondentes às suas partes esquerdas adicionadas à lista de objetivos correntes. Caso uma dessas regras tenha todas as suas condições satisfeitas diretamente pela memória de trabalho, o objetivo em sua parte direita é também adicionado à memória de trabalho. Um objetivo que não possa ser satisfeito diretamente pela memória de trabalho, nem inferido através de uma regra, é abandonado. Quando o objetivo inicial é satisfeito, ou não há mais objetivos, o processamento termina.

O tipo de encadeamento normalmente é definido de acordo com o tipo de problema a ser resolvido. Problemas de *planejamento*, *projeto* e *classificação* tipicamente utilizam encadeamento progressivo, enquanto problemas de *diagnóstico*, onde existem apenas

algumas saídas possíveis mas um grande número de estados iniciais, utilizam encadeamento regressivo.

Em geral, os ASE's adotam apenas um modo de raciocínio; no entanto, existem alguns que permitem ambos os modos, mas de maneira independente, e ainda outros que permitem um *encadeamento misto* , onde os encadeamentos progressivo e regressivo se alternam de acordo com o desenvolvimento da solução do problema e com a disponibilidade de dados.

Uma característica importante do modo de raciocínio se refere à monotonicidade ou não do método de inferência. Sistemas monotônicos não permitem a revisão de fatos, isto é, uma vez um fato declarado verdadeiro, ele não pode mais tornar-se falso. Sistemas não monotônicos, por outro lado, permitem a alteração dinâmica dos fatos. O preço desta capacidade é a necessidade de um mecanismo de revisão de crenças, pois uma vez que um fato, antes verdadeiro, torna-se falso, todas as conclusões baseadas neste fato também devem tornar-se falsas. Uma solução para este problema é apresentada em [Doy79].

**Estratégia de busca**

Uma vez definido o tipo de encadeamento, o motor de inferência necessita ainda de uma estratégia de busca para guiar a pesquisa na memória de trabalho e na base de regras. Este tipo de problema é conhecido como *busca em espaço de estados* . Este tópico foi um dos primeiros estudados em IA, no contexto de solução de problemas (do tipo quebra-cabeças) e jogos por computador (damas, xadrez, go, etc.). Descrições detalhadas dos algoritmos de busca podem ser encontradas na maioria dos livros texto de IA, por exemplo, Charniak e McDermott [CM85], Nilsson [Nil71], Rich [Ric83], Winston [Win84] entre outros.

**Resolução de conflito**

Ao terminar o processo de busca, o motor de inferência dispõe de um conjunto de regras que satisfazem à situação atual do problema, o chamado *conjunto de conflito* . Se esse conjunto for vazio, a execução é terminada; caso contrário, é necessário escolher que regras serão realmente executadas e em que ordem. Os métodos de resolução de conflito mais utilizados ordenam as regras de acordo com os seguintes critérios [MF78]: prioridades

atribuídas estaticamente; características da estrutura das regras como complexidade, simplicidade e especificidade; características dos dados associados às regras como o tempo decorrido desde sua obtenção, sua confiabilidade ou seu grau de importância; e, finalmente, seleção ao acaso.

Em geral, a utilização de um desses critérios é insuficiente para resolver os conflitos. Neste caso, o ASE pode combinar mais de um método na forma de método primário, secundário, etc. Os melhores ASE's dispõem de diversos métodos de resolução de conflito e permitem ao usuário a especificação de quais métodos utilizar e em que ordem.

**Representação de incerteza**

O tratamento de incerteza é uma ativa área de pesquisa em SE's, pois os domínios adequados à implementação de SE's se caracterizam exatamente por não serem modelados por nenhuma teoria geral, o que implica descrições incompletas, inexatas ou incertas. Diversos métodos foram propostos para tratar este problema, por exemplo, método Bayesiano, fatores de certeza (conforme o modelo adotado no MYCIN), teoria de Dempster-Shafer, teoria dos conjuntos nebulosos, teoria de probabilidades subjetivas e teoria de possibilidades (ver seção 4).

De maneira geral, estes métodos atribuem aos fatos e regras uma medida numérica que represente de alguma forma a ``confiança" do especialista. Os métodos utilizados não são necessariamente coerentes uns com os outros e cada método adapta-se melhor a determinados tipos de problemas. Diversos ASE's dispõem de mais de um método de tratamento de incerteza, deixando ao usuário a escolha do mais adequado ao seu problema. Uma característica freqüente desses métodos é a existência de um limite mínimo para a medida de incerteza, abaixo do qual o fato ou regra é desconsiderado. Este limite pode, em geral, ser fixado pelo usuário.

**Representação de conhecimento**

Em geral, os ASE's se limitam a oferecer um único tipo de representação de conhecimento (ver seção 5.2). Alguns sistemas dispõem de mais de um formalismo, os quais, no entanto,

devem ser utilizados de maneira isolada. Alguns poucos ASE's possuem os chamados *sistemas híbridos de representação de conhecimento* [NL87] que, além de possuir diversos formalismos de representação, dispõem também de algoritmos de acesso que integram os conhecimentos representados nos diversos formalismos para permitir sua utilização de maneira integrada.

Exemplos de sistemas especialistas

Nesta seção, são descritos alguns SE's mais conhecidos [BF81].

**Mycin**

O sistema MYCIN [Sho76] foi um dos primeiros SE's. Seu objetivo é prover conselho a respeito de diagnóstico e terapia de doenças infecciosas. Este tipo de aconselhamento pode ser muito útil, pois nem sempre o médico responsável é um especialista em infecções, principalmente em ambiente hospitalar.

Uma seção do sistema inicia-se com um questionário, a ser respondido pelo usuário, a respeito do paciente. Informações como nome, idade, sexo, tempo de manifestação dos sintomas, resultados de exames, etc. são solicitadas. A partir dessas informações, e utilizando sua base de regras, o sistema é capaz de estabelecer um diagnóstico e propor uma terapia adequada.

A base de regras do sistema contém 450 regras, que lhe permitem diagnosticar e prescrever tratamentos para bacteremia (infecção no sangue), meningite e cistite infecciosa. As regras são representadas internamente na forma de uma lista (em Lisp) do tipo:

```
((and (same cntxt infect primary-bacteremia)
      (membf cntxt site sterilesites)
      (same cntxt portal gi))
 (conclude cntxt ident bacteroides tally 0.7))
```

Esta representação é automaticamente traduzida pelo sistema para uma forma mais legível. O resultado desta tradução, que pode ser utilizado para explicar o raciocínio do sistema, é o seguinte:

***If***

---

1.
the infection is primary-bacteremia, and
2.
the site of the culture is one of the sterile sites, and
3.
the suspected portal of entry of the organism is gastrointestinal tract,

---

***then*** *there is suggestive evidence (0.7) that the identity of the organism is bacteroides.*

As premissas das regras são constituídas por combinações booleanas de *cláusulas*. Cada cláusula é composta por um predicado e uma tripla de parâmetros, com a seguinte interpretação:

(<predicado> <objeto> <atributo> <valor>)

Os predicados são independentes do domínio. Existem 24 predicados pré-definidos, por exemplo, SAME, KNOWN, DEFINITE, etc. Os objetos, atributos e valores dependem do domínio de aplicação. Alguns exemplos de triplas (<objeto> <atributo> <valor>)

utilizados no sistema são: (ORGANISM IDENTITY E-COLI), (CULTURE SITE BLOOD).

A parte de ação da regra contém um ou mais fatos a serem concluídos.

Os elementos da memória de trabalho, que contêm fatos médicos referentes ao paciente e à sua doença, são representados na forma de listas de 4 elementos:

(<atributo> <objeto> <valor> <coeficiente de certeza>)

Por exemplo:

(IDENT ORGANISM-2 KLEBSIELLA 0.25)

Cada regra e cada elemento da memória de trabalho são associados a um coeficiente de certeza. Os coeficientes de certeza no MYCIN variam entre -1 (totalmente falso) e 1 (totalmente verdadeiro). Estes coeficientes são utilizados para propagar a incerteza inicial de uma informação através da cadeia de inferências. O método para realizar esta propagação foi desenvolvido de maneira ``ad hoc'', mas provou-se eficiente para a aplicação desenvolvida. Caso o coeficiente de certeza fique entre -0.2 e 0.2, o fato ou conclusão da regra é considerado desconhecido.

O sistema MYCIN utiliza o encadeamento regressivo associado a uma busca em profundidade. A busca realizada é completa, no sentido em que, dado um objetivo, todas as evidências a favor e contra o objetivo são pesquisadas. A aquisição de conhecimento para o sistema MYCIN é facilitada pela possibilidade do sistema explicar seu raciocínio. Essa facilidade é explorada no sistema TEIRESIAS [Dav80], que permite modificar interativamente a base de regras do MYCIN.

**Dendral**

DENDRAL [FBL71] é o nome de um projeto desenvolvido a partir de 1965 na Universidade de Standford (USA). O objetivo do projeto é desenvolver programas capazes de determinar automaticamente o conjunto de estruturas moleculares, constituídas de

átomos conhecidos, capazes de explicar dados provenientes da análise espectrográfica de uma molécula desconhecida. Uma abordagem algorítmica havia sido tentada anteriormente e se mostrou impraticável devido ao enorme número de estruturas possíveis. O primeiro programa do projeto, o Heuristic DENDRAL, fazendo uso de regras obtidas junto a especialistas humanos em espectroscopia que impõem restrições aos tipos possíveis de estrutura, foi capaz de reduzir o espaço de busca para um tamanho tratável, mantendo os mesmo resultados. Outros programas desenvolvidos no âmbito do projeto são: o sistema Meta-DENDRAL, capaz de inferir automaticamente regras de espectroscopia de massa a partir de exemplos de moléculas devidamente analisadas por seres humanos e o sistema CONGEN, um gerador de estruturas moleculares não relacionado diretamente com técnicas de IA.

O sistema DENDRAL não segue a estrutura tradicional dos SE's, pois ele integra três programas independentes, dos quais apenas dois são baseados em regras. Os três programas têm as seguintes funções: (i) **Planejamento:** determinação das combinações de átomos consistentes com um conjunto de regras heurísticas sobre espectroscopia de massa. As restrições impostas às estruturas são de dois tipos: determinação de fragmentos moleculares que devem estar necessariamente presentes ou ausentes da estrutura final. (ii) **Geração:** construção de todas as estruturas moleculares que obedecem às restrições inferidas na parte de planejamento, isto é, que incluem todos os fragmentos necessários e nenhum dos proibidos. Este processo é realizado por um algoritmo tradicional. Originalmente foi utilizado um algoritmo devido a Lederberg e mais tarde o programa CONGEN. (iii) **Teste:** classificação das estruturas geradas através da simulação de seu comportamento em um espectrógrafo de massa. As estruturas cujos espectros simulados se aproximam do espectro real são classificadas com um escore mais alto. A simulação é realizada utilizando-se regras que prevêem a posição de picos no espectro a partir da estrutura molecular.

Desta forma, o sistema DENDRAL consiste em dois SE's e um programa tradicional. As regras utilizadas na parte de planejamento têm a seguinte forma:

> ***If*** *the spectrum for the molecule has two peaks at masses x1 and x2, such that:*

---

a. x1 + x2 = M + 28, and
b. x1 - 28 is a high peak, and
c. x2 - 28 is a high peak, and
d. at least one of x1 or x2 is high,

---

***then*** *the molecule contains a ketone group.*

Esta regra, caso suas condições sejam satisfeitas, restringe as possíveis estruturas apenas àquelas que contêm um grupo ketona. Ambos os grupos de regras do sistema são utilizados segundo o modo de encadeamento para frente, como é típico de problemas dirigidos por dados.

O sistema DENDRAL, a partir de 1968 até o presente, foi utilizado em diversas pesquisas sobre química orgânica. Alguns resultados de análises realizadas pelo sistemas foram considerados melhores do que os obtidos por especialistas humanos e publicados em revistas especializadas.

**Prospector**

O sistema PROSPECTOR [HDE78] foi desenvolvido no SRI International (USA) com o objetivo de auxiliar geologistas envolvidos em prospecção mineral.

A principal função do sistema é determinar a correspondência entre dados que descrevem uma determinada situação com *modelos* que descrevem classes disjuntas de situações

possíveis. Os modelos são descrições formais dos tipos mais importantes de depósitos minerais e os dados de entrada se referem a observações geológicas de superfície.

O conhecimento geológico é armazenado na forma de uma *rede de inferência* : uma rede semântica onde os nodos representam fatos sobre o domínio, por exemplo, ``there is pervasively biotized hornblende", e os arcos indicam como a probabilidade associada a um nodo influencia um outro nodo. Um fato pode ser interpretado como um indício da veracidade ou da falsidade de um outro. Existem ainda fatos que são independentes, isto é, não existem arcos entre eles. Apesar da representação em forma de grafo e do uso de probabilidades, os arcos de inferência da rede podem ser interpretados como um conjunto de regras. Além dos arcos de inferência, o sistema dispõe de arcos de contexto que indicam que um determinado fato é um contexto necessário a outro fato.

O sistema contém cinco modelos de tipos de depósitos minerais desenvolvidos com a ajuda de cinco geólogos diferentes. Os modelos são representados na forma de redes de inferência. O quadro 4 dá uma idéia da complexidade dos modelos.

**Table 4:** Tamanho dos modelos

| Modelo | N º de nodos | N º de arcos | | |
|---|---|---|---|---|
| Koroko-type massive sulfide | 39 | 34 | | |
| Mississippi-Valey-type lead/zinc | 28 | 20 | | |
| Type A porphyry copper | 187 | 91 | | |
| Komatiitic nickel sulfide | 75 | 49 | | |
| Roll-front sandstone uranium | 212 | 133 | | |

| | | | | |
|---|---|---|---|---|
| Total | 541 | 327 | | |

Além da rede de inferência, o sistema conta com uma rede taxonômica que descreve relações do tipo classe/elemento e classe/subclasse entre os termos mencionados na descrição dos modelos, por exemplo, biotide *é um tipo de* mica.

O funcionamento do sistema é dividido em duas partes. Inicialmente o usuário fornece as informações disponíveis sobre o local de prospecção. Estas informações são representadas na forma de uma rede semântica que é comparada com os modelos disponíveis. O resultado desta primeira parte é a escolha do modelo mais adequado. Na segunda parte, o sistema utiliza o modelo escolhido e as informações disponíveis, através de um mecanismo de encadeamento regressivo, para refinar a análise. Alguns fatos são ditos ``perguntáveis" e durante o refinamento da análise o sistema solicitará ao usuário as probabilidades associadas aos fatos deste tipo, que sejam relevantes para a situação analisada.

**Bibliography**


**AB73**

J. Anderson and G. Bower.

*Human Associative Memory*.

Winston, Washington, D.C., 1973.

**Bal87**

J.F. Baldwin.

Evidential support logic programming.

*Fuzzy Sets and Systems*, 24(1):1-26, 1987.

**Bar75**

A.G. Barto.

*Cellular Automata as Models of Natural Systems*.

The University of Michigan, 1975.

**Bel77**

N.D. Belnap.

A useful four-valued logic.

In J.M. Dunn and G. Epstein, editors, *Modern Uses of Multiple-Valued Logics*. D. Reidel Pub. Co., 1977.

**BF81**

A. Barr and E.A. Feigenbaum, editors.
*The Handbook of Artificial Intelligence*, volume I-II.
William Kaufmann Inc., Los Altos, California, 1981.

**BFL83**

R.J. Brachman, R.E. Fikes, and H.J. Levesque.
Krypton: A functional approach to knowledge representation.
*IEEE Computer (Special Issue on Knowledge Representation)*, 16(10):67-73, October 1983.

**Bit95**

G. Bittencourt.
Um ambiente para ensino e desenvolvimento de sistemas especialistas.
In *III Workshop sobre Educação em Informática/IV Congresso Ibero-Americano de Educação Superior em Computação*, 1995.
29 de julho a 4 de agosto, Canela, RS.

**BM93a**

G. Bittencourt and M. Marengoni.
A customizable tool for the generation of production-based systems.
In G. Rzevski, J. Pastor, and R.A. Adey, editors, *Eighth International Conference on Applications of Artificial Intelligence in Engineering (AIENG'93)*, pages 337-352. Elsevier Applied Science, 1993.

**BM93b**

B. Bouchon-Meunier.
*La Logique Floue*.
Presses Universitaires de France, 1993.
Collection Que sais-je?

**BS84**

B.G. Buchanan and E.H. Shortliffe.
*Rule-Based Expert Systems, the MYCIN Experiments of the Stanford Heuristics*

*Programming Project*.

Addison Wesley Publishing Company, Reading, MA, 1984.

**BS93**

T. Bäck and H.-P. Schwefel.

An overview of evolutionary algorithms for parameter otimization.

*Evolutionary Computation*, 1(1):1-23, 1993.

**BW77**

D.G. Bobrow and T. Winograd.

An overview of KRL, a knowledge representation language.

*Cognitive Science*, 1(1):3-46, 1977.

**CM85**

E. Charniak and D. McDermott.

*Introduction to Artificial Intelligence*.

Addison-Wesley Publishing Company, Reading, MA, 1985.

**Coo71**

S.A. Cook.

The complexity of theorem-proving procedures.

In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, ACM, New York*, pages 151-158, 1971.

**Dav80**

R. Davis.

Meta-rules: Reasoning about control.

*Artificial Intelligence*, 15(3):179-222, 1980.

**Dav91**

L. Davies.

*Handbook of Genetic Algorithms*.

Van Nostrand Reinhold, New York, 1991.

**DHR93**

D. Driankov, H. Hellendoorn, and M. Reinfrank.

*An Introduction to Fuzzy Control*.

Springer-Verlag, 1993.

**DK77**

R. Davis and J.J. King.

An overview of productions systems.

In E. Elcock and D. Michie, editors, *Machine Intelligence*, volume 8, pages 300-332. Ellis Horwood, Chichester, England, 1977.

**Doy79**

J. Doyle.

A truth maintenance system.

*Artificial Intelligence*, 12(3):231-272, 1979.

**DP80**

D. Dubois and H. Prade.

*Fuzzy Sets and Systems: Theory and Applications*.

Academic Press, 1980.

**DP86**

D. Dubois and H. Prade.

A set-theoretic view of belief functions.

*International Journal of General Systems*, 12:193-226, 1986.

**DP88**

D. Dubois and H. Prade.

*Possibility Theory - An Approach to the Computerized Processing of Uncertainty*.

Academic Press, 1988.

**DPS93**

D. Dubois, H. Prade, and S. Sandri.

On possibility/probability transformations.

In *Fuzzy Logic*. Kluwer, 1993.

**DPS96**

D. Dubois, H. Prade, and S. Sandri.

Possibilistic logic augmented with fuzzy unification.

In *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'96), Granada, Spain*, 1996.

**EN69**

G. Ernst and A. Newell.

*GPS: A Case Study in Generality and Problem Solving*.

Academy Press, New York, 1969.

**FBL71**

E.A. Feigenbaum, B.G. Buchanan, and J. Lederberg.

On generality and problem solving : A case study using the dendral program.

In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 6, pages 165-190. Edinburgh University Press, Edinburgh, GB, 1971.

**FHRSW82**

J. Fain, F. Hayes-Roth, H. Sowizral, and D.A. Waterman.

Programming in rosie: An introduction by means of examples.

Technical Report Report N-1646-ARPA, Rand Corporation, February 1982.

**FM77**

C.L. Forgy and J. McDermott.

Ops : A domain independent production system.

In *Proceedings of IJCAI 5*, pages 933-939, 1977.

**FTW83**

J.D. Farmer, T. Toffoli, and S. Wolfram.

*Cellular Automata: Proceedings of an Interdisciplinary Workshop at Los Alamos*.

New Mexico, North-Holland, Amsterdam, March 7-11 1983.

**Gen92**

Gensym.

*G2 Reference Manual*.

GENSYM Coorporation, 125 Cambridge Park Drive, Cambridge, MA 02140, 1992.

**Gev87**

W.B. Gevarter.

The nature and evaluation of commercial expert systems building tools.

*IEEE Computer*, pages 24-41, May, 1987.

**GH88**

D.E. Goldberg and J.H. Holland.
Genetic algorithms and machine learning: Introduction to the special issue on genetic algorithms.
*Machine Learning*, 3, 1988.

**Gol89**

D.E. Goldberg.
*Genetic Algorithms in Search, Optimization, and Machine Learning*.
Addison-Wesley Publishing Company, Reading, MA, 1989.

**Hay77**

P.J. Hayes.
In defense of logic.
In *Proceedings of IJCAI 5*, pages 559-565, 1977.

**HB94**

J. Heitkoetter and D. Beasley, editors.
*The Hitch-Hiker's Guide to Evolutionary Computation: A list of Frequently Asked Questions*.
USENET:comp.ai.genetic. Available via anonymous EMAIL from rtfm.mit.edu:/pub/usenet/news.answers/ai-faq/genetic/, 1994.

**HDE78**

P.E. Hart, R.O. Duda, and M.T. Einaudi.
Prospector - a computer-based consultation system for mineral exploration.
*Mathematical Geology*, 10(5), 1978.

**Hil89**

W.C. Hill.
The mind at ai: Horseless carriage to clock.
*The AI Magazine*, pages 29-41, Summer 1989.

**HM85**

J.Y. Halpern and Y.O. Moses.
A guide to the modal logics of knowledge and belief.
In *Proceedings of IJCAI 9*, pages 480-490, 1985.

**HN90**

R. Hecht-Nielsen.
*Neurocomputers*.
Addison-Wesley Publishing Company, Reading, MA, 1990.

**Hol75**

J.H. Holland.
*Adaptation in Natural and Artificial Systems*.
University of Michigan Press, Ann Arbor, 1975.

**Hol86**

J.H. Holland.
Escaping brittleness: The possibilities of general purpose learning algorithms applied to parallel rule-based systems.
In R.S. Mishalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning 2*, pages 593-623. Kaufman, 1986.

**Hop82**

J.J. Hopfield.
Neural networks and physical systems with emergent collective computational abilities.
In *Proceedings of the National Academy of Science, U.S.A.*, volume 79, pages 2554-2558, 1982.

**HRWL83**

F. Hayes-Roth, D.A. Waterman, and D.B. Lenat.
*Building Expert System*.
Addison-Wesley Publishing Company, Reading, MA, 1983.

**JSS95**

A.V. Joshi, S.C. Sahasrabudhe, and K. Shankar.
Sensitivity of combination schemes under conflicting conditions and a new method.
In J. Wainer and A. Carvalho, editors, *Advances in Artificial Intelligence*, pages 39-47. Lecture Notes in Computer Science, Springer-Verlag, 1995.

**KJ83**

H.E. Kyburg Jr.

The reference class.

*Philosophy of Science*, 50:374-397, 1983.

**Kli90**

G.J. Klir.

A principle of uncertainty and information invariance.

*International Journal of General Systems*, 17:249-275, 1990.

**Koh87**

T. Kohonen.

*Content-Addressable Memories*.

Springer-Verlag, Berlin, 1987.

**Kos92**

B. Kosko.

*Neural Networks and Fuzzy Systems*.

Prentice-Hall, 1992.

**Lee90**

C.C. Lee.

Fuzzy logic in control systems: Fuzzy logic controller - part. i e ii.

*IEEE Transactions on Systems, Man and Cybernetics*, 20(2):404-435, 1990.

**McC79**

P. McCorduck.

*Machines Who Think*.

Freeman, San Francisco, 1979.

**McC80**

J. McCarthy.

Circumscription - a form of non-monotonic reasoning.

*Artificial Intelligence*, 13(1,2):27-39, 1980.

**McD78**

J. McDermott.

Tarskian semantics, or no notation without denotation.

*Cognitive Science*, 2(3):277-282, July-September 1978.

**McD82**

J. McDermott.

R1 : A rule-based configurer of computer systems.

*Artificial Intelligence*, 19(1):39-88, September 1982.

**MF78**

J. McDermott and C. Forgy.

Production system conflict resolution strategies.

In D. Waterman and F. Hayes-Roth, editors, *Pattern Directed Inference Systems*, pages 177-199. Academic Press, New York, 1978.

**MH69**

J. McCarthy and P.J. Hayes.

Some philosophical problems from the standpoint of artificial intelligence.

In D. Michie and B. Meltzer, editors, *Machine Intelligence 4*, pages 463-502. Edinburgh University Press, Edinburgh, GB, 1969.

**Min75**

M. Minsky.

A framework to represent knowledge.

In *The Psychology of Computer Vision*, pages 211-277. McGraw-Hill, 1975.

**Mor73**

T.P. Moran.

The symbolic nature of visual imagery.

In *Proceedings of IJCAI 3*, pages 472-477, 1973.

**MP43**

W.S. McCulloch and W.H. Pitts.

A logical calculus of ideas immanent in nervous activity.

*Bulletin of Mathematical Biophysics*, 5:115-133, 1943.

**MP69**

M.L. Minsky and S.A. Papert.

*Perceptrons: An Introduction to Computational Geometry*.

M.I.T. Press, 1969.

**New80**

A. Newell.

Physical symbol systems.

*Cognitive Science*, 4:135-183, 1980.

**Nil71**

N.J. Nilsson.

*Problem Solving Methods in Artificial Intelligence*.

McGraw-Hill, New York, 1971.

**NL87**

B. Nebel and K. Von Luck.

Issues of integration and balancing in hybrid knowledge representation systems.

In K. Morik, editor, *Proceedings of the 11th German Workshop on Artificial Intelligence*, pages 115-123, 1987.

September-October.

**Paw92**

Z. Pawlak.

Rough sets: a new approach to vagueness.

In L.A. Zadeh and J. Kacprzyk, editors, *Fuzzy Logic for the Management of Uncertainty*. John Wiley & Sons Inc., New York, 1992.

**Pea87**

J. Pearl.

*Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*.

Morgan Kaufmann Publishers Inc., San Mateo, CA, 1987.

**Pos43**

E. Post.

Formal reductions of the general combinatorial problem.

*American Journal of Mathematics*, 65:197-268, 1943.

**PS85**

P.F. Patel-Schneider.

A decidable first-order logic for knowledge representation.

In *Proceedings of IJCAI 9*, pages 455-458, 1985.

Also AI Technical Report Number 45, Schlumberger Palo Alto Research.

**Qui68**

M.R. Quillian.

Semantic memory.

In M. Minsky, editor, *Semantic Information Processing*, pages 216-270. M.I.T. Press, Cambridge, MA, 1968.

**Rei80**

R. Reiter.

A logic for default reasoning.

*Artificial Intelligence*, 13(1-2):81-132, April 1980.

**RHW86**

D.E. Rumelhart, G.E. Hilton, and R.J. Williams.

Learning internal representations by error propagation.

In D.E. Rumelhart and J. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition 1: Foundations*, volume 1, pages 318-362. M.I.T. Press, Cambridge, MA, 1986.

**Ric83**

E. Rich.

*Artificial Intelligence*.

McGraw-Hill Book Company, 1983.

**RJP79**

Duda R.O, Gaschnig J., and Hart P.E.

Model design in the prospector consultant system for mineral exploration.

In D. Michie, editor, *Expert Systems in the Micro-eletronic Age*. Edinburgh University Press, 1979.

**RN75**

D.E. Rumelhart and D.A. Norman.

The active structural networks.

In D.A. Norman and D.E. Rumelhart, editors, *Explorations in Cognition*. W.H. Freeman, San Franscisco, 1975.

**Ros62**

F. Rosenblatt.
*Principles of Neurodynamics: Perceptrons and Theory of Brain Mechanisms*.
Spartan Books, Washington, D.C., 1962.

**SA75**

R.C. Schank and R.P. Abelson.
Scripts, plans and knowledge.
In *Proceedings of IJCAI 4*, pages 151-157, 1975.

**SA77**

R.C. Schank and R. Abelson.
*Scripts, Plans, Goals and Understanding*.
Laurence Ealbaum Associates Inc., Milesdale, N.J., 1977.

**San93**

S. Sandri.
Local propagation of information on directed markov trees.
In B. Bouchon-Meunier Et Al., editor, *Uncertainty in Intelligent Systems*. Elsevier Science Publishers, 1993.

**SBM93**

S. Sandri, G. Bittencourt, and M. Marengoni.
The use of possibilistic logic PL1 in a customizable tool for the generation of production-rule based systems.
In *Second European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'93), Granada, Spain, November 8-10*, 1993.

**Sch95**

Hans-Paul Schwefel.
*Evolution and Optimum Seeking*.
John Wiley & Sons Inc., New York, 1995.

**SDP95**

S. Sandri, D. Dubois, and H. Prade.
Elicitation, pooling and assessement of expert judments using possibility theory.
*IEEE Transactions on Fuzzy Systems*, 1995.

**Sha76**

G. Shafer.

*A Mathematical Theory of Evidence*.

Princeton University Press, 1976.

**Sho76**

E.H. Shortliffe.

*Computer-Based Medical Consultations : MYCIN*.

American Elsevier, New York, 1976.

**SJ84**

G.L. Steele Jr.

*Common LISP, the Language*.

Digital Press, Burlington, 1984.

**SMS92**

A.C. Stylianou, G.R. Madey, and R.D. Smith.

Selection criteria for expert system shells: A socio-technical framework.

*Communications of the ACM*, 35(10):32-48, October 1992.

**Som90**

LÉA SombÉ.

*Reasoning Under Incomplete Information in Artificial Intelligence*.

John Wiley & Sons Inc., New York, 1990.

O grupo Léa Sombé é formado por P. Besnard, M.-O. Cordier, D. Dubois, L.F. del Cerro, C. Froidevaux, Y. Moinard, H. Prade, C. Schwind e P. Siegel.

**THT87**

D.S. Touretzky, J.F. Horty, and R.H. Thomason.

A clash of intuitions : The current state of nonmonotonic multiple inheritance systems.

In *Proceedings of IJCAI 10*, pages 476-482, 1987.

**Wat73**

D.A. Waterman.

Pas-ii reference manual.

Technical report, Carnegie-Mellon University, June 1973.

Computer Science Department Report.

**Wat86**

D.A. Waterman.

*A Guide to Expert Systems*.

Addison-Wesley Publishing Company, Reading, MA, 1986.

**Win84**

P.H. Winston.

*Artificial Intelligence (2nd Edition).*

Addison-Wesley Publishing Company, Reading, MA, 1984.

**Woo75**

W.A. Woods.

What's in a link : Foundations for semantic networks.

In D.G. Bobrow and A. Collins, editors, *Representation and Understanding: Studies in Cognitive Science*. Academic Press, New York, 1975.

**Zad65**

L.A. Zadeh.

Fuzzy sets.

*Information and Control*, 8:338-353, 1965.

**Zad78**

L.A. Zadeh.

Fuzzy sets as a basis for a theory of possibility.

*Fuzzy Sets and Systems*, 1:3-28, 1978.

**Zad79**

L.A. Zadeh.

A theory of approximate reasoning.

In D. Mitchie J.E. Hayes and L.I. Mikulich, editors, *Machine Intelligence 9*. Wiley Masson, New York, 1979.

Disponível em:< http://www.das.ufsc.br/gia/softcomp/> Acesso em.: 17 set. 2007.