UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

Apostila

CIC250 - Inteligência Artificial

Álvaro Guarda

Ouro Preto - MG
Setembro de 2004

Sumário

1. Introdução 3
1.1. Objetivos do Curso 3
1.2. Evolução da Inteligência Artificial 3
1.3. Áreas de Pesquisa da IA 3
1.4. Referências 4
2. Caracterização 5
2.1. Discussão e Tentativa de Definição 5
2.2. Comparação entre IA e Computação Convencional 7
2.3. Automação de Atividades 7
3. Resolução de Problemas 8
3.1. Espaço de Estados 8
3.1.1. Representação do Problema em Espaço de Estados 8
3.1.2. Implementação da Representação do Problema em Espaço de Estados 10
3.1.3. Métodos de Busca Cegos em Espaço de Estados 10
3.1.3.1. Busca em Profundidade 11
3.1.3.2. Busca em Profundidade Acíclica 12
3.1.3.3. Busca em Extensão (Largura) 12
3.2. Resolução de Problemas como Busca por Redução de Problemas 15
3.3. Métodos de Busca Heurística 16
3.3.1. Subida da encosta ("Hill-Climbing") 17
3.3.2. Subida da encosta para a trilha mais íngreme ("Stepest-Hill-Climbing") 18
3.3.3. Busca pela melhor escolha ("Best-First") 19
3.3.4. Têmpera Simulada ("Simulated Annealing") 25
3.3.5. Técnicas Evolutivas 26
4. Sistemas Baseados em Conhecimento 28
4.1. Propriedades de Sistemas de Representação do Conhecimento 28
4.2. Formas de Representação do Conhecimento 28
4.3. Sistemas de Produção (SP) 29
4.4. Sistemas Baseados em Regras 31
5. Introdução aos Sistemas Especialistas 32
5.1. Componentes Básicos de um SE 32
5.2. Arquitetura do Motor de Inferência 33
5.3. Alguns Exemplos de Sistemas Especialistas 37
5.4. Ferramentas de Desenvolvimento de Sistemas Especialistas 38
5.5. Aquisição de Conhecimento 39
5.6. Problemas 39
5.7. Conclusões 39
6. Engenharia de Conhecimento 40
6.1. Metodologias de Desenvolvimento 40
6.2. Atividades em um Sistema com IA 40
6.3. Etapas na Construção de um pequeno SE 42
Ponderações 45

1. Introdução

O termo "Inteligência Artificial" (IA) desperta interesse e muitas expectativas, freqüentemente equivocadas. Além do termo que é um tanto ambicioso, existe muito sensacionalismo sobre IA em revistas, jornais e inclusive em livros. Alguns exemplos de afirmações exageradas são:

- "O homem vai tentar reproduzir o computador à sua imagem". Artigo "A revolução da IA" (Estado de Minas, 18/11/96).
- "O objetivo principal da IA é construir uma pessoa, ou mais humildemente, um animal", Charmiak e McDermott.
- "Um computador é inteligente se possui qualquer uma das habilidades mentais que fazem uma pessoa ser considerada inteligente" [Araribóia88].

Assim, como as expectativas não são realistas, os resultados que são obtidos na área freqüentemente são muito mais tímidos do que se esperava. Isto gera uma certa de desconfiança por parte de pesquisadores de outras áreas e acaba até prejudicando o desenvolvimento da IA. Entretanto, a IA já desenvolveu muitas técnicas que são úteis nas mais diversas áreas: otimização, *datamining*, jogos, tradução automática, processamento de imagens, interface homem-máquina, etc.

1.1. Objetivos do Curso

No curso alguns tópicos são estudados procurando aprofundar os aspectos importantes, permitindo que o aluno possa dar continuidade ao estudo da área e aplicar os conhecimentos adquiridos de forma autônoma. Outros pontos são apresentados sem caráter formativo, mas procurando dar uma visão abrangente da área e de suas aplicações.

Ao final do curso o aluno deverá dominar diversos pontos da IA clássica e conhecer alguns tópicos avançados em IA. Em algumas partes do curso, além do conhecimento teórico, são abordados aspectos práticos, fazendo experimentações e implementando diversas técnicas.

1.2. Evolução da Inteligência Artificial

. . . . - 1960	**Pré-IA**
	Lisp
	Sistemas de Produção (evolução de sistemas genéricos).
1960 - 1965	**Período do entusiasmo**
	Macsyma, Dendral
1965 - 1970	**Período das Trevas**
	Conscientização das limitações
1970 - 1975	**Período da renascença**
	Hearsy,
	Mycin (ajudava a diagnosticar doenças no sangue, tinha 450 regras e fazia análise do sangue),
	Prospector (fazia prospecção geológica),
	Prolog
1975 - 1980	**Período da associação**
	Áreas de aplicação
1980 -	**Período do empreendimento**
	Popularização de SE
	Ferramentas e Ambientes

1.3. Áreas de Pesquisa da IA

- Processamento do Conhecimento (simbólico)
  - Modelagem e Representação
  - Métodos de Inferência
  - Sistemas Baseados em Conhecimento
  - Aquisição de Conhecimento
  - Aprendizado Computacional

- Redes Neuronais
- Sistemas e Linguagens para IA
- Arquiteturas Específicas para IA
- Processamento de Linguagem Natural
- Robótica Inteligente
    - Síntese de Voz
    - Percepção
    - Manipulação
    - Planejamento

1.4. Referências

O conteúdo desta apostila é fortemente baseado na bibliografia básica do programa da disciplina, CIC250 – Inteligência Artificial, [Rich93] e [Bratko90]. Diversas partes do texto também foram baseadas no material de aula do Prof. Elton Silva.

2. Caracterização

2.1. Discussão e Tentativa de Definição

O termo "Inteligência Artificial" não é de fácil definição e há controvérsias entre vários pesquisadores. Abaixo são apresentadas algumas definições para as palavras que compõem o termo retirado do Dicionário Michaelis e outros.

Artificial

- Produzido por arte ou indústria do homem e não por causas naturais.
- Que envolve artifício.
- Contrafeito, fingido, postiço.
- Feito pelo homem.
- Imitação.
- Oposto de natural.
- Tenta imitar o comportamento humano.

Inteligência

- Faculdade de entender, pensar, raciocinar e interpretar; entendimento, intelecto.
- Compreensão, conhecimento profundo.
- Pessoa de grande esfera intelectual.
- Habilidade em fazer determinada coisa.
- Faculdade ou capacidade de aprender, compreender.
- Facilidade de adaptação.
- Intelectualidade.
- Destreza mental, perspicácia.
- Habilidade mental medida pelo QI (capacidade de manipulação simbólica).

A inteligência deve ser compreendida não apenas como aptidão para o manejo de símbolos e números, mas sobretudo como habilidade para interagir com o meio ambiente, lidar com situações inesperadas, fazer uso de analogias, metáforas e abstrações.

Os grandes gênios da humanidade ajudaram a fixar na cultura ocidental um padrão de "homem inteligente". Entretanto, este padrão apresenta alguns problemas:

- Albert Einstein : Maltratava a mulher como uma escrava particular, era excessivamente rabugento.
- Sigmund Freud : Sofria do complexo de Édipo, era obcecado pela mãe.
- Karl Max : Queria demolir o capitalismo, mas era um gastador inveterado, não sabia administrar as suas finanças e vivia pedindo dinheiro aos amigos.
- Charles Darwin : Pesquisador compulsivo, tinha síndrome de pânico. Tinha crises de vômito com medo da Teoria da Evolução.

Assim, a inteligência passou a ser vista e discutida sob vários aspectos:

- Inteligência racional (acadêmica): geralmente associada a aptidões lingüísticas e numéricas.
- Inteligência intra-pessoal : conhecer-se a si mesmo.
- Inteligência social : relações humanas, entender os outros.
- Inteligência corporal : relacionamento com o espaço (ex.: Garrincha, Pelé, Barishnikov).
- Inteligência emocional : autocontrole, zelo, persistência, capacidade de se motivar, etc.

Atualmente há um consenso de que um alto QI não é garantia de sucesso e passou-se a adotar o que chamamos de Inteligência Emocional (IE) juntamente com o QI. A habilidade de lidar com características avaliadas pela IE pode dar a verdadeira medida da inteligência humana: autocontrole, zelo, persistência, capacidade de se motivar, etc. Hoje em dia as empresas estão valorizando muito essas capacidades em seus futuros empregados.

Neste contexto surge a questão da necessidade e da possibilidade de se prover capacidades emocionais em máquinas. Desconsiderando esta questão, apesar de parecer que as definições apresentadas são suficientes, a definição de inteligência é superficial, descrevendo apenas qualidades ou apresentando sinônimos. Nem mesmo os pesquisadores em Psicologia Cognitiva sabem como a inteligência funciona, como é formada e o que é necessário para que ela se manifeste.

Existem muitas discussões filosóficas sobre o assunto. Para muitos cientistas, a inteligência está relacionada com vontade própria, intencionalidade, e existem outras esferas de discussão:

Criacionista :

- ser inteligente transcende a criação.
- Inteligência ligada ao conceito de alma, espírito.
- Capacidade de auto-avaliação, não agir só por instinto.

Evolucionista :

- Alma = cérebro
- A mente seria um efeito complexo das leis que governam as partículas físicas que compõe o cérebro.
- homem seria apenas um robô biológico.

Alguns cientistas da computação enfatizam que os computadores são "idiotas" com capacidade de processar informação em altíssima velocidade, seguindo exatamente as "ordens" dos seus programadores.

Russel e Norvig [Russel95] propõem uma organização de diferentes definições de IA variando em duas dimensões principais e formando quatro categorias como apresentado na Figura 1. A dimensão ocupando o eixo horizontal trata da forma de funcionamento: humano x máquina (racionalmente). A dimensão no eixo vertical trata da ação: pensar x agir.

Sistemas que pensam como humanos	**Sistemas que pensam racionalmente**
"The exciting new effort to make computers think ... machines with minds, in the full and literal sense" [Haugeland75] "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning, …" [Bellman78]	"The study of mental faculties through the use of computational models" [Charniak85] "The study of the computations that make it possible to perceive, reason, and act" [Winston92]
Sistemas que agem como humanos	**Sistemas que agem racionalmente**
"The art of creating machines that perform functions that require intelligence when performed by people" [Kurzweil90] "The study of how to make computers do things at which, at the moment, people are better" [Rich91]	"A field of study that seeks to explain and emulate intelligent behavior in terms of computational processes" [Schalkoff90] "The branch of computer science that is concerned with the automation of intelligent behavior" [Luger93]

Figura 1 - Classificação de algumas definições de IA

2.2. Comparação entre IA e Computação Convencional

A Figura 2 apresenta as principais características que diferenciam um sistema de IA e um sistema construído de forma convencional.

IA	Computação Convencional
Não algorítmico	Algorítmico
Processamento Simbólico	Processamento Numérico
Não Determinista	Determinista
Fácil Modificação	Difícil Modificação
Estrutura Dados + Programas ↓ Estrutura de Controle ↓ Resultados	Estrutura Dados ↓ Programas ↓ Resultados

Figura 2 - IA x Computação Convencional

2.3. Automação de Atividades

A Figura 3 mostra uma classificação das atividades que são executadas nas empresas e o tipo de automação que é utilizado em cada caso.

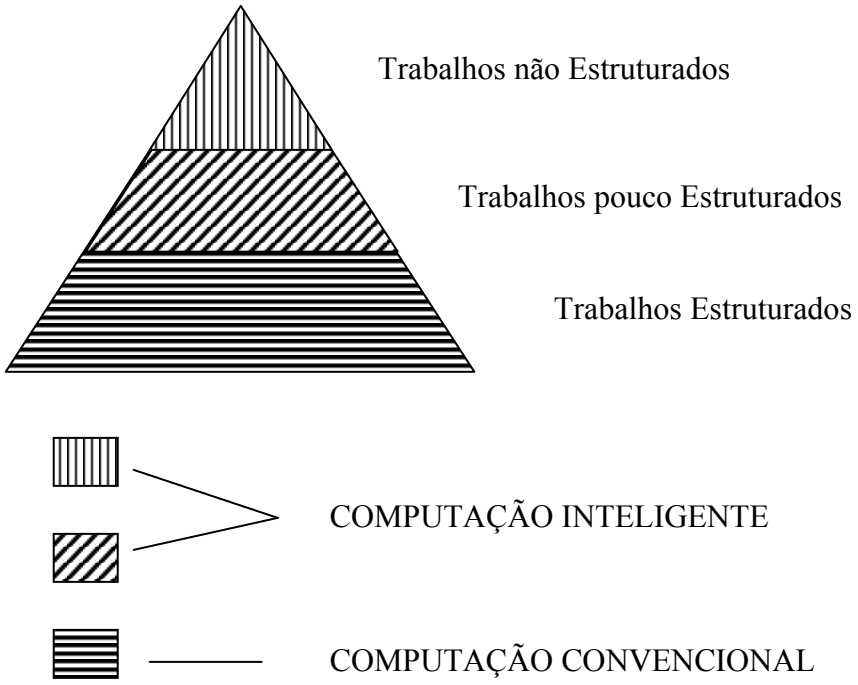


Figura 3 - Perfil das atividades em relação à mão-de-obra

3. Resolução de Problemas

Neste capítulo serão estudadas duas abordagens que permitem resolver a maior parte dos problemas:

- resolução utilizando Espaço de Estados;
- resolução por Decomposição de Problemas.

A escolha da abordagem depende do problema, como será visto mais adiante.

3.1. Espaço de Estados

Nesta abordagem a modelagem do problema é elaborada conforme a Figura 4. Partindo-se de um estado inicial deseja-se chegar a um estado final utilizando-se um conjunto de operações de transformação de estados.

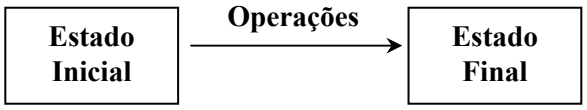


Figura 4 Representação do problema.

3.1.1. Representação do Problema em Espaço de Estados

Para facilitar a assimilação dos conceitos, a introdução ao assunto será feita através do mundo dos blocos. O mundo dos blocos proposto para estudo possui apenas três blocos (A, B e C) e uma mesa. Os blocos podem estar sobre a mesa ou empilhados. O problema é elaborar um plano para re-arranjar a disposição dos blocos de acordo com o desejado. Assim, um plano é uma seqüência de movimentos de blocos que, partindo de uma situação inicial, leva a uma situação desejada.

A Figura 5 apresenta um problema específico, onde estão definidas a situação inicial e a situação desejada. A definição do mundo estabelece uma classe de problema, pois é possível definir para o mesmo mundo novos problemas através da escolha de outras situações inicial e desejada.

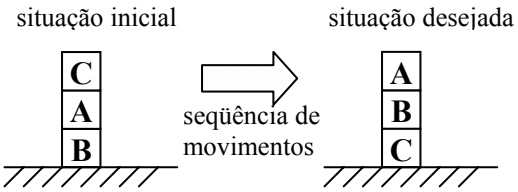


Figura 5 Um problema de re-arranjo de blocos (baseado em [Bratko2001]).

Neste problema, a modelagem utilizando a abordagem *Espaço de Estados* é natural: a situação inicial dos blocos é o estado inicial, a situação desejada é o estado final e a seqüência de movimentos é uma seqüência de operações válidas do conjunto de operações de transformação de estados. O conjunto de operações pode ser formado pelas operações apresentadas na Figura 6.

(1) coloque o bloco A sobre a mesa;
(2) coloque o bloco A sobre o bloco B;
(3) coloque o bloco A sobre o bloco C;
(4) coloque o bloco B sobre a mesa;
(5) coloque o bloco B sobre o bloco A;
(6) coloque o bloco B sobre o bloco C;
(7) coloque o bloco C sobre a mesa;
(8) coloque o bloco C sobre o bloco A;
(9) coloque o bloco C sobre o bloco B.

Figura 6 Um possível conjunto de operações para o problema da Figura 5.

Uma operação é válida quando ela respeita as restrições impostas na definição do problema. As restrições de movimentos no problema proposto são bastante simples: é permitido mover apenas um bloco de cada vez, não pode haver outros blocos em cima do bloco a ser movimentado nem do bloco sobre o qual se quer colocar um bloco em cima. Por exemplo, para aplicar a operação (2) não poder haver blocos em cima do bloco A nem do bloco B. Desta forma, partindo da situação inicial do problema apresentado na Figura 5, existe apenas um movimento legal: coloque o bloco C sobre a mesa. Na nova situação, após

o movimento, existem três movimentos legais possíveis: coloque o bloco A sobre a mesa, coloque o bloco A sobre o bloco C ou coloque o bloco C sobre o bloco A.

Os estados do problema e as operações de transformação de estados formam um grafo dirigido chamado *espaço de estados*, onde os nodos são os estados e os arcos são as operações. Como o mundo dos blocos sobre o qual se está estudando possui um número limitado de elementos, então a classe de problemas e o espaço de estados são finitos, conforme a Figura 7.

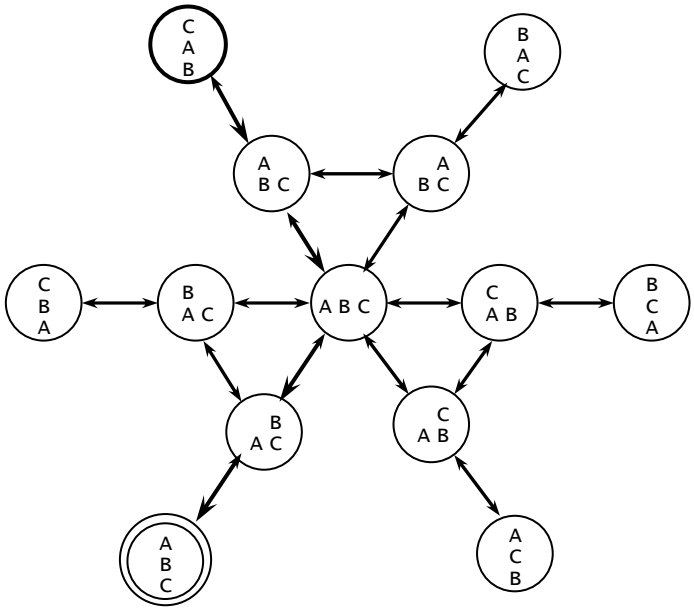


Figura 7 Representação gráfica do espaço de estados para a classe de problemas em estudo (baseado em [Bratko2001]). O problema apresentado na Figura 5 está em evidência: o estado inicial está com linha mais grossa, o estado final está com linha dupla concêntrica e uma solução possível é o caminho com setas maiores.

Encontrar uma solução equivale a achar um caminho no espaço de estados que, partindo do estado inicial, chega ao estado final. A interpretação da solução no problema do mundo dos blocos é uma seqüência de movimentos de blocos, ou seja, um plano para re-arranjar a pilha de blocos de acordo com a definição do problema.

Resumindo, um problema pode ser descrito formalmente utilizando a abordagem *Espaço de Estados* pelos seguintes elementos:

1. especificação do espaço de estados;
2. definição de um ou mais estados neste espaço de estados que descreva situações de partida do processo de solução do problema, são os estados iniciais de uma problema;
3. definição de um ou mais estados que poderiam ser aceitos como solução do problema, são os estados finais, metas ou objetivos de um problema e servem para determinar quando se chegou a uma solução. Isto pode ser feito através de uma condição.

No item (1), o espaço de estados pode ser especificado de forma extensiva, contendo todas as possíveis configurações dos objetos relevantes, conforme a Figura 7, entretanto, normalmente, isto é inviável. Assim, a forma usual é feita apenas através da definição do conjunto de operações de transformação de estados (regras do jogo) e o espaço de estados fica implícito.

Alguns pontos a serem considerados nesta atividade são:

- Que hipóteses não declaradas explicitamente estão presentes na descrição não-formal do problema?
- Quão geral devem ser as operações?

  Por exemplo, no lugar das nove operações definidas para o problema da Figura 5 poderia-se definir apenas duas operações:

  (1) coloque o bloco X sobre a mesa;
  (2) coloque o bloco X sobre o bloco Y.
- Quanto do trabalho necessário para resolver o problema deve ser pré-computado e representado nas operações?

  Por exemplo, as restrições podem estar representadas explicitamente nas operações:

  (1) se X está no topo de uma pilha então coloque o bloco X sobre a mesa;
  (2) se X e Y estão no topo de pilhas coloque o bloco X sobre o bloco Y.

Em um outro exemplo, as restrições são implicitamente representadas nas operações, conforme a Figura 8.

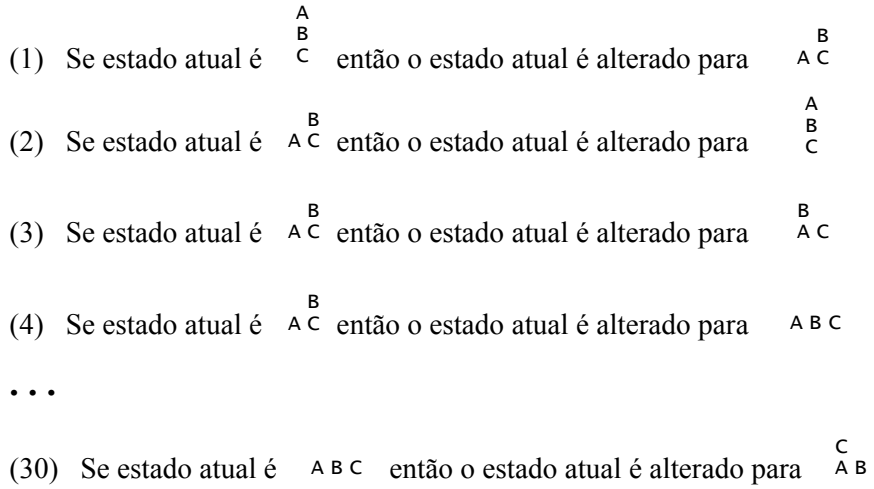


Figura 8 Conjunto de operações onde as restrições estão implicitamente representadas.

3.1.2. Implementação da Representação do Problema em Espaço de Estados

A implementação da representação depende da linguagem. Em Prolog, no caso do problema da Figura 5, os estado podem ser representados como uma lista de pilhas. Se houver no máximo três pilhas e uma pilha for representada também como uma lista, sendo que o topo é primeiro elemento da lista, então a representação final dos estados é [[<bloco no topo>, <bloco abaixo do topo>, ... , <bloco na base>], [<bloco no topo>, <bloco abaixo do topo>, ... , <bloco na base>], [<bloco no topo>, <bloco abaixo do topo>, ... , <bloco na base>]]. Desta forma, o estado inicial do problema é representado como a lista [[c, a, b], [], []] e o estado final pode ser representado como a lista [[a, b, c], [], []] ou [[], [a, b, c], []] ou [[], [], [a, b, c]]. A cláusula abaixo é uma implementação para se identificar quando se atinge o objetivo.

```
objetivo(Estado) :- member([a, b, c], Estado).
```

O(s) estado(s) sucessor(es) de um estado qualquer pode(m) ser definido(s) pela cláusula sucessor(<estado atual>, <estado sucessor>). Com esta sintaxe, o conjunto de operações da Figura 8 pode ser implementado como apresentado na Figura 9.

```
sucessor([[a, b, c], [], []], [[b, c], [a], []]).
sucessor([[a, b, c], [], []], [[b, c], [], [a]]).
sucessor([[],[a, b, c], []], [[a], [b, c], []]).
sucessor([[],[a, b, c], []], [[], [b, c], [a]]).
. . .
```

Figura 9 Uma implementação do conjunto de operações Figura 8.

Entretanto, esta implementação não é muito adequada porque exige escrever todas as configurações possíveis de pilhas. Uma implementação mais econômica é apresentada abaixo:

```
sucessor(Estado, [Pilha1, [Topo1 | Pilha2] | OutrasPilhas]) :-
    remove([Topo1 | Pilha1], Estado, EstadoAux),
    remove(Pilha2, EstadoAux, OutrasPilhas).

remove(Pilha, [Pilha | RestoEstado], RestoEstado).

remove(Pilha, [Paux | RestoEstado], [Paux | Resto]) :-
    remove(Pilha, RestoEstado, Resto).
```

3.1.3. Métodos de Busca Cegos em Espaço de Estados

A busca de uma solução é feita através da procura de um caminho no espaço de estados que leva do estado inicial para o estado final. Os métodos cegos fazem uma pesquisa sistemática do espaço de estados, porém não utilizam nenhum conhecimento para guiar a busca. Os dois principais métodos cegos são a "busca em profundidade" e a "busca em extensão".

A ação de percorrer o espaço de estados é feita de acordo com uma estratégia de controle que seleciona um estado e um operador que será aplicado ao estado para gerar os estados subseqüentes. A aplicação dos operadores nos estados iniciais e

intermediários é feita até que se chegue a um estado objetivo. Este processo, à medida que vai sendo executado, vai gerando a árvore de busca. Quando se encontra o estado final o processo é interrompido com sucesso. Dependendo do problema, a solução pode ser o próprio estado final ou o caminho entre o estado inicial e o estado final. A Figura 10 apresenta uma parte da árvore de busca para o problema da Figura 5.

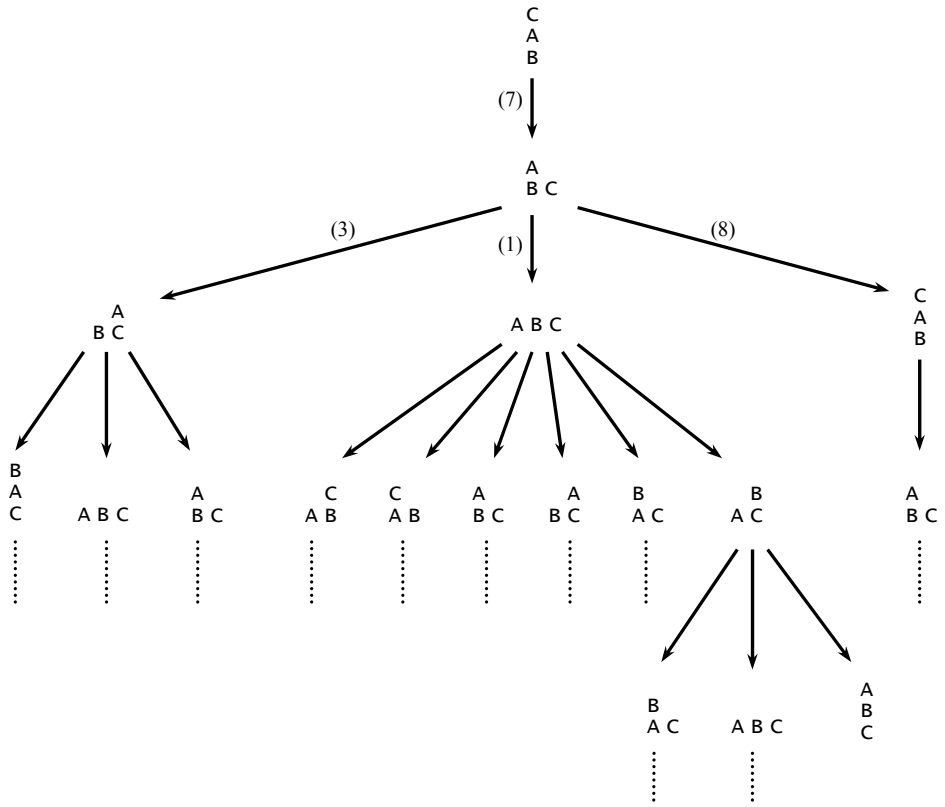


Figura 10 Árvore de pesquisa para o problema da Figura 5, utilizando o conjunto de operações da Figura 6.

3.1.3.1. Busca em Profundidade

A busca em profundidade explora um ramo da árvore de cada vez, ou seja, todos os nodos de um caminho da árvore são examinados antes dos nodos dos outros caminhos. A estrutura de dados associada é uma pilha, pois deve visitar primeiro os nodos mais recentes, aqueles que estão mais distantes da raiz. É necessário armazenar apenas o caminho que está sendo explorado e os caminhos já explorados podem ser descartados. A exploração dos caminhos alternativos pode ser feita através do retrocesso. No algoritmo abaixo não está previsto nenhum mecanismo de retrocesso.

Algoritmo:

1. Se o estado inicial é um estado objetivo, pare e retorne sucesso.
2. Senão, faça o seguinte até o sucesso (ou insucesso):
   a) Gere um sucessor "E" do estado inicial.
      Se não há mais sucessores, assinale insucesso.
   b) Senão, chame BUSCA EM PROFUNDIDADE com E como estado inicial.
   c) Se sucesso então é retornado sucesso
      Senão, continue o passo 2.

Abaixo é apresentada uma implementação em Prolog bastante simples e, portanto fácil de entender. A desvantagem desta implementação é que a mesma permite ciclos, o que pode conduzir a pesquisa a um ramo sem saídas.

```
pesquise(N, [N]) :- objetivo(N).

pesquise(N, [N | Caminho]):-
    sucessor(N, Naux),
    pesquise(Naux, Caminho).
```

A consulta em Prolog tem a sintaxe *pesquise(<início>, <solução>)*. A solução <solução> é uma lista de estados, partindo do estado <início> até um dos estados objetivos. Para o problema da Figura 5 a consulta é expressa como a seguir:

```
-? pesquise([c, a, b], [], []], Solução).
```

3.1.3.2. Busca em Profundidade Acíclica

A implementação abaixo evita ciclos. Entretanto ainda podem existir na árvore de pesquisa ramos infinitos que não contenham a solução. Esta situação é plausível em problemas cujo espaço de estados é infinito.

```
pesquise(N, Solução) :-  profundidade([], N, Solução).

profundidade(Caminho, N, [N | Caminho]) :-  objetivo(N).

profundidade(Caminho, N, Solução) :-
sucessor(N, Naux),
not(member(Naux, Caminho)),
profundidade([N | Caminho, Naux, Solução).
```

3.1.3.3. Busca em Extensão (Largura)

Todos os nodos de um nível da árvore são examinados antes dos nodos do nível seguinte. A estrutura de dados associada é uma fila, pois deve visitar primeiro os nodos mais antigos, ou seja, aqueles mais próximos da raiz.

Algoritmo:

1. Crie uma variável chamada "LISTA_DE_NODOS" e faça-a igual ao estado inicial.
2. Até que (um estado objetivo seja encontrado) ou (LISTA_DE_NODOS seja vazia) faça:
   a) Remova o primeiro elemento de LISTA_DE_NODOS e o chame "E".
      Se LISTA_DE_NODOS estava vazia, fim.
   b) Para cada maneira que cada regra combina com o estado descrito em E faça:
      i) Aplique a regra para gerar um novo estado.
      ii) Se o novo estado é um estado objetivo, pare e retorne este estado
      iii) Senão, adicione o novo estado ao final de LISTA_DE_NODOS.

Abaixo é apresentada uma implementação em Prolog.

```
% solve( Start, Solution):
%    Solution is a path (in reverse order) from Start to a goal

solve( Start, Solution)  :-
  breadthfirst( [ [Start] ], Solution).

% breadthfirst( [ Path1, Path2, ...], Solution):
%   Solution is an extension to a goal of one of paths

breadthfirst( [ [Node | Path] | _], [Node | Path])  :-
  goal( Node).

breadthfirst( [Path | Paths], Solution)  :-
  extend( Path, NewPaths),
  append( Paths, NewPaths, Paths1),
  breadthfirst( Paths1, Solution).

extend( [Node | Path], NewPaths)  :-
  bagof( [NewNode, Node | Path],
        ( sucessor( Node, NewNode), not(member( NewNode, [Node | Path] )) ),
        NewPaths),
  !.

extend( _, [] ).          %  Nodo não tem sucessores (bagof falhou!)
```

3.1.3.4. Comparação: Busca em Profundidade x Busca em Extensão

Busca em profundidade:

- Mesmo que haja uma solução, não há garantias de encontra-la, pois pode explorar um caminho infinito que não contém a solução.
- Por sorte, pode achar uma solução sem examinar muito o espaço de estado.
- É adequada quando existe solução na maioria dos ramos.
- Requer menos memória.

Busca em Extensão:

- Se houver solução, garante a obtenção da solução de menor caminho.
- Não corre o risco de examinar por longo tempo becos sem saída.
- Não é muito adequada quando o fator de ramificação é muito alto, pois exige armazenar muitos caminhos alternativos.
- Consome muita memória.

3.1.3.5. Outros Exemplos de Problemas e de Modelagem utilizando Espaço de Estados

Exemplo 1: Torre de Hanói

A representação gráfica do problema da Torre de Hanói é apresentada na Figura 11.

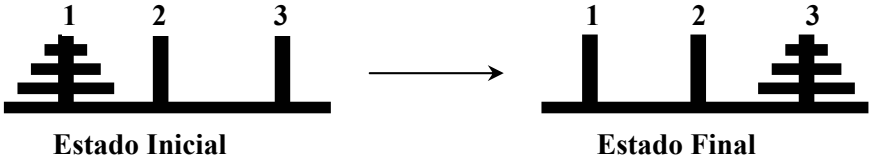


Figura 11 Torre de Hanói.

a) Estados

Sintaxe: (Menor, Média, Maior)

Menor = Número do poste onde está a argola menor.
Média = Número do poste onde está a argola média.
Maior = Número do poste onde está a argola maior.

Assim os estado inicial e final podem ser representados como:

Estado inicial : (1,1,1).
Estado final : (3,3,3).

b) Operações

Sintaxe: move (poste de origem, poste de destino)

move (1,2)
move (1,3)
move (2,1)
move (2,3)
move (3,1)
move (3,2)

Nesse caso devemos sempre observar as restrições do problema e a validade das operações. A Figura 12 apresenta a árvore de pesquisa deste problema.

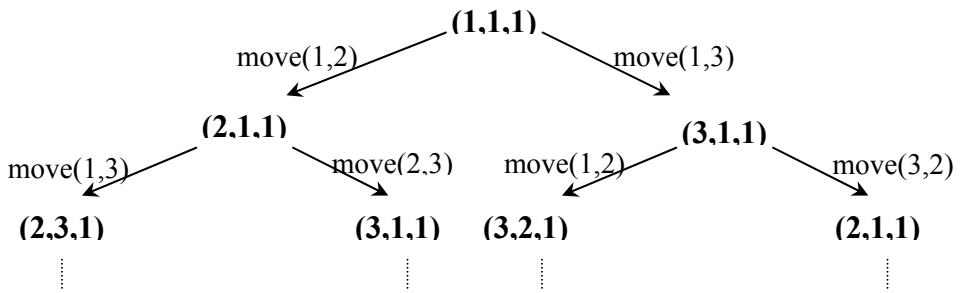


Figura 12 Árvore de pesquisa para o problema da Torre de Hanói.

Exemplo 2: Jogo dos 8

A Figura 13 mostra a representação gráfica para o problema do jogo dos 8.

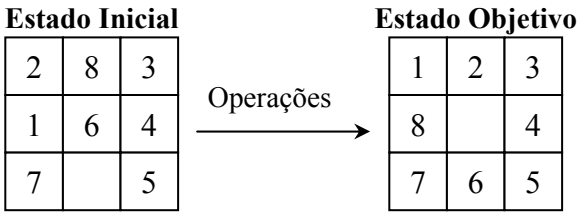


Figura 13 Representação do problema para o jogo dos 8.

É possível modelar as operações de diversas maneiras, mas a forma mais eficiente consiste em aplicar operações na casa vazia, o que possibilita que sejam necessárias apenas quatro operações:

(1) Mover a casa vazia para cima.
(2) Mover a casa vazia para baixo.
(3) Mover a casa vazia para esquerda.
(4) Mover a casa vazia para direita.

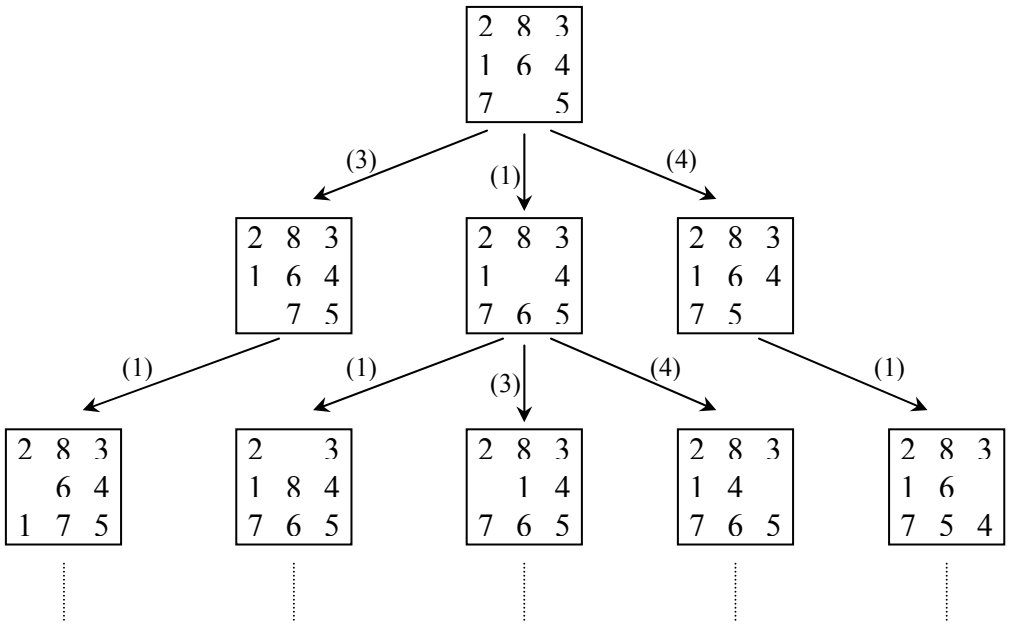


Figura 14 Árvore de pesquisa para o jogo dos 8.

Exemplo 3: Jogar Xadrez

1. Especificar a posição inicial do tabuleiro
   Matriz 8x8, onde cada posição contendo um símbolo que representa a peça apropriada (ou indicação de casa vazia) na posição oficial de abertura do jogo.
2. Definir as operações que representam movimentos legais
   As operações também podem ser definidas como um conjunto de regras de produção da forma :
   LADO.ESQ → LADO.DIR onde LADO.ESQ : Padrão a ser comparado antes de uma jogada.
   LADO.DIR : Descreve a mudança a ser feita.
   Exemplo: Regra "move peão branco"
   peãobranco em posição (j, i) ∧
   posição (j, i+1) está vazia ∧
   posição (j, i+2) está vazia → move peãobranco de posição (j, i) para posição (j, i+2).
3. Definir posições que representam vitória
   Qualquer posição do tabuleiro na qual o oponente não tenha um movimento legal, e seu rei esteja sendo atacado.

3.2. Resolução de Problemas como Busca por Redução de Problemas

Na resolução de problemas utilizando como enfoque a busca por redução de problemas, a idéia utilizada é a da decomposição de problemas em subproblemas com complexidade menor que a do problema original. A modelagem do problema é feita através da definição dos seguintes elementos:

- Descrição do problema inicial.
- Conjunto de operações de transformação de problemas.
- Conjunto de problemas primitivos (problemas que têm solução imediata, conhecida).

A solução é um conjunto de Problemas Primitivos encontrados a partir da aplicação das operações no problema inicial e nos subproblemas já gerados.

O espaço de busca é representado graficamente por um grafo E/OU. Neste grafo, os nodos contêm a descrição dos problemas (subproblemas) e os arcos indicam relações entre os problemas, isto é, como os problemas são decompostos. As folhas da árvore podem ser problemas primitivos ou nodos terminais (problemas insolúveis). Os nodos intermediários, sucessores de um determinado nodo, são do tipo OU, indicando que os nodos sucessores são problemas alternativos, ou do tipo E, indicando que os nodos sucessores são problemas conjuntivos.

A busca é a construção sistemática do grafo E/OU através da aplicação das operações que vão decompondo os problemas.

Exemplo 1: Torre de Hanói

A representação gráfica do problema da Torre de Hanói é apresentada na Figura 11.

Representação do problema

[<TAM. PILHA>,<PINO DE SAÍDA>,<PINO DE CHEGADA>]

Problema inicial : [3,1,3]

Problema primitivo : mover uma pilha de tamanho 1.

Operação

Para diferentes pinos I, J e K, o problema de mover uma pilha de argolas de tamanho n>1 do pino I para o pino K pode ser substituído por 3 subproblemas:

- Mover uma pilha de tamanho n-1 de I para J.
- Mover uma pilha de tamanho 1 de I para K.
- Mover uma pilha de tamanho n-1 de J para K.

Grafo E/OU

A Figura 15 apresenta o grafo E/OU que é o espaço de busca para problemas representados por Redução de Problemas.

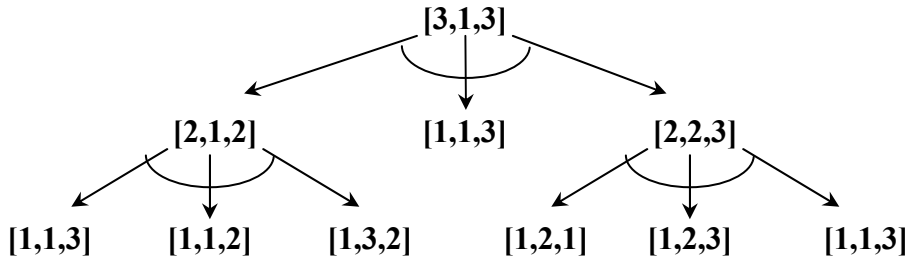


Figura 15 - Grafo E/OU para o problema da Torre de Hanói.

Solubilidade dos nodos

Um nodo é solúvel se:

- É um problema primitivo.
- É um nodo "E" cujos sucessores são todos solúveis.
- É um nodo "OU" onde pelo menos um dos sucessores é solúvel

Um nodo é insolúvel se:

- É um nodo terminal.
- É um nodo "E" onde pelo menos um dos sucessores é insolúvel.
- É um nodo "OU" cujos sucessores são todos insolúveis.

A Figura 16 apresenta uma árvore E/OU e como é a feita a busca em profundidade de uma solução. Os nodos são numerados por ordem de visita, sendo que os nodos marcados com T são problemas terminais e os nodos marcados com P são problemas primitivos. A Figura 17 apresenta como é feita a busca em extensão.

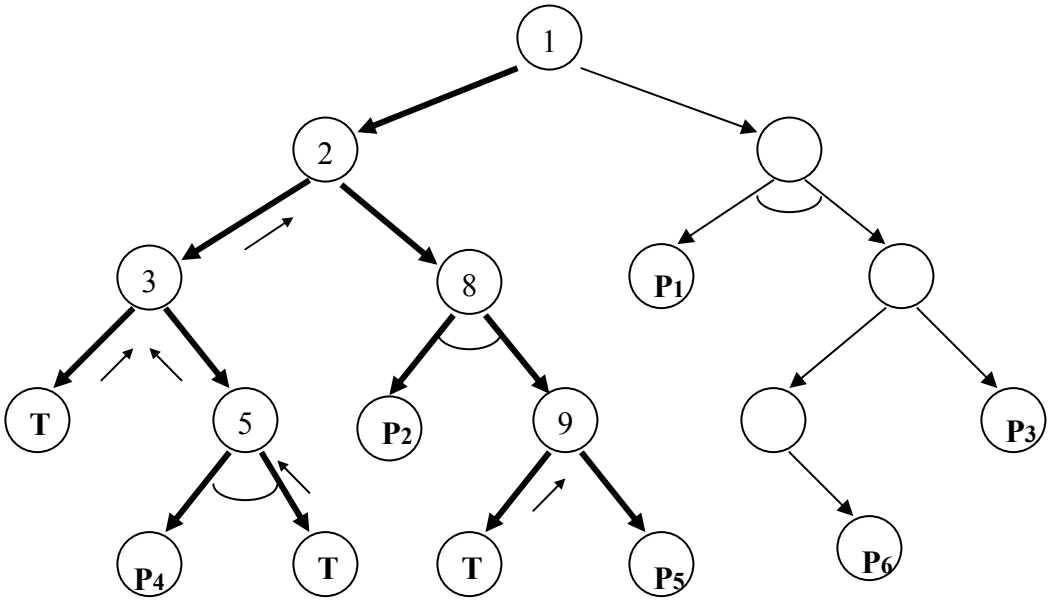


Figura 16 Exemplo de busca em profundidade em uma árvore E/OU. A solução é o conjunto de problemas primitivos {P2, P5}.

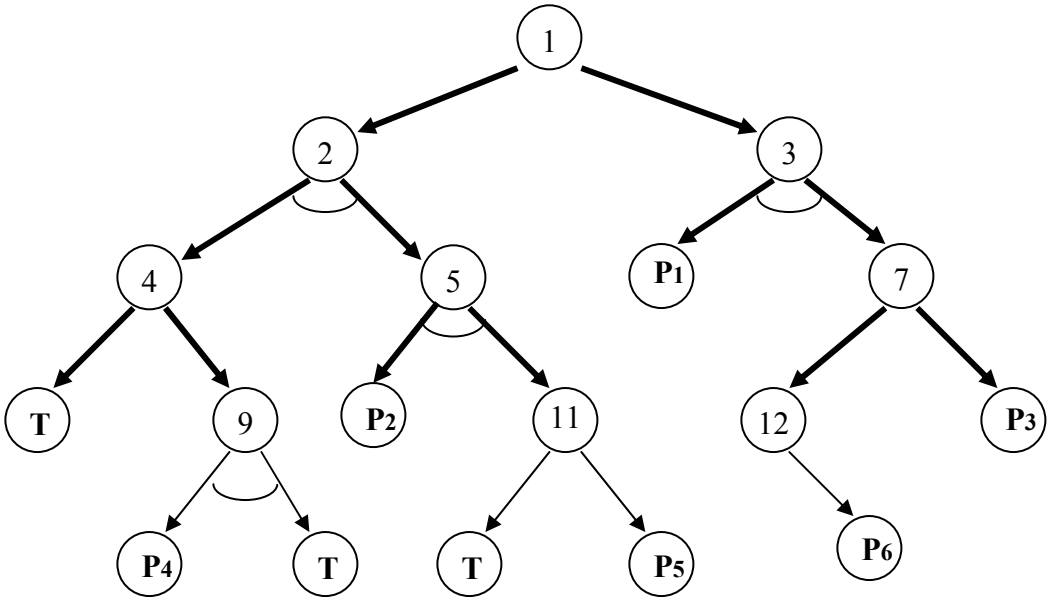


Figura 17 Exemplo de busca em extensão em uma árvore E/OU. A solução é o conjunto de problemas primitivos {P1, P3}.

3.3. Métodos de Busca Heurística

Os métodos de busca vistos anteriormente fazem uma pesquisa sistemática do espaço de estados ou do espaço de problemas, porém não são adequados para muitos problemas reais devido à explosão combinatória.

Os métodos heurísticos(1) utilizam alguma forma de conhecimento para orientar a busca e evitar a explosão combinatória. Entretanto, a eficiência depende fortemente em como o conhecimento é explorado.

O conhecimento a ser utilizado pode ser inserido manualmente, ou através de aprendizado automático.

(1) Heurística vem da palavra grega "heuriskein" que significa descobir.

Tipos de heurística no processo de busca

a) Heurística genérica:

Diferentes aspectos gerais do problema são considerados e avaliados de modo que o valor resultante da função heurística seja uma boa estimativa de quão efetivo pode ser o estado para alcançar a solução. Os métodos que utilizam apenas heurísticas genéricas, úteis a muitos tipos de problemas, são denominados métodos fracos e não são capazes de evitar a explosão combinatória.

Freqüentemente, esta medida consiste em uma estimativa da distância da solução. Por exemplo, no Jogo dos 8 a heurística pode ser o número de peças que estão fora do lugar.

b) Heurística específica:

Apresenta as convicções dos especialistas em uma determinada área. A maioria das regras dos Sistemas Especialistas pertence a essa classe. É extremamente eficiente se bem utilizada.

Função Heurística

Nos métodos de busca, o conhecimento sobre o problema é utilizado para compor uma função heurística. Esta função heurística tem a forma *f(Estado)* e é aplicada aos estados para atribuir notas indicando a proximidade do estado em relação à solução.

3.3.1. Subida da encosta ("Hill-Climbing")

A idéia deste método é maximizar a nota fornecida pela função heurística. A busca é feita com a geração de um estado sucessor do estado corrente. Se o estado sucessor tiver uma nota melhor, então ele passa a ser o estado corrente.

Algoritmo:

1. Estado corrente ← estado inicial
2. Repita até que o estado corrente seja a solução ou não existam operadores a serem aplicados ao estado corrente
   a) Escolha um operador que ainda não tenha sido aplicado ao estado corrente e aplique-o para produzir um novo estado.
   b) Avalie o novo estado.
      i) Se for a solução (um estado meta) retorne-o e encerre.
      ii) Se for melhor que o estado corrente então Estado corrente ← novo estado.

Características:

- Seleciona o primeiro vizinho que seja melhor.
- Problema: tende a convergir para uma solução que é um máximo local e não para o máximo global.
- É irrevogável (sem retrocesso).

Exemplo: Mundo dos Blocos

A Figura 18 apresenta o problema no qual será aplicado o método da subida da encosta.

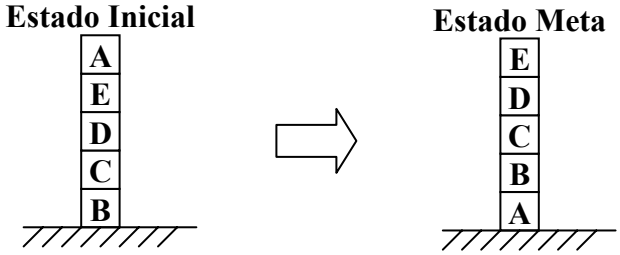


Figura 18 Representação gráfica de um problema do Mundo dos Blocos

Os operadores são:

(1) Pegue um bloco e coloque-o sobre a mesa
(2) Pegue um bloco e coloque-o sobre outro

A função heurística que será utilizada faz o seguinte cálculo: some um ponto para cada bloco em cima do bloco em que ele deva esta e subtraia um ponto para cada bloco que estiver sobre o bloco errado.

Com esta heurística os estados inicial e final recebem as notas: f(estado inicial) = 1 e f(estado meta) = 5. A árvore de busca é apresentada na Figura 19.

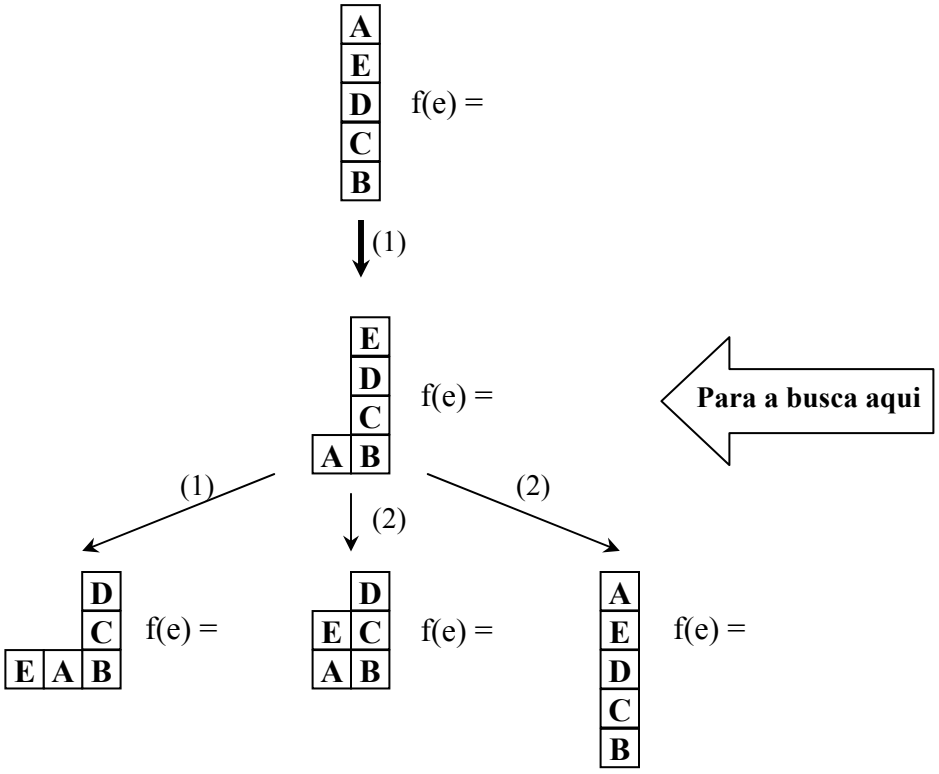


Figura 19 Árvore de busca para o problema da Figura 18 utilizando o método Subida da Encosta.

A utilização desta função heurística, neste problema em particular, conduz a busca a um máximo local. Isto corresponde à situação onde a vizinhança (sucessores) possui nota menor que o estado e não consegue encontrar uma solução. O motivo para este problema é que a heurística é local, não observando o contexto.

Uma função heurística melhor, que leve em conta o contexto global, pode ser implementada da seguinte forma: quando o bloco tem a estrutura de apoio correta, some um ponto para cada bloco da estrutura de apoio; para cada bloco com estrutura de apoio incorreta subtraia um ponto.

Esta nova função heurística permite que o processo de busca chegue a uma solução, como sugere a Figura 20.

3.3.2. Subida da encosta para a trilha mais íngreme ("Stepest-Hill-Climbing")

A idéia é a mesma do método anterior, porém examina toda a vizinhança e seleciona o melhor. Este método também é denominado busca do gradiente.

A heurística é aplicada localmente, ou seja, o caminho a ser seguido é selecionado apenas entre os nodos recém gerados, sendo a escolha baseada no resultado da aplicação da função heurística. A implementação pode ser feita ordenando-se o conjunto de nodos sucessores recém avaliados antes de incluí-los na pilha de nodos abertos. O campo de ordenação é a nota resultante da avaliação da função heurística, sendo que deve ficar no topo da pilha o nodo com a melhor nota, daqueles recém gerados.

Algoritmo:

1. Estado corrente ← Estado inicial.
2. Repita até que estado corrente seja a solução ou não haja mudança no estado corrente.
   a) Sucessor ← pior absoluto
   b) Para cada operador aplicável ao estado corrente.
      i) aplique o operador e gere um novo estado.
      ii) Avalie o novo estado.
      Se for a solução retorne-o e encerre
      Se for melhor então sucessor ← novo estado
   c) Se sucessor for melhor que estado corrente
      Então Estado corrente ← sucessor

Características:

- Seleciona o melhor vizinho, desde que seja melhor que o estado corrente.

- Problema: tende a convergir para uma solução que é um máximo local e não para o máximo global.
- É irrevogável (sem retrocesso).

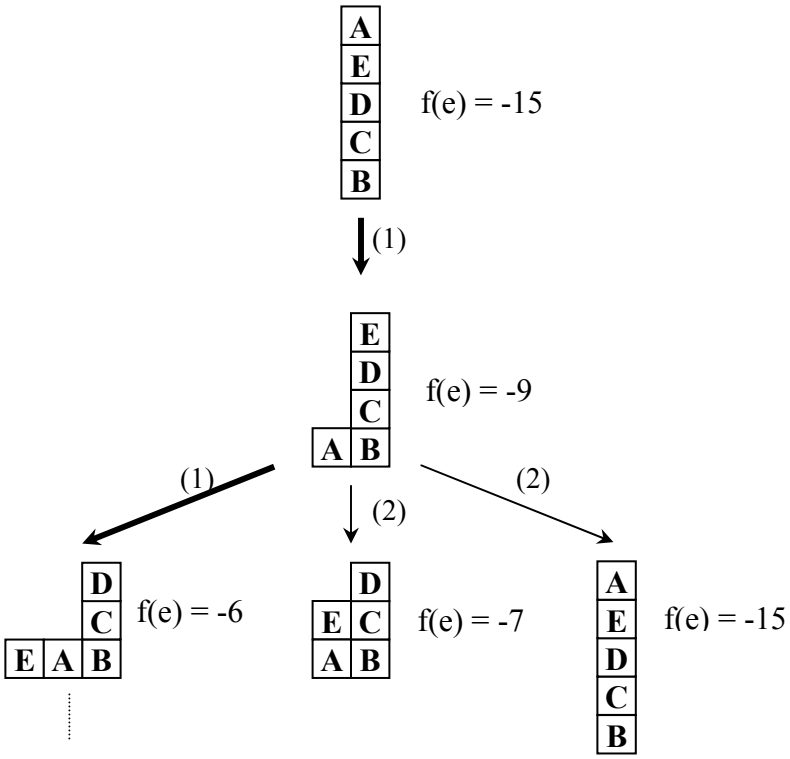


Figura 20 Árvore de busca para o problema da Figura 18 utilizando o método Subida da Encosta com uma função heurística mais adequada.

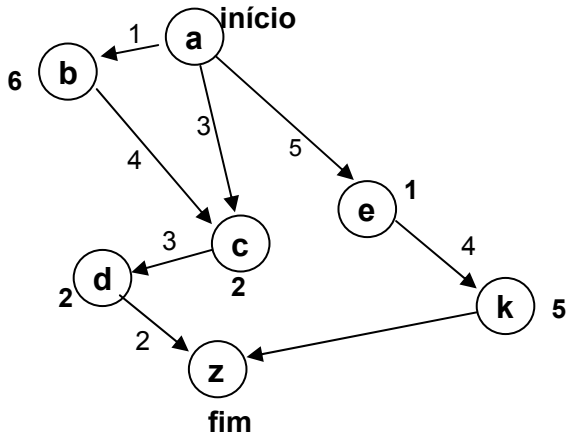
3.3.3. Busca pela melhor escolha ("Best-First")

A heurística é aplicada globalmente, isto é, o caminho a ser seguido é selecionado entre todos os nodos abertos até o momento. O nodo aberto com a melhor nota é escolhido para a expansão.

Exemplo: Achar o Menor Trajeto

Definição do Problema: Procurar o menor trajeto, ou uma alternativa aceitável, entre a cidade ***a*** e a cidade ***z.***

Mapa:



Representação do Estado: Nome da cidade
Representação do Espaço de Estados: O próprio mapa ou um grafo
Estado Inicial: a
Estado Final: z

Operações: percorrer uma estrada até uma cidade vizinha

Função Heurística (faz parte da modelagem do problema)

f(x) = g(x) + h(x)

onde

g(x) = distância percorrida da cidade **a** até a cidade **x**
(números em negrito, próximo às cidades)

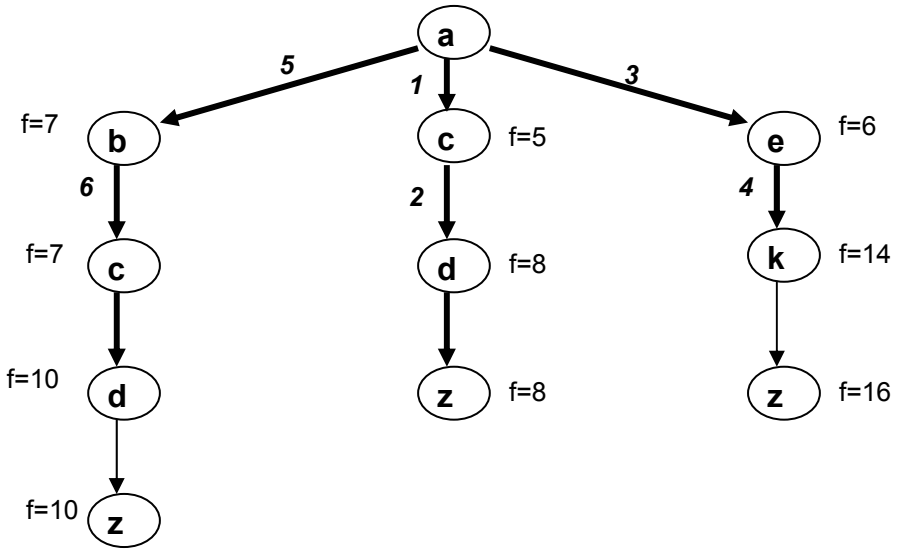
h(x) = distância em linha reta da cidade **x** até a cidade **z**
(números em negrito, próximo às cidades)

Problema: Achar o Menor Trajeto

Exemplos de Aplicação da Função Heurística

f(b) = g(b) + h(b) = 1 + 6 = 7
f(c) = g(c) + h(c) = 3 + 2 = 5
f(e) = g(e) + h(e) = 5 + 1 = 6

Árvore de Busca: os números nos arcos indicam a ordem de expansão



A solução encontrada pode eventualmente não ser o menor trajeto, dependendo do problema, mas pode ser uma solução bastante aceitável.

Exemplo: Achar o Menor Trajeto

Representação em Prolog

objetivo: goal(z).

sintaxe das operações: s(<nodo>, <nodo sucessor>, <custo>)

onde <custo> é a distância entre as cidades

```
s(a,b,1).
s(a,c,3).
s(a,e,5).
s(b,c,4).
s(c,d,3).
s(d,z,2).
s(e,k,4).
s(k,z,7).
```

Funções heurísticas:

g(x) é calculado a partir de <custo>

h(x) é definido como abaixo

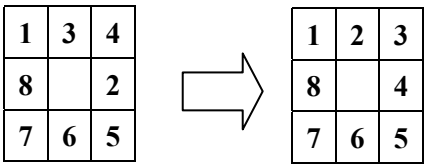
```
h(b,6).
h(c,2).
h(d,2).
h(e,1).
h(k,5).
h(z,0).
```

Consulta e resultado:

```
?- bestfirst(a, Sol), showsol(Sol).
---a
---c
---d
---z
Sol = [z, d, c, a]
```

Representação do Jogo dos 8

Definição do Problema



Representação em Prolog

Posições das peças: coordenadas cartesianas **X/Y**

Tabuleiro: Lista onde 1° elemento: casa vazia
2° elemento: peça 1
3° elemento: peça 2
. . .
9° elemento: peça 8

3	**1**	**2**	**3**	O estado inicial é representado como:
2	**8**		**4**	[2/2, 1/3, 3/2, 2/3, 3/3, 3/1, 2/1, 1/1, 1/2]
1	**7**	**6**	**5**	
	1	2	3	

3	**1**	**2**	**3**	O estado final é representado como:
2	**8**		**4**	[2/2, 1/3, 2/3, 3/3, 3/2, 3/1, 2/1, 1/1, 1/2]
1	**7**	**6**	**5**	
	1	2	3	

A cláusula *showsol* apresenta a solução como uma seqüência de posições do tabuleiro.

```
?- bestfirst([2/2,1/3,3/2,2/3,3/3,3/1,2/1,1/1,1/2], Solução), showsol(Solução).
```

Implementação:
A implementação em Prolog é apresentada abaixo e é baseada no algoritmo A*.

Principal Predicado: expand(Path, Tree, Bound, Tree1, Solved, Solution)

Parâmetros:

Path: caminho entre o nodo inicial e a sub-árvore Tree

Tree: sub-árvore corrente, que está sendo vizitada

Bound: limite para a expansão de Tree (f-value da próxima melhor alternativa)

Tree1: árvore expandida dentro do limite (bound)

Solved: yes - Solution é instanciado
no - Tree1 = Tree expandida, mas f-value > limite
never - indica que Tree é ramo sem saída

Solution: caminho entre o nodo inicial e o estado final

Cláusulas:

Caso 1: o nodo a ser expandido é um nodo final
- Conclui a busca

Caso 2: o nodo é folha e f-value < limite (bound)
- Gera sucessores e os expande enquanto não atinge o limite

Caso 3: o nodo não é folha e f-value < limite
- Percorre a árvore até chegar ao nodo folha c/ f-value < limite

Caso 4: o nodo não é folha e não possui sub-árvores
- Não deve ser expandido, pois é um ramo sem saída

Caso 5: o nodo com f-value > limite
- Não deve ser expandido, pois há opções mais interessantes

Representação da Árvore

Nodos em Aberto (folhas):
l(<estado> **,** <f(<estado>)>/<g(<estado>) **)**

Nodos Internos ((sub-)árvores)
t(<estado> **,** <f(<estado>)>/<g(<estado>)**,** <lista de sub-árvores> **)**

```
bestfirst(Start, Solution):-
  expand([], l(Start, 0/0), 9999, _, yes, Solution).          % Assume que 9999 é maior que qualquer f-value

% expand( Path, Tree, Bound, Treel, Solved, Solution):   Path is path between start node of search and subtree Tree,
% Treel is Tree expanded within Bound,  if goal found then Solution is solution path and Solved = yes

% Case 1: goal leaf-node, construct a solution path
        expand(P, l(N, _), _, _, yes, [N | P]):-
          goal(N).

% Case 2: leaf-node, f-value less than Bound  -  Generate successors and expand them within Bound
        expand(P, l(N, F/G), Bound, Tree1, Solved, Sol) :-
```

```
        F =< Bound,
        (bagof(M/C, (s(N, M, C), not(member(M, P))), Succ),
        !,                                          % Node N has successors
        succlist(G, Succ, Ts),                      % Make subtrees Ts
        bestf(Ts, F1),                              % f-value of best successor
        expand(P, t(N, F1/G, Ts), Bound, Tree1, Solved, Sol)
        ;
        Solved = never                              % N has no successors - dead end
        ).

% Case 3: non-leaf, f-value less than Bound
% Expand the most promising subtree; depending on results, procedure continue will decide how to proceed
        expand(P, t(N, F/G, [T | Ts]), Bound, Tree1, Solved, Sol):-
          F =< Bound,
          bestf(Ts, BF), min(Bound, BF, Bound1),
          expand([N | P], T, Bound1, T1, Solved1, Sol),
          continue(P, t(N, F/G, [T1 | Ts]), Bound, Tree1, Solved1, Solved, Sol).

% Case 4: non-Leaf with empty subtrees   -   This is a dead end which will never be solved
        expand( _, t(_, _, []), _, _, never, _):- !.

% Case 5: value greater than Bound    -    Tree may not grow
        expand( _, Tree, Bound, Tree, no, _):-
          f(Tree, F), F> Bound.

% continue(Path, Tree, Bound, NewTree, SubtreeSolved, TreeSolved, Solution)
        continue( _, _, _, _, yes, yes, Sol).

        continue(P, t(N, F/G, [T1 | Ts]), Bound, Tree1, no, Solved, Sol):-
          insert(T1, Ts, NTs),
          bestf(NTs, F1),
          expand(P, t(N, F1/G, NTs), Bound, Tree1, Solved, Sol).

        continue(P, t(N, F/G, [_ | Ts]), Bound, Tree1, never, Solved, Sol):-
          bestf(Ts, F1),
          expand(P, t(N, F1/G, Ts), Bound, Tree1, Solved, Sol).

% succlist(G0, [Nodel/Cost1, ...], [l(BestNode. BestF/G), ...]):   make list of search leaves ordered by their f-values
        succlist(_,[],[]).

        succlist(G0, [N/C | NCs], Ts):-
          G is G0 + C,
          h(N, H),                                  % Heuristic term h(N)
          F is G + H,
          succlist(G0, NCs, Ts1),
          insert(l(N, F/G), Ts1, Ts).

% Insert T into list of trees Ts preserving order with respect to f-values
        insert(T,Ts,[T|Ts]):-
          f(T, F), bestf(Ts, F1),
          F =< F1,!.

        insert(T, [T1 | Ts], [T1 | Ts1]):-
          insert(T, Ts, Ts1).
```

```
% Extract f-value
        f(l(_,F/_), F).                 % f-value of a leaf
        f(t(_,F/_,_), F).               % f-value of a tree

        bestf([T|_],F):-                % Best f-value of a list of trees
          f(T, F).

        bestf([], 9999).                % No trees: bad f-value

        min(X,Y,X):-
          X =< Y,!.

        min(X, Y, Y).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Procedimentos específicos para o jogo dos 8

% s( Node, SuccessorNode, Cost)
        s([Empty | Tiles], [Tile | Tiles1], 1):-        % All arc costs are 1
          swap(Empty, Tile, Tiles, Tiles1).             % Swap Empty and Tile in Tiles

        swap(Empty, Tile, [Tile | Ts], [Empty | Ts]):-
          mandist(Empty, Tile, 1).                      % Manhattan distance = 1

        swap(Empty, Tile, [T1 | Ts], [T1 | Ts1]):-
          swap(Empty, Tile, Ts, Ts1).

        mandist(X/Y, X1/Y1, D):-                        % D is Manh. dist. between two squares
          dif(X, X1, Dx),
          dif(Y, Y1, Dy),
          D is Dx + Dy.

        dif(A,B,D):-                                    % D is |A-B|
          D is A-B, D >= 0,!
          ;
          D is B-A.

% Heuristic estimate h is the sum of distances of each tile from its 'home' square plus 3 times 'sequence' score
        h([Empty | Tiles], H):-
          goal([Empty1 | GoalSquares]),
          totdist( Tiles, GoalSquares, D),              % Total distance from home squares
          seq(Tiles, S),                                % Sequence score
          H is D + 3*S.

        totdist([], [], 0).

        totdist([Tile | Tiles], [Square | Squares], D):-
          mandist(Tile, Square, D1),
          totdist(Tiles, Squares, D2),
          D is D1 + D2.

% seq(TilePositions, Score): sequence score
        seq([First | OtherTiles], S):-
          seq([First | OtherTiles], First, S).
```

```
seq([Tile1, Tile2 | Tiles], First, S):-
  score(Tile1, Tile2, S1),
  seq( [Tile2 | Tiles], First, S2),
  S is S1 + S2.

seq([Last], First, S):-
  score(Last, First, S).

score(2/2, _, 1).                          % Tile in centre scores 1

score(1/3, 2/3, 0):- !.                    % Proper successor scores 0
score(2/3, 3/3, 0):- !.
score(3/3, 3/2, 0):- !.
score(3/2, 3/1, 0):- !.
score(3/1, 2/1, 0):- !.
score(2/1, 1/1, 0):- !.
score(1/1, 1/2, 0):- !.
score(1/2, 1/3, 0):- !.

score(_, _, 2).                            % Tiles out of sequence score 2

goal([2/2,1/3,2/3,3/3,3/2,3/1,2/1,1/1,1/2]).      % Goal squares for tiles

% Display a solution path as a list of board positions
showsol([]).

showsol([P | L]):-
  showsol(L),
  nl, write('---'),
  showpos(P).

% Display a board position
showpos([S0,S1,S2,S3,S4,S5,S6,S7,S8]):-
  member(Y, [3,2,1]),                      % Order of Y-coordinates
  nl, member(X, [1,2,3]),                  % Order of X-coordinates
  member(Tile-X/Y,                         % Tile on square X/Y
       [0-S0,1-S1,2-S2,3-S3,4-S4,5-S5,6-S6,7-S7,8-S8]),
  write(Tile),
  fail                                     % Backtrack to next square
  ;
  true.                                    % All squares done
```

3.3.4. Têmpera Simulada ("Simulated Annealing")

Método de busca inspirado no processo de temperar metais: inicia com uma temperatura elevada, permitindo transições de alto nível de energia, depois, gradualmente, a temperatura é resfriada até alcançar um estado sólido, onde apenas transições de baixo nível de energia são permitidas. O objetivo é alcançar uma estrutura molecular com um mínimo de energia.

Neste processo, as substâncias físicas tendem a acomodar-se em configurações de baixa energia, mas sempre existe uma certa probabilidade de ocorrer uma transição para um estado de maior energia, calculada pela fórmula: $p = e^{(-\Delta E/kT)}$ onde,

$\Delta E = |E' - E^2|$ = mudança positiva de energia;

k = constante de Boltzmann, escreve a relação entre as unidades de energia e temperatura;

T = temperatura.

No processo de resolução de problemas, os valores energia e temperatura não têm o mesmo sentido que na física, assim k deixa de ser necessário e o cálculo da probabilidade fica: $p = e^{(-\Delta E/T)}$. A diferença de energia é medida pela diferença da aplicação da *função-objetivo* (função heurística) no estado corrente e no novo estado.

A mudança de temperatura no decorrer do processo chama-se "cronograma de têmpera" e tem três componentes:

1. Valor inicial
2. Critério(s) para decidir quando a temperatura deve ser reduzida.
3. Valor da redução (quanto)

A maneira como a temperatura diminui influencia no resultado do processo, se T esfriar muito rápido, provavelmente será encontrado um mínimo local e não global, se T esfriar muito lentamente haverá muita perda de tempo.

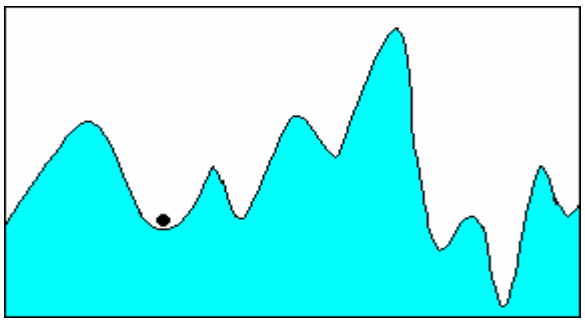


Figura 21 Analogia: dispositivo onde uma esfera fica encerrada em uma caixa com vidro na face da frente e na face de trás e um elemento sólido formando uma curva. Quando o dispositivo fica na vertical a esfera tende a acomodar-se ao fundo de um vale. Uma temperatura alta equivale a agitar o dispositivo, permitindo que a esfera suba colinas.

Algoritmo:

1. Estado Corrente ← Estado Inicial.
2. Melhor ← Estado Corrente
3. T ← cronograma (inicio)
4. Repita até que Estado Corrente seja a solução ou não haja mudança no Estado Corrente.
   a) Escolha um operador que ainda não tenha sido aplicado ao Estado Corrente e aplique-o para produzir um Novo Estado
   b) Analise o Novo Estado
      i) Se for a solução (um estado meta) retorne-o e encerre.
      ii) Se for melhor que Estado Corrente
      então faça
            Estado Corrente ← Novo Estado
            Se Novo Estado for melhor que Melhor
            então Melhor ← Novo Estado
      senão faça
            $\Delta E \leftarrow f(\text{Estado Corrente}) - f(\text{Novo Estado})$
            $p \leftarrow e^{(-\Delta E/T)}$
            Se p é maior que número aleatório no intervalo [0; 1]
            então Estado Corrente ← Novo Estado
   c) T ← cronograma
5. Retorne Melhor como resposta do problema.

3.3.5. Técnicas Evolutivas

É um conjunto de técnicas de otimização inspiradas na teoria da evolução. A principal idéia da teoria da evolução é a seleção natural, onde uma população de indivíduos evolui durante gerações devido à maior probabilidade daqueles mais aptos de terem uma prole mais numerosa, resultando em indivíduos cada vez melhores. As técnicas estabelecem uma analogia onde um indivíduo é visto como sendo uma solução e esta solução é melhorada até chegar a uma solução ótima ou um outro critério de parada seja satisfeito.

Esta abordagem remonta o início da computação (1960), mas o interesse na sua utilização cresceu muito nos meados da década de 80. As principais técnicas que foram desenvolvidas são: programação evolucionista, estratégia evolucionista, sistemas classificadores, algoritmos genéticos e programação genética.

A técnica mais conhecida são os algoritmos genéticos, onde a analogia com os sistemas biológicos é mais forte. Cada

hipótese de solução (indivíduo) é representada por uma cadeia binária de tamanho fixo, como se fosse a cadeia genética do indivíduo.

A avaliação de um indivíduo é feita com base na aptidão do seu fenótipo.

*** Ver anotações de aula

4. Sistemas Baseados em Conhecimento

A evolução da pesquisa em resolução de problemas pode ser dividida em três fases distintas. Durante a primeira fase procurava-se imitar a capacidade do homem de resolver problemas, ou seja, buscava-se compreender e reproduzir a capacidade de raciocínio e de representação de problemas do homem. Um exemplo de sistema que foi desenvolvido nesta fase é o GPS (General Problem Solver). O grande problema desta abordagem é a explosão combinatória quando se tenta resolver problemas mais complexos.

A segunda fase surgiu a partir do reconhecimento de que a habilidade dedutiva do homem resulta menos de sua capacidade de raciocínio do que de sua habilidade de armazenar experiências anteriores e adaptá-las a novas situações. Nesta fase passou-se a utilizar os sistemas de produção e surgiram os sistemas baseados em conhecimento (cognitivos). Nos sistemas baseados em conhecimento a ênfase deixa de ser o método de busca e a forma de representar o conhecimento passa a ter um papel fundamental.

A terceira fase iniciou-se a partir da percepção de que o conhecimento geral é muito extenso e pouco útil para problemas específicos. Para permitir que os problemas sejam tratáveis em tempo hábil surgem os sistemas especialistas. Quanto maior a quantidade e a qualidade de conhecimento específico (relativo ao problema) embutido nestes sistemas, melhor a performance do sistema.

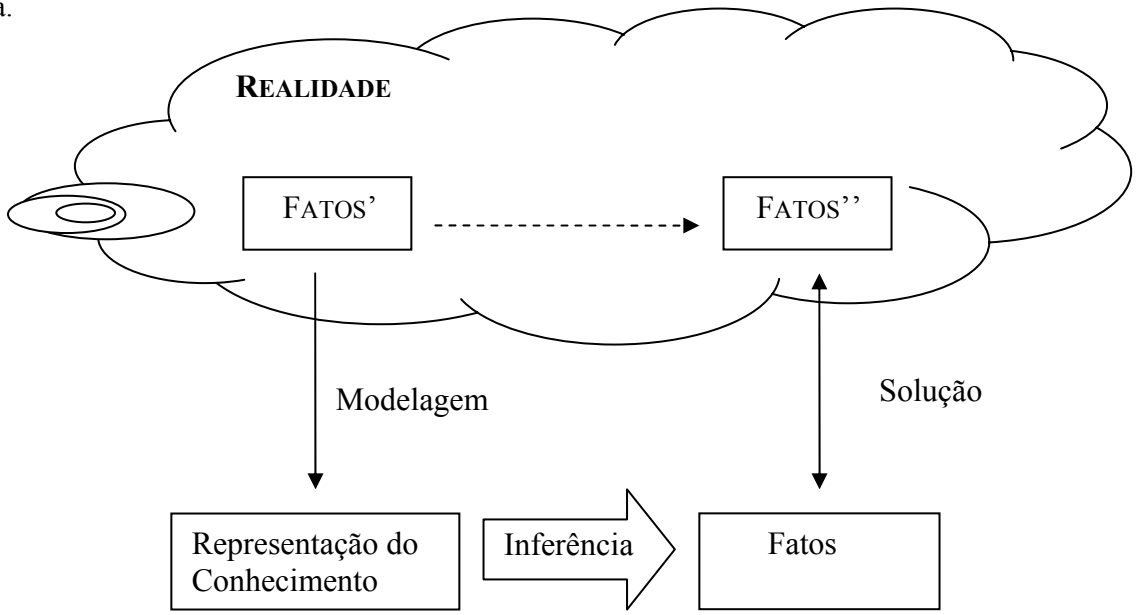


Figura 22 - Modelagem da realidade

Conforme está esquematizado na Figura 22, uma parte da realidade, correspondendo a um problema, pode ser modelada utilizando-se uma forma (linguagem) de representação do conhecimento e a seguir pode-se inferir "novos" fatos que podem conter a solução para o problema modelado. Os possíveis tipos de inferência que podem ser efetuados dependem da forma de representação do conhecimento.

4.1. Propriedades de Sistemas de Representação do Conhecimento

- Adequação Representacional - Permite representar todos os tipos de conhecimento necessários ao domínio do problema que se quer solucionar.
- Adequação Inferencial - Permite manipular estruturas representacionais gerando "novos" conhecimentos, dentre os quais, possíveis soluções.
- Eficácia Inferencial - Permite representar meta-conhecimento visando melhorar os mecanismos de inferência.
- Eficácia de Comunicação - Permite facilmente entender o conhecimento armazenado, adquirir novos conhecimentos e fornecer "explicações".

4.2. Formas de Representação do Conhecimento

- Sistemas de Produção
- Sistemas Baseados em Regras

- Lógica
  - Convencional
  - Não-monotônica
  - Difusa ou nebulosa (“fuzzy”)
  - Temporal
  - Senso-comum

  ...
- Raciocínio Probabilístico
- Árvores de Decisão
- Redes Semânticas
- Quadros (“frames”)
- Roteiros (“scripts”)

4.3. Sistemas de Produção (SP)

SP foram inicialmente propostos por Post em 1943 para modelar o processamento da informação a partir da noção estímulo/resposta, simulando os processos mentais de raciocínio e utilização do conhecimento. Desde então SP tiveram um grande desenvolvimento e atualmente é um formalismo que descreve um grupo de sistemas baseados em pares condição-ação denominados regras de produção.

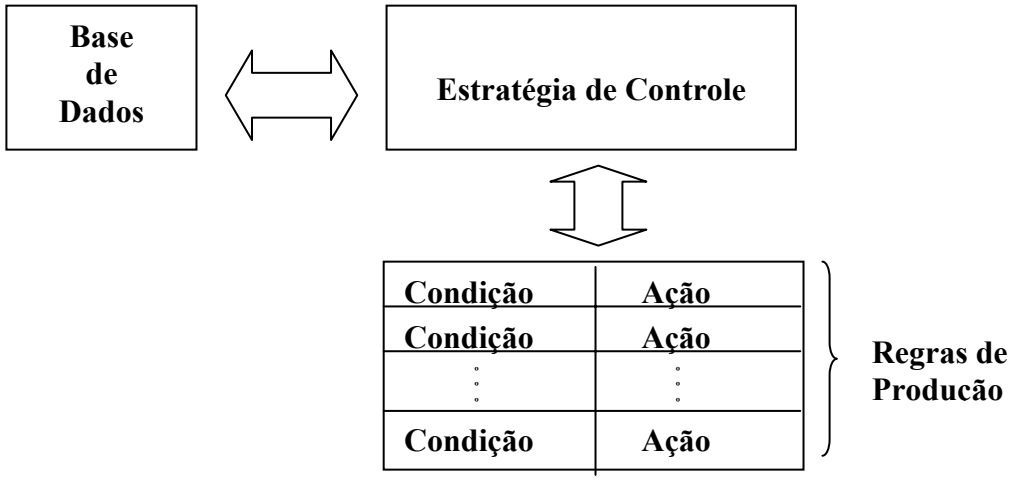


Figura 23 - Arquitetura básica de um SP

A Figura 23 apresenta a arquitetura de um SP com os seus principais componentes:

- Base de Dados: também chamado, memória de trabalho ou contexto, contém dados informados pelo usuário sobre o problema ou gerados pelas próprias regras durante a execução (estado).
- Regras de Produção: conjunto de regras que operam sobre a base de dados, modificando-a.
- Estratégia de Controle: também denominado, interpretador de regras, é o módulo responsável pela execução do sistema e pela identificação da solução sinalizando quando o sistema deve parar. Este processo possui 3 fases:
  1) casamento (“matching”) – seleciona as regras aplicáveis comparando as condições (lado esquerdo) das regras com o estado da Base de Dados, as regras que casarem são selecionadas;
  2) resolução de conflitos – apenas uma regra de cada vez pode ser ativada e quando mais de uma regra é aplicável, ocorre um conflito, sendo que a forma mais óbvia de se resolver um conflito é ativar a primeira regra que foi satisfeita;
  3) ação – ativa as regras executando as ações (lado direito) das mesmas, o que normalmente altera o Banco de Dados.

Características de SP

- Monotônico – a aplicação de uma determinada regra nunca impede a posterior aplicação de outra regra que também poderia ter sido aplicada quando a primeira regra foi selecionada;

- Parcialmente Comutativo – se a aplicação de uma seqüência particular de regras transforma o estado x no estado y, então qualquer permutação dessas regras que seja viável também transforma o estado x no estado y;
- Comutativo - é um sistema que é monotônico e parcialmente comutativo.

Exemplo 1: SP para representar um numero inteiro ($x \leq 40$) em número romano.
Base de Dados: variável x armazenando inicialmente o valor a ser convertido; o resultado será impresso.
Estratégia de Controle:
- testar as condições das regras em ordem arbitrária até que uma seja satisfeita;
- executar a ação correspondente;
- repetir indefinidamente.

Regras de Produção:
1) $(x = 0) \rightarrow$ ("avise" o usuário) e (leia(x))
2) $(x > 39) \rightarrow$ ("avise" que x é muito grande)
3) $(10 \leq x \leq 39) \rightarrow$ (imprima "X") e (x := x – 10)
4) $(x = 9) \rightarrow$ (imprima "IX")
5) $(5 \leq x \leq 8) \rightarrow$ (imprima "V") e (x := x – 5)
6) $(x = 4) \rightarrow$ (imprima "IV")
7) $(1 \leq x \leq 3) \rightarrow$ (imprima "I") e (x := x – 1)

Exemplo 2: SP para solucionar o problema das jarras.

O objetivo deste problema é, tendo-se duas jarras, uma com capacidade de 3 litros e a outra de 4 litros, chegar-se a um volume de dois litros na jarra de 3 litros. Pode-se colocar ou retirar água das jarras, porém os únicos volumes conhecidos são os das capacidades das jarras.

Base de Dados: Par de valores (x, y), onde o primeiro elemento representa o volume de água contido na jarra de 4 litros e o segundo elemento representa o volume de água contido na jarra de 3 litros. Assim, o estado inicial (as duas jarras vazias) é representado como (0, 0) e o estado final como (x, 2).

Regras de Produção:
1) (x, y) | x < 4 → (4, y)
2) (x, y) | y < 3 → (x, 3)
3) (x, y) | x > 0 → (x-d, y)
4) (x, y) | y > 0 → (x, y-d)
5) (x, y) | x > 0 → (0, y)
6) (x, y) | y > 0 → (x, 0)
7) (x, y) | x+y >= 4 e y > 0 → (4, y-(4-x))
8) (x, y) | x+y >= 3 e x > 0 → (x-(3-y), 3)
9) (x, y) | x+y <= 4 e y > 0 → (x+y, 0)
10) (x, y) | x+y <= 3 e x > 0 → (0, x+y)
11) (0, 2) → (2, 0)
12) (2, y) → (0, y)

⋮

Deve ser definido um número mínimo de regras que modele adequadamente o problema, permitindo que se chegue à solução. Além disso, é desejável inserir regras adicionais, mais especializadas, que representem "atalhos" na procura da solução.

4.4. Sistemas Baseados em Regras

Os sistemas baseados em regras surgiram a partir dos Sistemas de Produção e evoluíram utilizando resultados das pesquisas em lógica. As regras têm a forma geral ***se*** <*condições*> ***então*** <*conclusões*> e os mecanismos de inferência utilizados para gerarem novos fatos são baseados na lógica.

A maior aplicação dos sistemas baseados em regras são os Sistemas Especialistas (SE). Evidentemente os SE podem utilizar qualquer uma das formas de representação do conhecimento citadas anteriormente, porém a grande maioria deles utiliza regras. Isto acontece porque a maioria do conhecimento utilizado em problemas de IA é diretamente representável em expressões gerais de implicação, além de serem facilmente compreensíveis pelos usuários.

No restante do capítulo os sistemas baseados em regras são aprofundados através do estudo dos SE, pois estes últimos representam um dos maiores marcos comercias na exploração da Inteligência Artificial.

5. Introdução aos Sistemas Especialistas

Sistemas Especialistas (SE) são sistemas de Inteligência Artificial, baseados em conhecimento, que emulam um especialista humano na resolução de um problema significativo em um domínio específico.

Assim, os SE solucionam problemas que normalmente são solucionados por especialistas humanos. Para solucionar tais problemas, os sistemas especialistas precisam acessar uma substancial base de conhecimento do domínio da aplicação, que precisa ser criada de modo mais eficiente possível. Eles também precisam explorar um ou mais mecanismos de raciocínio, para aplicar aos problemas que têm diante de si. Depois eles precisam de um mecanismo para explicar o que fizeram aos usuários que dele dependem.

O termo *especialista* freqüentemente leva as pessoas a terem expectativas irrealistas do desempenho de um SE. É importante lembrar que o termo *especialista* originalmente representa uma limitação na capacidade de representação e armazenamento de conhecimento do sistema. Na verdade, um SE trata de um determinado conhecimento específico sobre um domínio de problema, ao contrário de um *solucionador geral de problemas*. Quanto mais geral a classe de problemas que podem ser resolvidos por um sistema, mais insatisfatória será a solução produzida.

Os problemas com os quais lidam os sistemas especialistas são altamente diversificados. Há questões gerais que surgem em vários domínios. Os sistemas especialistas que fizeram mais sucesso comercialmente estão na classe de problemas em que dado um conjunto de fatos observados, o sistema infere possíveis problemas, detecta resultados relevantes, apresenta alternativas de projeto, etc.

Características de SE

- Possuem conhecimento especializado em alta qualidade e quantidade;
- O conhecimento pode ser incompleto, subjetivo e inexato;
- O Banco de Conhecimento é independente da Estrutura de Controle;
- Explicam o raciocínio utilizado na resolução do problema;
- Incluem tratamento de incerteza;
- Podem adquirir novos conhecimentos.

Tipos de SE

- Classificação
- Diagnose - infere mal-funções.
- Interpretação - compreende situações.
- Predição - infere conseqüências de situações.
- Projeto - configura objetos sob restrições.
- Planejamento - projeta seqüência de ações.
- Depuração - prescreve "remédios" para mal-funções.
- Reparo - executa um plano para administrar um "remédio" prescrito.
- Instrução - diagnose, depuração e reparo para deficiências de estudantes.
- Controle - interpretação, predição, reparo, e monitoramento de comportamento de sistemas.

O SE e seu Ambiente

A Figura 24 mostra o ambiente de um SE, onde há dois momentos distintos: a construção e a utilização. O profissional encarregado de construir um SE é o engenheiro de conhecimento, o qual deve dominar as ferramentas de desenvolvimento e as linguagens de representação de conhecimento. O conhecimento pode ser privado, obtido através de entrevistas com especialistas no assunto relativo ao problema, ou público, obtido pela consulta em livros e revistas especializadas.

Quando o SE estiver em produção, ele pode ser utilizado para resolver problemas de usuários não especialistas no assunto.

5.1. Componentes Básicos de um SE

Existe um consenso sobre a arquitetura geral de um sistema especialista. Quase todos eles podem ser separados em três componentes distintos: a *Base de Conhecimento,* o *Motor de Inferência (Inference Engine) e a Interface com* o *Usuário.*

Base de Conhecimento

Consiste na parte central de um SE. É a representação do conhecimento no domínio no problema em questão, geralmente extraída de um ou vários especialistas, de forma declarativa, e livre de detalhes de controle e implementação. Idealmente ela é composta de declarações, em algum formalismo de representação de conhecimento disponível, descrevendo o domínio da aplicação.

Memória de Trabalho

Armazena as informações fornecidas pelo usuário e todo o conhecimento inferido pelo sistema durante uma consulta. Ao final da consulta esta informações são apagadas.

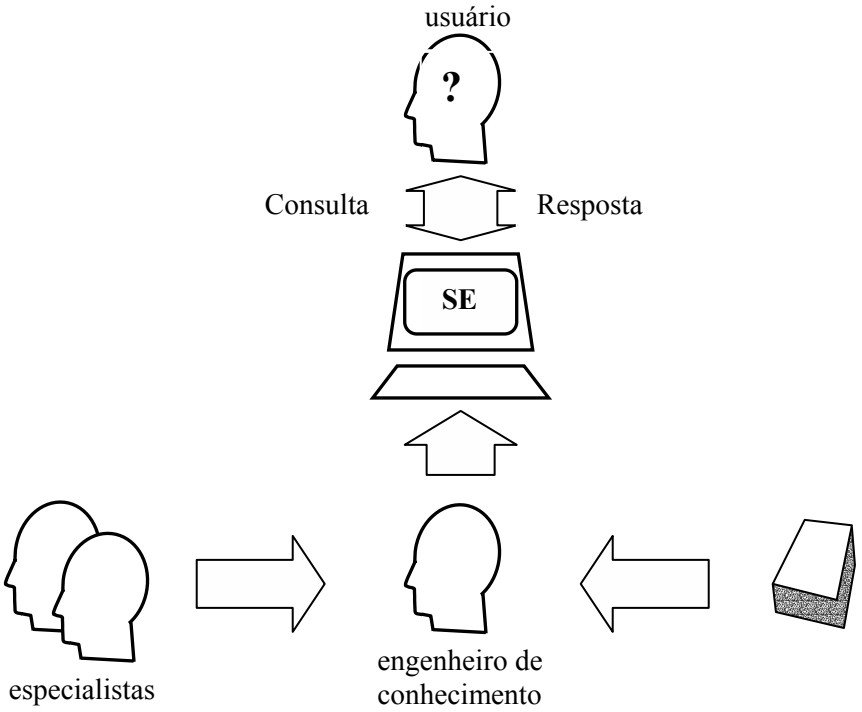


Figura 24 - Ambiente de um SE

Motor de Inferência

O *motor de inferência,* também chamado *máquina de* inferência, é responsável pela manipulação da base de conhecimento durante a resolução de problemas. É chamado de motor de inferência porque ele usa o conhecimento da base e os fatos relativos a uma determinada consulta para obter conclusões. A natureza do motor de inferência irá depender do formalismo utilizado para representar a base de conhecimento e da estratégia de solução de problemas considerada apropriada pelo projetista do sistema. Por exemplo, se o formalismo de representação escolhido for regras de *produção,* o motor de inferência irá necessariamente tomar a forma de um interpretador de regras.

A arquitetura do motor de inferência está intimamente relacionada com a linguagem de representação do conhecimento e inclui os elementos: Mecanismo de Inferência, Estratégia de Controle, Métodos de tratamento da Incerteza e Métodos de tratamento de Conhecimento Incompleto, os quais serão detalhados mais adiante.

Interface com o Usuário

Todo sistema especialista é interativo. e precisa de um componente para gerenciar a interação entre o usuário e o sistema A interação básica numa sessão de uso de um SE consiste no sistema perguntar questões relevantes. apresentar conselhos, respostas e prover explicações requeridas pelo usuário

Dois tipos de explicação que são freqüentemente fornecidas pelo SE são aquelas a perguntas do tipo *Por que?,* e explicações de como o sistema chegou a determinada conclusão, ou seja Como?

5.2. Arquitetura do Motor de Inferência

Mecanismo de Inferência

O mecanismo de inferência gera novas conclusões a partir da manipulação do conhecimento existente na base de conhecimento e na memória de trabalho. O mecanismo depende da forma de representação de conhecimento. Em sistemas baseados em regras o mecanismo encontrado com mais freqüência é o "modus ponens": se as premissas de uma regra são verdadeiras, então é pode-se acreditar que as suas conclusões também o são, assim as conclusões podem ser acrescentadas no conhecimento existente. Conforme o exemplo mostrado na Figura 25, o fato B é derivado a partir do fato A e da Regra A $\rightarrow$ B. Uma das vantagens do "modus ponens" é a sua simplicidade, entretanto é importante ressaltar que ele não conclui todas as inferências válidas.

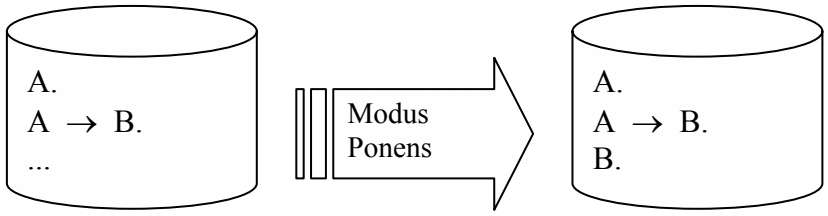


Figura 25 - Exemplo de aplicação do "modus ponens"

Em formalismos de representação do conhecimento orientados a objetos um dos mecanismos de inferência utilizados é a herança. No capítulo sobre formas de representação conhecimento são estudadas outros mecanismos de inferência.

Estratégia de Controle

É composto por vários elementos:

- **Modo de seleção de "peças" de conhecimento** (regras, quadros, roteiros, nodos, ...) - é feito através de casamento de padrões.
- **Direção do raciocínio** - é a direção em que é feita a busca em um espaço de soluções. No caso de regras é a direção como é feito o encadeamento das regras:
  - Encadeamento progressivo (encadeamento para frente, "forward", "data-driven", "bottom-up")
    As regras da base de conhecimento são selecionadas com base no conhecimento existente (fatos da base de conhecimento, informações do usuário e conhecimento já inferido), comparando-se o lado esquerdo das regras (condições), e ativadas para gerar novos fatos até encontrar a solução. Equivale à busca em espaço de estados, onde o estado é o banco de conhecimento. A Figura 26 mostra um exemplo de encadeamento progressivo. Os números associados aos arcos da árvore de pesquisa correspondem à aplicação das regras da base de conhecimento. No início a regra 1 não é aplicável, talvez será no decorrer da pesquisa.
  - Encadeamento regressivo (encadeamento para trás, "backward", "goal-directed", "top-down")
    As regras da base de conhecimento são selecionadas a partir dos objetivos, comparando-se o lado direito das regras (conclusões), e ativadas para gerar novos objetivos até encontrar a solução. Equivale à busca por redução de problemas, onde os problemas (objetivos) são "quebrados" em subproblemas (novos objetivos). A Figura 27 apresenta um exemplo de encadeamento regressivo, onde os nodos da árvore de pesquisa em cinza são os problemas primitivos, já resolvidos, e os nodos em branco são problemas em aberto, ainda não resolvidos.

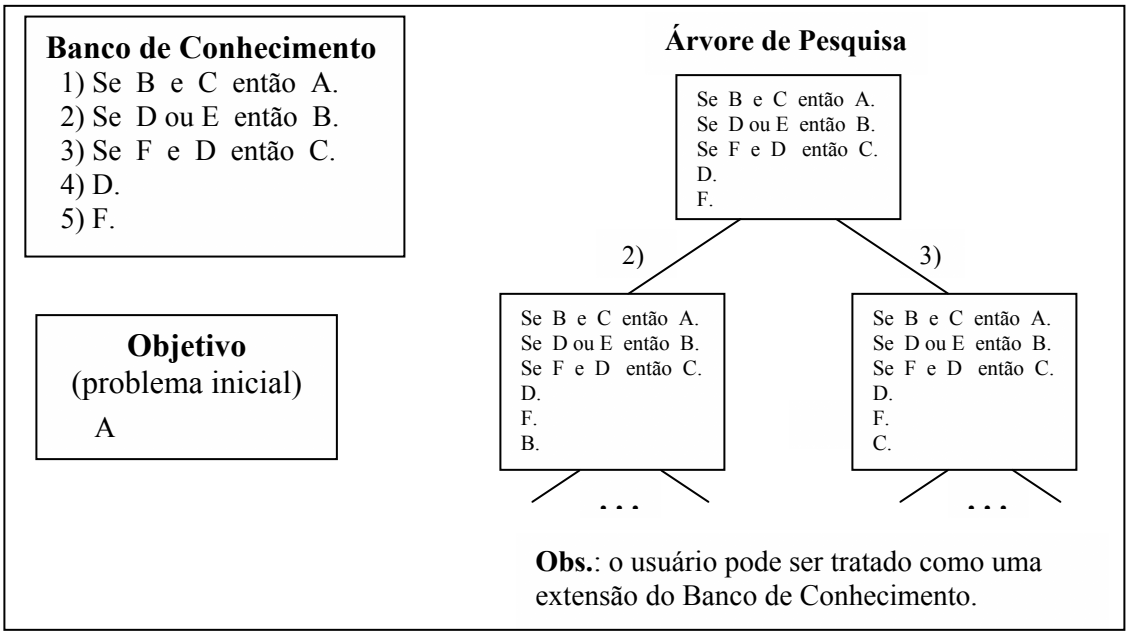


Figura 26 - Exemplo de encadeamento progressivo.

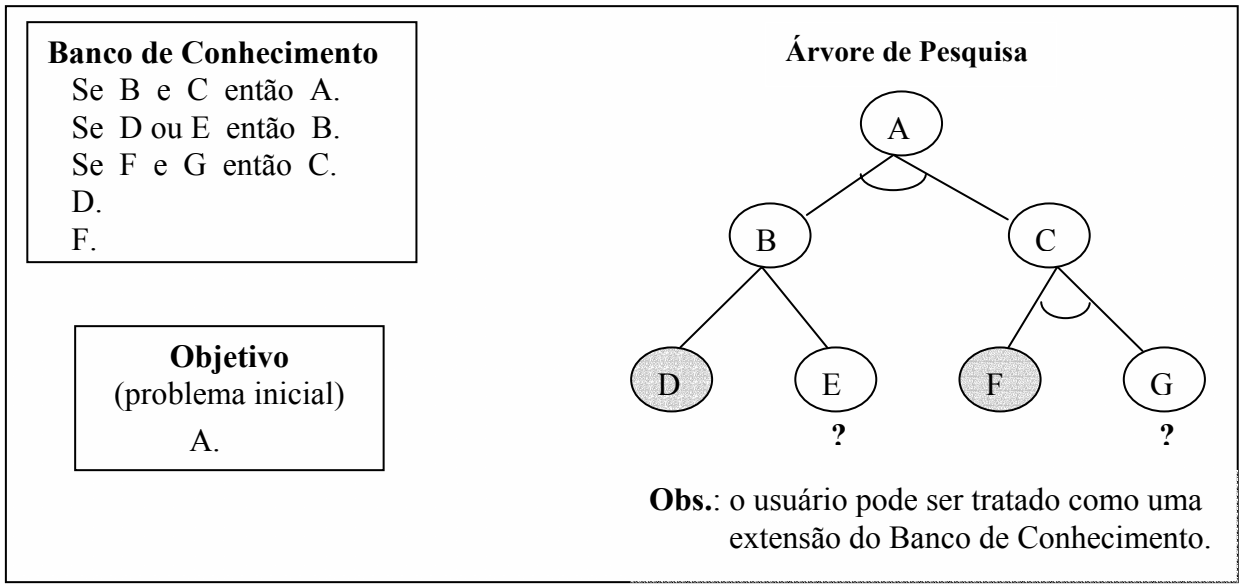


Figura 27 - Exemplo de encadeamento regressivo.

A escolha da direção do raciocínio depende de vários aspectos:

- quantidade de fatos iniciais x quantidade de conclusões alternativas
- quantidade média de regras elegíveis a cada ciclo
- necessidade de processamento interativo
- necessidade de justificar o resultado
- natureza do problema:
  - deduzir conseqüências a partir de fatos;
  - comprovar hipóteses buscando indícios.

- **Estratégia de Pesquisa** - pode ser:
  - em profundidade;
  - em extensão;
  - heurística.

- **Utilização de Meta-conhecimento** - é o conhecimento do especialista de como deve ser utilizado o conhecimento, influindo no modo como o conhecimento deve ser tratado em alguma parte do raciocínio. Exemplo: ativação de um conjunto de regras e fatos específicos sob uma certa condição.

- **Análise de diferença** - identifica regras que diminuem a diferença entre o conhecimento atual e o objetivo.

Métodos de tratamento de Conhecimento Incompleto

O tratamento de conhecimento incompleto pode ser feito com a introdução da noção de valores "default" e com operações lógicas com o desconhecido, conforme a tabela verdade abaixo.

A	B	¬A	A e B	A ou B
F	F	V	F	F
F	V	V	F	V
F	D	V	F	D
V	F	F	F	V
V	V	F	V	V
V	D	F	D	V
D	F	D	F	D
D	V	D	D	V
D	D	D	D	D

Figura 28 - Tabela verdade incluindo o valor desconhecido (D).

Métodos de tratamento da Incerteza

O tratamento da incerteza lida com o raciocínio inexato e pode ser feito através da utilização de valores intermediários entre os valores verdadeiro e falso. A forma clássica de implementação é feita pela quantificação do grau de certeza do conhecimento, conforme o exemplo apresentado na Figura 29.

Fator de Certeza	Significado
100	Definitivamente certo
80	Quase Certo
60	Provável
30	Há evidências
20	Ignorado
0	
-20	
-40	Há evidências contra
-60	Provavelmente não
-80	Quase certo que não
-100	Definitivamente não

Figura 29 - Escala do grau de certeza do conhecimento no MYCYN

Forma de Inferência

- Fator de certeza (FC) da conclusão de uma regra = (FC(conjunto de antecedentes) * FC(regra)) / 100
- FC(conjunto de antecedentes) = min(FCs dos antecedentes) se antecedentes conjuntivos (unidos pelo "E") ou
  = max(FCs dos antecedentes) se antecedentes disjuntivos (unidos pelo "OU")
- Várias regras inferem a mesma conclusão - abaixo são apresentadas três formas de cálculo com um exemplo na figura Figura 30.
  a) FC(conclusão) = max(FCs das conclusões das regras)
  b) FC(conclusão) = média(FCs das conclusões das regras)
  c) FC(conclusão) = x + ((100 – x) * y / 100) se ambos (x e y) são positivos (técnicas probabilísticas)

Banco de Conhecimento

1. Se A então B FC(80).
2. Se A e C então B FC(70).
3. Se A ou D então B FC(60).
4. B FC(60).
5. B FC(30).
6. B FC(40).

Resultado do cálculo do FC das Conclusões

Regra	FC(B)
1.	48
2.	21
3.	36

Formas de tratar a situação

Forma	FC(B)
a) Max	48
b) Média	35
c) Probabilidade	74

Observação

Como pode ser observado, as diferentes formas de tratar a situação apresentam uma grande variação.

Figura 30 - Exemplo de cálculo quando várias regras inferem a mesma conclusão.

5.3. Alguns Exemplos de Sistemas Especialistas

Os sistemas especialistas representam uma área onde a IA conseguiu bons resultados. Alguns sistemas fizeram, e continuam fazendo, muito sucesso comercialmente, resolvendo de maneira bastante satisfatória os problemas para os quais eles foram criados.

MYCIN

O *MYCIN* foi um dos primeiros e mais conhecidos sistemas especialistas e é considerado um marco na IA. Foi criado inicialmente para diagnosticar e sugerir tratamento para doenças no sangue e desenvolvido na Universidade de Stanford, a partir de 1972. A principio era essencialmente um projeto acadêmico, mas influenciou enormemente a maioria dos sistemas especialistas que estavam por vir.

O *MYCIN* usa o encadeamento para trás *(backward)* para descobrir que organismos estavam presentes no sangue; depois ele usava o encadeamento para frente *(forward)* para raciocinar sobre o regime de tratamento.

O *MYCIN* tenta atingir seu projeto de recomendar uma terapia para um determinado paciente encontrando primeiro a causa da doença do paciente. Para atingir objetivos de nível mais alto do diagnóstico, ele procura regras cujos lados direitos sugiram doenças. Depois ele usa os lados esquerdos dessas regras (as precondições) para definir sub-objetivos cujos êxitos permitirão que as regras sejam invocadas. Esses sub-objetivos são novamente casados com as regras, e suas precondições são usadas para estabelecer mais sub-objetivos.

PROSPECTOR

O *PROSPECTOR* é um sistema especialista que dá conselhos sobre a exploração de minerais. Foi criado no final da década de 70 pela *SRI International*. É um sistema digno de ser mencionado pelas seguintes razões: primeiro, ele se propõe a resolver problemas de extrema dificuldade, até mesmo os melhores especialistas disponíveis (geólogos principalmente) não executam essa tarefa com um alto grau de certeza; segundo, ele ilustra uma abordagem bastante diferente para sistemas especialistas (manipulação de probabilidades).

As regras no *PROSPECTOR* são do tipo

Se: magnetita ou pinta em forma disseminada ou em veios está presente
Então: (2,-4) há mineralização e textura favoráveis ao estágio propilítico.

No *PROSPECTOR,* cada regra contém duas estimativas de confiança. que variam de -5 a 5. A primeira indica até que ponto a presença da evidência descrita na condição da regra sugere a validade da conclusão da regra. Na regra do *PROSPECTOR* mencionada, o número 2 indica que a presença da evidência é *suavemente encorajadora.* A segunda estimativa de confiança mede até que ponto a evidência é necessária para a validade da conclusão, ou em outras palavras, até que ponto a falta da evidência indica que a conclusão não é válida. No exemplo anterior, o número -4 indica que a ausência de evidência é *fortemente desencorajadora* para a conclusão.

XCON ou *R1*

Enquanto o *MYCIN* influenciava bastante o desenvolvimento de sistemas especialistas na área acadêmica, um outro sistema clássico, o *Ri,* depois chamado *XCON,* despertou mais o interesse no mundo comercial. Trata-se de um sistema que configura os sistemas *DEC VAX.* Suas regras são do tipo:

Se: o contexto mais atual é a distribuição de dispositivos barramento *massbus, E*
há um *drive* de disco de porta única que ainda não foi atribuído a um barramento *massbus, E*
não há nenhum *drive* de disco de porta única sem designação*, E*
o número de dispositivos que cada barramento *massbus* deve suportar é conhecido, *E*
há um barramento *massbus* ao qual foi atribuído pelo menos um *drive* de disco e que deve suportar *drives* adicionais*, E*
o tipo de cabo necessário para conectar o *drive* de disco ao dispositivo anteriormente no barramento *massbus* é conhecido
Então: atribua o *drive* ao barramento *massbus*

DENDRAL

O META-DENDRAL, criado em 1978, foi o primeiro programa a usar técnicas de aprendizagem para construir automaticamente regras para um sistema especialista. Ele criava regras para serem usadas pelo DENDRAL, cuja tarefa era determinar a estrutura de compostos químicos complexos. O META-DENDRAL era capaz de induzir suas regras com base em um conjunto de dados sobre espectrometria de massa: desse modo, ele conseguia identificar as estruturas moleculares com precisão bastante alta.

5.4. Ferramentas de Desenvolvimento de Sistemas Especialistas

Inicialmente, cada sistema especialista era criado a partir do nada, em geral em E/SE. Mas, depois de vários sistemas terem sido desenvolvidos, ficou claro que esses sistemas tinham muito em comum. Em particular. devido ao fato de os sistemas serem construídos como um conjunto de representações declarativas (em sua maioria, regras) combinadas com um interpretador dessas representações, era possível separar o interpretador do conhecimento específico do domínio da aplicação e assim criar uni sistema que podia ser usado para elaborar novos sistemas especialistas através da adição de novos conhecimentos, correspondentes ao novo domínio do problema. Os interpretadores resultantes são chamados de *shell* (ferramenta). No início, um exemplo de ferramenta que teve grande influência é o *EMYCIN* (*Empty MYCIN),* derivado do *MYCIN.*

Existem atualmente vários novas ferramentas comercialmente disponíveis, que servem de base para muitos dos sistemas especialistas que estão sendo desenvolvidos correntemente. Com essas ferramentas, a representação do conhecimento e o raciocínio são muito mais flexíveis do que nos primeiros sistemas especialistas criados. Elas tipicamente suportam regras, quadros*,* sistemas de manutenção da verdade e uma série de outros mecanismos de raciocínio.

As primeiras ferramentas de sistemas especialistas ofereciam mecanismos para a representação do conhecimento, raciocínio e explicações. Mais tarde, foram acrescentadas ferramentas para a aquisição de conhecimento. Mas, com o aumento das experiências com esses sistemas para solucionar problemas do mundo real, ficou claro que as ferramentas dos sistemas especialistas precisavam fazer alguma coisa a mais. Eles precisavam, por exemplo, facilitar a integração dos sistemas especialistas com outros tipos de programas. acessar bancos de dados de corporações; e esse acesso precisa ser controlado como em outros sistemas. Eles em geral estão embutidos em programas aplicativos maiores, que usam praticamente técnicas de programação convencional. Então, uma das características importantes que uma ferramenta precisa ter é uma interface entre o sistema especialista, escrita com a ferramenta e que seja fácil de usar, e um ambiente de programação maior e provavelmente mais convencional.

A Figura 31 classifica os diferentes *software* que podem ser utilizados no desenvolvimento de SE. As linguagens de alto nível são orientadas para a manipulação simbólica, dando flexibilidade e permitindo ajustar o SE às particularidades de cada domínio de problemas, entretanto exige grande experiência em computação. Os ambientes são um meio termo entre as linguagens de alto nível e as ferramentas, e oferecem código testado e depurado para implementar coisas úteis na construção de SE. As ferramentas já possuem uma forma de representação de conhecimento e uma máquina de inferência bem definidos, assim são fáceis de utilizar e permitem uma grande velocidade de desenvolvimento, permitindo que o pessoal envolvido se concentre na montagem da base de conhecimento.

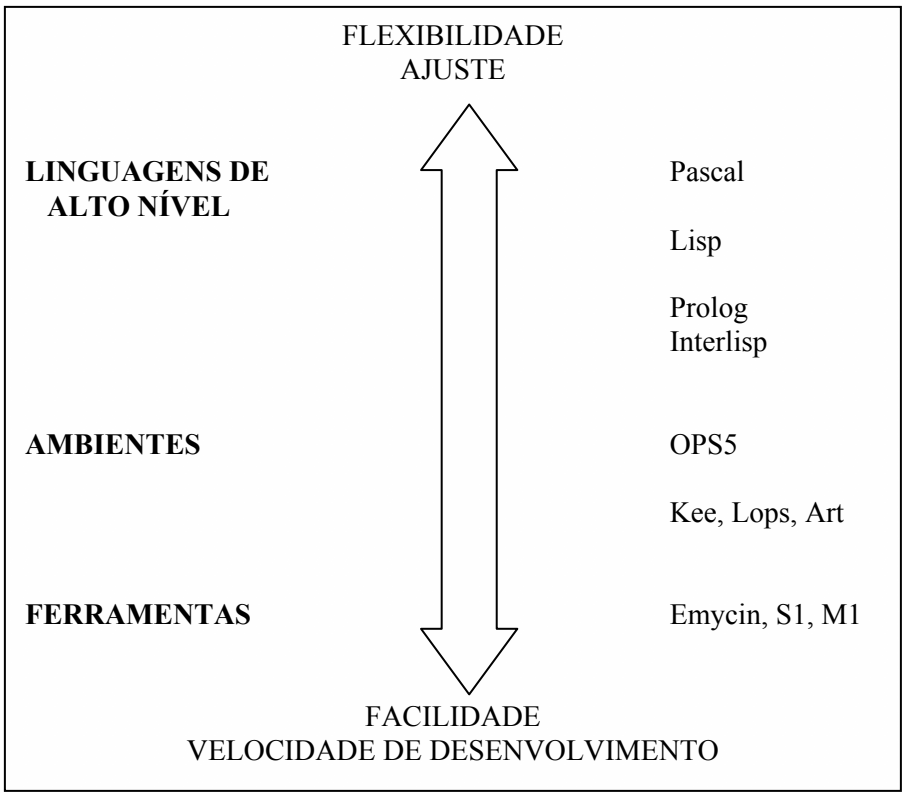


Figura 31 - *Software* para o desenvolvimento de SE

5.5. Aquisição de Conhecimento

Como são construídos os sistemas especialistas? Tipicamente, um engenheiro do conhecimento entrevista um especialista no domínio da aplicação para tentar extrair o conhecimento especialista, que é então traduzido em regras. Depois que o sistema inicial estiver pronto. ele precisa ser iterativamente refinado até aproximar-se do nível de desempenho de um especialista. Este processo é caro e lento, portanto vale a pena procurar maneiras mais automatizadas de construir bases de conhecimento especialistas. Embora ainda não exista nenhum sistema de aquisição de conhecimento totalmente automatizado, há muitos programas que interagem com os especialistas dos domínios para extrair conhecimento especializado com eficiência. Estes programas oferecem suporte às seguintes atividades:

- Inserção de conhecimento;
- Manutenção da consistência da base de conhecimento;
- Garantia de completeza da base de conhecimento.

5.6. Problemas

Alguns dos principais problemas enfrentados pelos sistemas especialistas atuais são:

- *Fragilidade:* Como os sistemas especialistas só têm acesso a conhecimentos altamente específicos do domínio, eles não podem contar com conhecimentos mais genéricos quando a necessidade surge. Por exemplo, suponha que cometamos um erro na entrada de dados para um sistema especialista médico e que descrevamos um paciente com 130 anos e 20 quilos. A maioria dos sistemas não seria capaz de adivinhar que podemos ter invertido os dois campos. já que os valores não são muito plausíveis. Alguns sistemas representam uma tentativa de remediar esse problema fornecendo um substrato de conhecimentos de senso comum, sobre o qual sistemas especialistas específicos podem ser criados.
- *Falta de Meta-conhecimento* Os sistemas especialistas não têm conhecimentos muito sofisticados sobre sua própria operação. Eles normalmente não conseguem raciocinar sobre seu próprio escopo e restrições, dificultando ainda mais a tarefa de lidar com a sua fragilidade.
- *Aquisição de Conhecimento* Apesar do desenvolvimento de ferramentas, a aquisição ainda continua sendo um dos maiores obstáculos à aplicação da tecnologia dos sistemas especialistas a novos domínios.
- *Validação:* Medir o desempenho de um sistema especialista é difícil, porque não sabemos como quantificar o uso do conhecimento. Certamente é impossível apresentar provas formais de correção para sistemas especialistas. Uma coisa que podemos fazer é comparar esses sistemas com especialistas humanos em problemas do mundo real Por exemplo. o *MYCIN* participou de um painel de especialistas para avaliar dez casos selecionados de meningite, obtendo resultados melhores do que qualquer um de seus concorrentes humanos.

5.7. Conclusões

Desde a metade da década de 60, quando começaram os trabalhos sobre os sistemas que hoje conhecemos como especialistas, muito progresso foi feito na construção de tais programas. As experiências obtidas com esses esforços sugerem as seguintes conclusões.

- Esses sistemas derivam sua potencialidade de uma grande quantidade de conhecimento especifico do domínio, e não de uma única técnica poderosa.
- Os sistemas bem-sucedidos têm muito conhecimento bem-definido sobre uma determinada área. Isto contrasta com o conhecimento amplo e de difícil definição que chamamos de *senso comum.* E mais fácil criar sistemas especialistas do que sistemas com senso comum.
- Um sistema especialista em geral é construído com a ajuda de um ou mais especialistas, que precisam estar dispostos a transferirem o seu conhecimento para o sistema.
- A transferência de conhecimento ocorre gradualmente, através de muitas interações entre o especialista e o sistema. O especialista nunca fornece de pronto o conhecimento correto ou completo.
- A quantidade de conhecimento exigida depende da tarefa. Ela pode variar de 40 regras a milhares delas.
- A escolha da estrutura de controle para um determinado sistema depende das características especificas do sistema.

- É possível extrair partes não-especificas do domínio, de sistemas especialistas existentes, e usá-las como ferramentas para a criação de novos sistemas e novos domínios.

6. Engenharia de Conhecimento

O objetivo da Engenharia de Conhecimento é desenvolver sistemas de IA baseados em conhecimento através da modelagem dos processos cognitivos para resolução de problemas utilizados pelos especialistas em algum domínio.

A Engenharia de Conhecimento é uma tarefa artesanal, pois os métodos utilizados pelos especialistas são particulares e, as atividades dependem do tipo e do domínio do problema.

Os profissionais envolvidos são

- Especialista;
- Engenheiro de Conhecimento;
- Programador.

6.1. Metodologias de Desenvolvimento

- Prototipação
  Construção incremental de protótipos do sistema, até que se torne funcional.
- Estruturado
  Projeto planejado e realizado em etapas que partindo das especificações chega diretamente à versão operacional do sistema.
- Misto
  O sistema é desenvolvido de forma estruturada, porém alguns módulos ou funções são construídos utilizando prototipação.

6.2. Atividades em um Sistema com IA

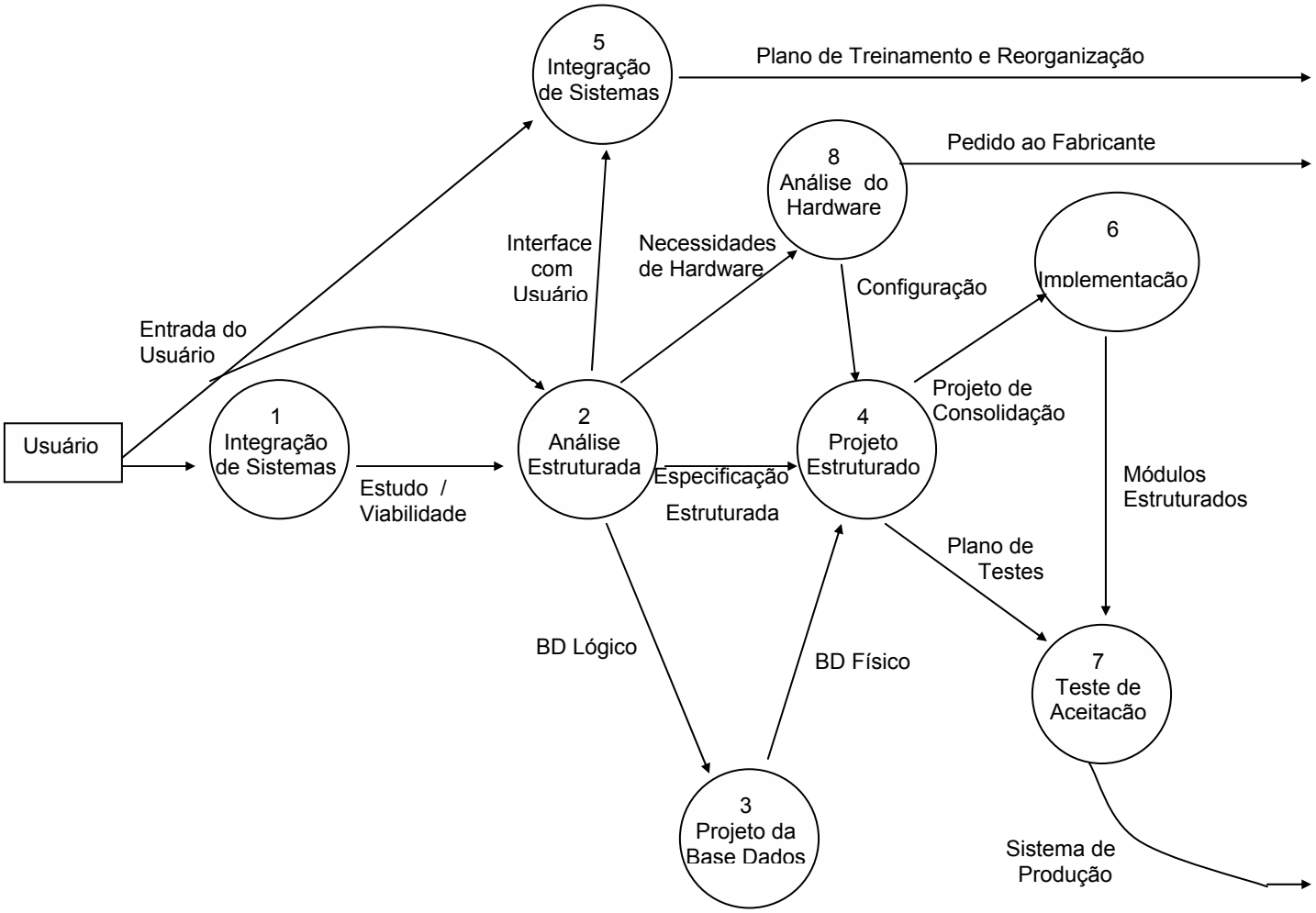


Figura 32 - DFD nível o do ciclo de vida de um projeto estruturado

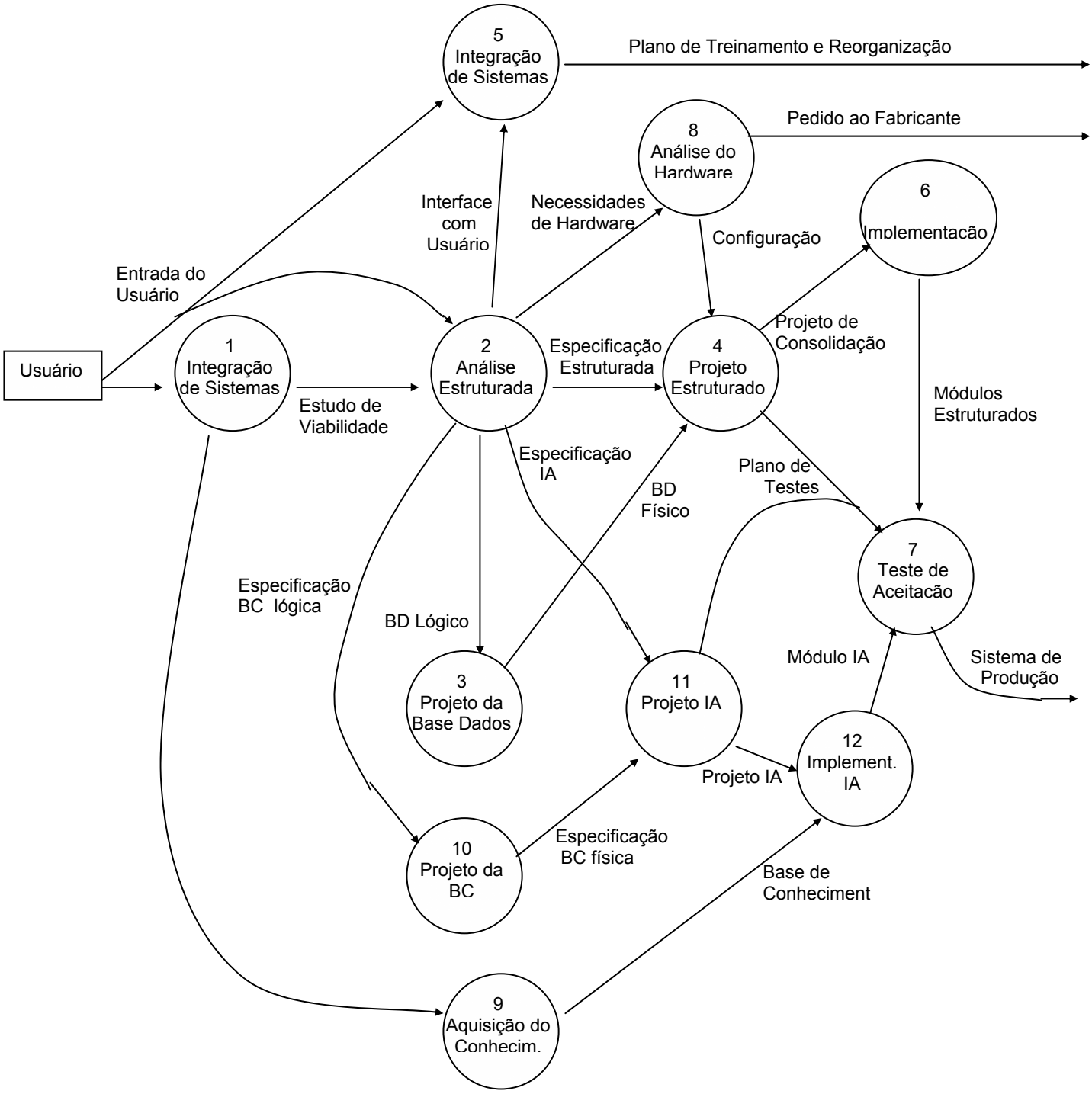


Figura 33 - DFD nível o do ciclo de vida de um projeto com IA

Objetivos adicionais por fase

1. Levantamento
   - Selecionar domínios apropriados para aplicações da IA.
   - Propor estratégias de curto e longo prazo para a integração da tecnologia de IA.
   - Analisar a relação custo / benefício da aplicação ao domínio.

- Definir a abrangência e resultados.
- Iniciar a definição da Base de Conhecimento (viabilidade).

2. Análise Estruturada
   - Elaborar um conjunto de DFDs com descrições narrativas dos processos (tarefas do especialista) e um dicionário de dados.
   - Especificar os requerimentos de interface com o usuário.
   - Especificar os requerimentos de desempenho.

3. Projeto do Banco de Conhecimento
   - Definir a forma de representar o conhecimento.

4. Projeto Estruturado
   - Definir a forma de raciocínio.
   - Definir a interface com o usuário.
   - Escolher o instrumento (linguagem, ambiente ou ferramenta).

5. Integração de Sistemas

6. Implementação
   - Codificação do conhecimento de acordo com o método da ferramenta.

7. Teste de Aceitação

8. Análise do Hardware

9. Aquisição do Conhecimento

6.3. Etapas na Construção de um pequeno SE

Ainda não existe uma metodologia consolidada para a construção de SE, entretanto abaixo são enumerados as etapas com alguns detalhes do processo de construção de um SE pequeno. É importante observar que na construção de um SE pequeno freqüentemente os passos 1 e 2 são invertidos, pois é mais fácil identificar um problema que seja adequado a uma ferramenta previamente selecionada do que o contrário.

1. Selecionar um instrumento.
   Implica em comprometer-se implicitamente com vários aspectos relacionados ao desenvolvimento de um SE. No caso de uma ferramenta, os aspectos são:
   - Paradigma de consulta - corresponde ao tipo de problema que será solucionado, sendo os principais: diagnóstico / prescrição, planejamento e projeto.
   - Forma de Representação do Conhecimento: regras, objetos, etc.
   - Formas de Inferência.
   - Interface com o usuário: tipo, qualidade, tratamento de dúvidas, etc.
2. Identificar um problema e analisar as suas características.
   As principais características a serem avaliadas são:
   - Tempo de solução.
   - Tipo de comunicação.
   - Tipo de conhecimento que o sistema deverá incluir.
   - Volume de alternativas de solução.
3. Projetar o sistema.
   Esta etapa é a mais trabalhosa e demorada. Exige estudar e levantar o conhecimento que será utilizado pelo SE, o que implica fazer diversas entrevistas com os especialistas para identificar como eles trabalham ou raciocinam para resolver os problemas.
   - Descrição do sistema.

- Diagramas de fluxo de consulta.
- Tabelas.

4. Desenvolver um protótipo.
   - Criar o banco de conhecimento, o que exige conhecimento da sintaxe da linguagem de representação de conhecimento.
   - Verificar fazendo algumas consultas. Esta atividade é importante porque serve para validar empiricamente o SE. Além de entrar com situações reais para verificar se as soluções apresentadas pelo sistema são válidas, é necessário entrar com valores inválidos e utilizar várias alternativas de resposta, como "Por que?" ou indicar dúvida (incerteza).
5. Expandir, verificar e revisar o sistema até cumprir o seu papel.
   - Incluir novas "peças" de conhecimento (regras, ...).
   - Incluir o tratamento de conhecimento incompleto.
   - Incluir ou alterar o tratamento de incerteza.
   - Incluir ou refinar os mecanismos de ajuda ou explicação.
6. Manter e atualizar o sistema.

Exemplo de desenvolvimento de um protótipo de SE:

Este exemplo foi desenvolvido até a quanta etapa, portanto o protótipo apresenta várias deficiências que seriam corrigidas no decorrer da quinta etapa.

1. Selecionar um instrumento.

O instrumento selecionado é o SINTA, uma ferramenta de desenvolvimento de SE bastante fácil de ser utilizada, permitindo que o engenheiro de conhecimento se concentre na aplicação.

O paradigma de consulta do Sinta é diagnóstico / prescrição, sendo que este paradigma pode ser utilizado na solução de uma gama bastante grande de problemas.

A forma de representação do conhecimento é baseada em regras, permitindo a definição de fatores de certeza. A inferência é feita através de encadeamento regressivo das regras.

A interface com o usuário tem uma forma padrão de pergunta, quando é necessário consultar o usuário sobre o valor de uma variável, entretanto permite que cada pergunta seja alterada. A interface possui também tratamento de dúvidas através de um mecanismo de explicação. A explicação padrão é a apresentação da linha de raciocínio utilizada que exigiu a consulta ao usuário. Entretanto, o engenheiro de conhecimento pode incluir uma explicação não padrão.

2. Identificar um problema e analisar as suas características.

O problema que será utilizado no exemplo é a seleção de meios instrucionais adequados para diferentes tipos de treinamento. As características deste problema são:

- Tempo de solução: até 30 minutos;
- Tipo de comunicação: o especialista apresenta uma proposta verbalmente ou por escrito;
- Tipo de conhecimento: o especialista utiliza regras e cálculos simples;
- Volume de alternativas de solução: poucas alternativas, da ordem de dezenas de possibilidades.

O problema é adequado à ferramenta que foi selecionada. Para problemas com um grau de complexidade grande, o Sinta não poderia ser utilizado.

3. Projetar o sistema.

O SE terá como objetivo aconselhar um cliente a respeito da seleção de meios instrucionais para um determinado treinamento. Assim, é necessário estudar o assunto através de livros especializados e reunir-se com especialistas na área de treinamento para determinar os principais atributos da seleção de meios instrucionais. Segundo Tosti e Ball (em "A behavioral approach to instructional design and media selection"), é necessário discutir os aspectos do problema de treinamento, considerar como estes aspectos se relacionam com as características dos meios e indicar um ou mais meios instrucionais que resolvam o problema.

A 0 apresenta um diagrama do fluxo de consulta sobre meios instrucionais. Para elaborar este diagrama é necessário entender como o especialista trabalha e identificar quais são e a ordem das perguntas que ele faz ao cliente para resolver o problema. O diagrama facilita a elaboração das tabelas e a construção da base de conhecimento. No nosso caso, o especialista começa perguntando sobre a tarefa que os treinandos cumprirão após o treinamento. Depois, sobre como simular a execução da tarefa, pois um programa de treinamento ideal procura fazer com que se pratique as atividades previstas na tarefa. A seguir são

considerados os aspectos instrucionais e as restrições orçamentárias. Ao final são recomendados de um a três meios eficazes respeitando o custo.

As tabelas servem para organizar o conhecimento que vai sendo levantado junto às fontes utilizadas. A tabela da Figura 35 apresenta os principais fatores que o especialista leva em conta para decidir quais meios são recomendados.

A Figura 36 faz uma análise levantando algumas das possíveis relações entre as ponderações apresentadas na Figura 35 e as recomendações que o especialista faria.

Se o especialista utilizar apenas estes fatores, é possível identificar quantas situações diferentes existem, através da multiplicação dos números de alternativas (respostas) de cada fator: 4 x 2 x 6 x 2 x 4 x 3 = 1152.

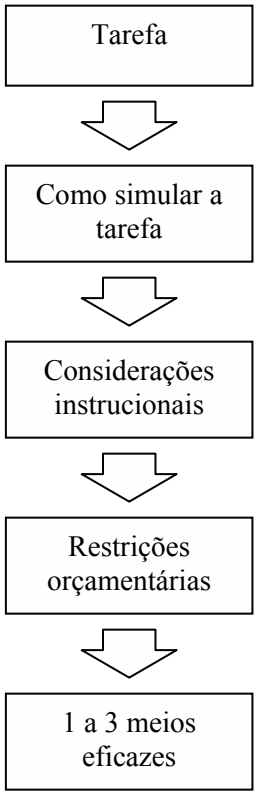


Figura 34 Fluxo de Consulta sobre meios instrucionais

Ponderações

de trabalho ou tarefa	de ensino
❑ **Situação do Estímulo** o Ambiental o Pictória o Simbólica o Verbal	❑ **Realimentação Instrucional** o Sim o Não
❑ **Duração do Estimulo** o Breve o Persistente	❑ **Modificação da Apresentação** o por Resposta o por Módulo o por Curso o Nenhuma
❑ **Resposta Apropriada** o Oculta o Seletiva o Construída o Oral o Motora o Afetiva	❑ **Orçamento de Treinamento** o Pequeno (até R$ 10.000,00) o Médio (até R$ 50.000,00) o Grande (mais de R$ 10.000,00)

Figura 35 - Ponderações do modelo Tosti-Ball (simplificado)

N°	Situação do Estímulo	Duração do Estímulo	Resposta Apropriada	Realimentação Instrucional	Modificação da Apresentação	Orçamento de Treinamento	Recomendações
1	verbal	breve	oculta	não	nenhuma	pequeno	livro
2	verbal	persistente	seletiva construída	sim	resposta módulo	pequeno	estudo pessoal
3	verbal	breve	oculta	não	curso	médio	aula
4	verbal simbólica pictória	breve	oculta	não	curso	médio	aula com dispositivos
5	verbal pictória	breve	oculta	não	módulo	médio	videocassete
6	verbal	breve	oral afetiva	sim	módulo	pequeno médio	prática com realimentação verbal
7	verbal	breve	oral afetiva	sim	módulo	médio grande	prática com reali-mentação por vídeo
8	ambiental	persistente	motora	não	módulo curso	grande	Laboratório com experiências
9	verbal	breve	oculta	não	nenhuma	pequeno	áudio cassete
10	simbólica verbal	breve	todas	sim	resposta	médio grande	tutor humano

Figura 36 - Análise matricial de escolha do meio instrucional

A situação de estímulo é ...	quando envolve ...
ambiental	modelo físico estrutura objeto máquina instrumento
pictória	figura fotografia diagrama ilustração vista
simbólica	gráfico esquema números fórmulas
verbal	ouvir conversa diálogo leitura texto

Figura 37 - Tabela de detalhamento da situação do estímulo.

A resposta apropriada é ...	quando envolve ...
oculta	ouvir ler observar meditar imaginar pensar
seletiva	múltipla escolha associação
construída	escrever desenhar datilografar
verbal	dizer algo conversar falar cantar
atividade motora	edificar montar mover construir organizar dançar
afetiva	emocionar-se simpatizar sentir mostrar afeição compreensivo manter a calma

Figura 38 - Tabela de detalhamento da resposta apropriada.

4. Desenvolver um protótipo.

Esta etapa envolve criar o banco de conhecimento e verificar fazendo algumas consultas.

O Sinta exige que se defina as variáveis e os respectivos domínios antes de entrar com as regras.

O objetivo do SE é estabelecer uma ou mais recomendações, então criou-se uma variável chamada "Recomendação" cujo domínio são as possíveis recomendações dos especialistas, conforme a tabela da Figura 36.

Para se criar as regras, inicialmente foi utiliza a tabela da Figura 36, resultando nas regras de 1 a 10, como listado abaixo. Na lógica o operador de conjunção ("E") possui maior precedência que o operador de disjunção ("OU"), entretanto, para facilitar a formação das regras, o Sinta utiliza como padrão o inverso: o operador de disjunção possui maior precedência que o operador de conjunção.

As tabelas da Figura 37 e da Figura 38 foram utilizadas para criar as demais regras: de 11 a 20.

REGRAS

Regra 1 - Recomendação é livro
SE situação do estímulo = verbal
E duração do estímulo = persistente
E resposta apropriada = oculta
E realimentação instrucional = não
E modificação da apresentação = nenhuma
E orçamento de treinamento = pequeno
OU orçamento de treinamento = médio
ENTÃO recomendação = livro

Regra 2 - Recomendação é estudo pessoal
SE situação do estímulo = verbal
E duração do estímulo = persistente
E resposta apropriada = seletiva
E resposta apropriada = construída
E realimentação instrucional = sim
E modificação da apresentação = por resposta
OU modificação da apresentação = por módulo
E orçamento de treinamento = pequeno
ENTÃO recomendação = estudo pessoal

Regra 3 - Recomendação é aula
SE situação do estímulo = verbal
E duração do estímulo = breve
E resposta apropriada = oculta
E realimentação instrucional = não
E modificação da apresentação = por curso
E orçamento de treinamento = pequeno
OU orçamento de treinamento = médio
ENTÃO recomendação = aula

Regra 4 - Recomendação é aula com dispositivos
SE situação do estímulo = verbal
OU situação do estímulo = simbólica
OU situação do estímulo = pictória
E duração do estímulo = breve
E resposta apropriada = oculta
E realimentação instrucional = não
E modificação da apresentação = por curso

E orçamento de treinamento = médio
ENTÃO recomendação = aula com dispositivos

Regra 5 - Recomendação é videocassete
SE situação do estímulo = verbal
OU situação do estímulo = pictória
E duração do estímulo = breve
E resposta apropriada = oculta
E realimentação instrucional = não
E modificação da apresentação = por módulo
E orçamento de treinamento = médio
ENTÃO recomendação = videocassete

Regra 6 - Recomendação é prática com realimentação verbal
SE situação do estímulo = verbal
E duração do estímulo = breve
E resposta apropriada = verbal
OU resposta apropriada = afetiva
E realimentação instrucional = sim
E modificação da apresentação = por módulo
E orçamento de treinamento = pequeno
OU orçamento de treinamento = médio
ENTÃO recomendação = prática com realimentação verbal

Regra 7 - Recomendação é prática com realimentação por vídeo
SE situação do estímulo = verbal
E duração do estímulo = breve
E resposta apropriada = verbal
OU resposta apropriada = afetiva
E realimentação instrucional = sim
E modificação da apresentação = por módulo
E orçamento de treinamento = médio
OU orçamento de treinamento = grande
ENTÃO recomendação = prática com realimentação por vídeo

Regra 8 - Recomendação é laboratório com experiências
SE situação do estímulo = ambiental
E duração do estímulo = persistente
E resposta apropriada = motora
E realimentação instrucional = não
E modificação da apresentação = por módulo
OU modificação da apresentação = por curso
E orçamento de treinamento = grande
ENTÃO recomendação = laboratório com experiências

Regra 9 - Recomendação é áudio cassete
SE situação do estímulo = verbal
E duração do estímulo = breve
E resposta apropriada = oculta
E realimentação instrucional = não
E modificação da apresentação = nenhuma
E orçamento de treinamento = pequeno
ENTÃO recomendação = audio cassete

Regra 10 - Recomendação é tutor humano

SE situação do estímulo = simbólica
OU situação do estímulo = verbal
E duração do estímulo = breve
E resposta apropriada = afetiva
OU resposta apropriada = construída
OU resposta apropriada = motora
OU resposta apropriada = oculta
OU resposta apropriada = seletiva
OU resposta apropriada = verbal
E realimentação instrucional = sim
OU modificação da apresentação = por resposta
E orçamento de treinamento = médio
OU orçamento de treinamento = grande
ENTÃO recomendação = tutor humano

Regra 11 - Situação de estímulo é ambiental
SE estímulo = modelo físico
OU estímulo = estrutura
OU estímulo = objeto
OU estímulo = máquina
OU estímulo = instrumento
OU estímulo = ambiental
ENTÃO situação do estímulo = ambiental

Regra 12 - Situação de estímulo é pictória
SE estímulo = figura
OU estímulo = fotografia
OU estímulo = diagrama
OU estímulo = ilustração
OU estímulo = vista
OU estímulo = pictória
ENTÃO situação do estímulo = pictória

Regra 13 - Situação de estímulo é simbólica
SE estímulo = gráfico
OU estímulo = esquema
OU estímulo = números
OU estímulo = fórmulas
OU estímulo = simbólica
ENTÃO situação do estímulo = simbólica

Regra 14 - Situação de estímulo é verbal
SE estímulo = ouvir
OU estímulo = conversa
OU estímulo = diálogo
OU estímulo = leitura
OU estímulo = texto
OU estímulo = verbal
ENTÃO situação do estímulo = verbal

Regra 15 - Resposta apropriada é oculta
SE resposta = ouvir
OU resposta = ler
OU resposta = observar
OU resposta = meditar

OU resposta = imaginar
OU resposta = pensar
OU resposta = oculta
ENTÃO resposta apropriada = oculta

Regra 16 - Resposta apropriada é seletiva
SE resposta = múltipla escolha
OU resposta = associação
OU resposta = seletiva
ENTÃO resposta apropriada = seletiva

Regra 17 - Resposta apropriada é construída
SE resposta = escrever
OU resposta = desenhar
OU resposta = datilografar
ENTÃO resposta apropriada = construída

Regra 18 - Resposta apropriada é verbal
SE resposta = dizer algo
OU resposta = conversar
OU resposta = falar
OU resposta = cantar
OU resposta = verbal
ENTÃO resposta apropriada = verbal

Regra 19 - Resposta apropriada é motora
SE resposta = edificar
OU resposta = montar
OU resposta = mover
OU resposta = construir
OU resposta = organizar
OU resposta = dançar
OU resposta = atividade motora
ENTÃO resposta apropriada = motora

Regra 20 - Resposta apropriada é afetiva
SE resposta = emocionar-se
OU resposta = simpatizar
OU resposta = sentir
OU resposta = mostrar aveição
OU resposta = comprensivo
OU resposta = calma
ENTÃO resposta apropriada = afetiva

PERGUNTAS
Variável:duração do estímulo
Pergunta:"Qual é a duração do estímulo?"
Variável:estímulo
Pergunta:"Qual é a situação do estímulo que a pessoa receberá?"
Variável:modificação da apresentação
Pergunta:"Quando é feita a modificação da apresentação (se houver) ?"
Motivo:"Informar se o modo de apresentação do conteúdo do
"curso é alterado e em que momento isto é feito.
"
"Exemplo: se a apresentação do conteúdo deve ser

```
        "alterada conforme a resposta do aluno, então escolha
        "a opção "por resposta".
Variável:orçamento de treinamento
  Pergunta:"Qual é o orçamento do treinamento?"
  Motivo:"Informe
        "
        ""pequeno"  se será gasto menos de R$ 100,00
        ""médio"  se será gasto menos de R$ 1.000,00
        ""grande"   se será gasto mais de R$ 1.000,00
Variável:realimentação instrucional
  Pergunta:"Existe realimentação instrucional?"
Variável:resposta
  Pergunta:"Qual é o tipo de resposta que a pessoa deve fornecer?"
```