5 Conhecimento

"Porque, às vezes, acredito em até seis coisas impossíveis antes do café da manhã."
(Lewis Carroll, *Através do Espelho*)

5.1 INTRODUÇÃO

O conhecimento pode ser definido como a **informação armazenada** ou os **modelos usados** por uma pessoa ou máquina para **interpretar, identificar, predizer e responder** apropriadamente ao mundo externo. De fato, o conhecimento e a inteligência sempre aparecem com uma grande dependência um do outro. A importância de se representar o conhecimento não é apenas poder recuperá-lo no futuro, mas também raciocinar com ele e, principalmente, a facilidade para agregar novos conhecimentos.

Para que o conhecimento possa ser armazenado e manipulado por um computador é preciso que sejam adotados **modelos adequados ao seu processamento**. Assim, a escolha do modelo mais adequado para a representação do conhecimento em um computador depende, basicamente, do tipo de conhecimento que será armazenado e do modo como este conhecimento será processado pelo computador, além disso, deve se considerar também a **facilidade para se incluir novas informações**.

Como sempre, os seres humanos constituem o primeiro modelo a ser observado, pois acumulam conhecimentos que lhes permitem agir de modo inteligente. Este processo pode ser descrito através dos seguintes passos:

1 – ocorre o estímulo sensorial (visão, audição, tato, etc.);

2 – ocorre a retenção do estímulo por algum tempo;

3 – a informação é trabalhada pela memória de trabalho;

4 – ocorre a resposta ao estímulo;

5 – a informação é registrada ou não na memória de longa duração.

Embora estes passos sejam comuns a todas as pessoas, elas não captam todo o conhecimento que a realidade do mundo lhe proporciona, além disso, pessoas diferentes, diante da mesma realidade, captam e entendem eventos diferentes. De qualquer modo, seja por uma pessoa ou por uma máquina, a manipulação do conhecimento sempre exige algum tipo de representação, observando que diferentes formas de representação facilitam a manipulação por diferentes agentes (computador ou humanos).

5.2 UMA BOA REPRESENTAÇÃO

A representação usada por um problema é muito importante: o modo pelo qual o computador **representa** um problema, as **variáveis** que ele usa e os **operadores** que aplica pode fazer diferença entre um algoritmo **eficiente** e um algoritmo **funcional**.

EXEMPLO: Procurando uma lente de contato que caiu em um campo de futebol - utilizaria algum conhecimento prévio de onde estava quando caiu, procurando a partir desse ponto. E o computador? Supostamente ele teria um "oráculo" que responderia qualquer pergunta acerca do campo.

- Dividir o campo em quatro quadrados iguais e perguntar ao "oráculo", para cada quadrado: "A lente está nesse quadrado?". Mesmo assim ainda teria muito espaço para procurar.
- Ter uma grade contendo uma representação de cada átomo do campo: "A lente está em contato com esse átomo?". Seria uma resposta altamente precisa, no entanto seria um modo ineficiente para encontrar a lente. Levaria muito tempo.
- Uma melhor representação seria dividir o campo em quadrados de um centímetro e eliminar todos os quadrados que você sabe que não estão por perto de onde você estava quando perdeu a lente.

Todas as representações são idênticas, exceto pelos diferentes níveis de granularidade. O mais difícil é determinar a estrutura de dados que será utilizada para representar o problema.

Ao aplicar IA em problemas de busca, é essencial uma representação **útil**, **eficiente** e **significativa**. A representação deve ser tal que um computador não gaste muito tempo em **computações desnecessárias**, que seja realmente **relacionada ao problema** e que forneça um meio pelo qual o computador possa realmente **resolver o problema**.

5.3 MODELOS DE REPRESENTAÇÃO DO CONHECIMENTO

É possível encontrar na literatura algo em torno de dez modelos conhecidos para a representação do conhecimento, porém, os mais conhecidos são: a representação por **Lógica**, a representação por **Regras de Produção**, a representação por **Redes Semânticas**, a representação por **Quadros e Roteiros** e a representação usando **Árvores**, que também podem ser consideradas um caso particular das redes.

Cada um destes modelos apresenta caraterísticas peculiares que sugerem o seu uso em diferentes situações, embora, uma combinação deles possa constituir também uma alternativa muito interessante para se obter sistemas mais robustos.

5.4 REPRESENTAÇÃO POR LÓGICA

A lógica foi criada por Aristóteles há mais de 2300 anos, objetivando ser uma linguagem que representasse os processos envolvidos no pensamento. A **Lógica Proposicional** é a forma mais simples e conhecida. Porém, por causa de suas limitações, várias adaptações e acréscimos foram propostos, sempre objetivando tratar situações mais complexas. Algumas destas extensões são a Lógica **Multivalorada**, a Lógica dos **Predicados** e a Lógica **Nebulosa** que falaremos mais tarde.

5.4.1. LÓGICA PROPOSICIONAL

Nesta teoria, um símbolo representa uma proposição inteira ou um fato, como, por exemplo, P = "A professora está falando" e esta sentença pode assumir apenas um valor verdadeiro ou falso (binário). Várias proposições podem ser combinadas, usando-se os operadores lógicos: E, OU, ¬ (negação), → (condicional ou implicação) e ←→ (bi condicional).

A grande vantagem do uso deste modelo é que existe uma forte teoria apoiando a sua manipulação. Assim, é possível usá-lo para implementar a análise automática do conhecimento através das **propriedades de equivalência** e de **regras de inferência** que, a partir de proposições conhecidas, conseguem gerar novas informações.

PROPOSIÇÃO: é uma sentença declarativa, afirmativa, que deve exprimir um pensamento de sentido completo, podendo ser escrita na forma simbólica ou na linguagem natural.
Ex: 23 < 4, O Brasil fica na América do Sul, Paris é a capital da Turquia

O valor lógico de cada proposição pode ser a verdade (1) ou a falsidade (0).

Proposição Simples: É aquela que não possui uma outra proposição como parte integrante de si mesma, sendo representada por uma letra minúscula.

p : está chovendo
q : a raiz quadrada de 4 é 2

Proposição Composta: É formada por duas ou mais proposições relacionadas pelos conectivos **E**, **OU**, e **SE..ENTÃO**, sendo representada por uma letras maiúsculas.

p : está chovendo e ventando
q : se está chovendo então a pista está molhada
r : casa ou compra uma bicicleta

Os princípios fundamentais da lógica matemática (proposicional) são:

1) Princípio da **não contradição**, que afirma que uma proposição não pode ser simultaneamente verdadeira e falsa

2) Princípio do **terceiro excluído**, que afirma que toda proposição ou é somente verdadeira ou somente falsa, nunca ocorrendo um terceiro estado.

5.4.2. LÓGICA DE PREDICADOS ou LÓGICA DE PRIMEIRA ORDEM

Predicados são declarações à respeito de objetos ou relações entre objetos, que tornam capaz a representação de **fatos**, **objetos** e **relações**, incluindo funções, variáveis e quantificadores.

Predicados são também capazes de representar **regras**, e estendem a lógica proposicional para se ter mais capacidade de expressão. Um mecanismo externo à base deverá manipula-la com as regras de inferência para, então, resolver os problemas. Alguns conceitos importantes da Lógica de Predicados são:

Objetos: substantivos (casa, lápis, maria, ...)

Relações: verbos usados para descrever relações entre objetos

Funções: relações em que existe somente um valor para uma dada entrada

Domínio: um conjunto de objetos

Elementos do domínio: são os objetos pertencentes ao domínio

Sentenças atômicas: usam-se termos para referenciar objetos e predicados para referenciar relações e fatos. Elas são formadas por um símbolo de predicado seguido por uma lista de termos entre parênteses.

pai(Luis, Pedro) → Luis é pai de Pedro
casado(pai(Ana),mãe(Carla)) → o pai de Ana é casado com a mãe de Carla

Sentenças complexas: São formadas com conectivos lógicos E, OU, e SE..ENTÃO, como abaixo:
irmão(Luis, Pedro) ^ casado(Lucas, Maria) → Luis é irmão de Pedro e Lucas é casado com Maria

Quantificadores: A lógica de primeira ordem possui dois quantificadores padrão, que são o quantificador universal ∀, que significa "para todo" e o quantificador existencial, ∃, que significa "existe"

∀x traidor(x) → enforcado(x) → Todo traidor é enforcado
∃x Árvore(x) ^ NoJardim(x, escola) → Existe uma árvore no jardim da escola

Os quantificadores podem ser aninhados, tal como:

∀x ∀y irmão(x,y) → parente (x,y) → Irmãos sempre são parentes =-)
∀x ∃y ama(x,y) → Todo mundo ama alguém <3

De fato, existe uma forte conexão entre os quantificadores ∀ e ∃, pois, afirmar "∀x é verdade" tem o mesmo significado que "negar ∃x falso"

∀x ¬ gosta(x,laranja) ⇔ ¬ ∃x gosta(x,laranja)
Todos não gostam de laranja ⇔ Ninguém gosta de laranja

5.4.3. LÓGICA TEMPORAL

Trata-se de uma extensão da lógica clássica em que conectivos descrevendo relações temporais são inclusos (**F** futuro, **P** presente, **G** todo o futuro, **H** todo o passado). Define uma variável de tempo nova e usa os mesmos conectivos E, OU e NÃO usuais.

Assim, **FA** significa que **A** será verdade em algum momento do futuro, **PA** indica que **A** foi verdade em algum momento do passado, **GA** será verdade em todos os momentos do futuro e, finalmente, **HA** afirma que **A** foi verdade em todos os momentos do passado.

5.4.4. LÓGICA MODAL

A lógica dos predicados não diferencia os conceitos de **possível** e **necessário**, pois, para ela, fórmulas são apenas verdadeiras ou falsas. A Lógica Modal considera os diferentes modos nos quais uma declaração pode ser verdadeira, suportando as expressões "é possível que" e "é necessário que".

5.5 REPRESENTAÇÃO POR SISTEMAS DE PRODUÇÃO

Este modelo foi proposto pelo matemático Emil Post, em 1943, para tratar procedimentos computáveis. Por causa da sua simplicidade, é o modelo de representação de conhecimento mais usado na prática. A ideia central consiste em transformar o problema em um **grafo de estados**, com a especificação de estados inicial e final, além das regras de transição entre os estados que sempre estão no formato **SE <condição> ENTÃO <ação>**.

Assim, na **condição**, um teste verifica a presença, ou não, de certas informações na base de conhecimento, enquanto que na **ação**, altera-se o estado atual da base de conhecimento, adicionando, modificando ou removendo unidades de conhecimento presentes na base.

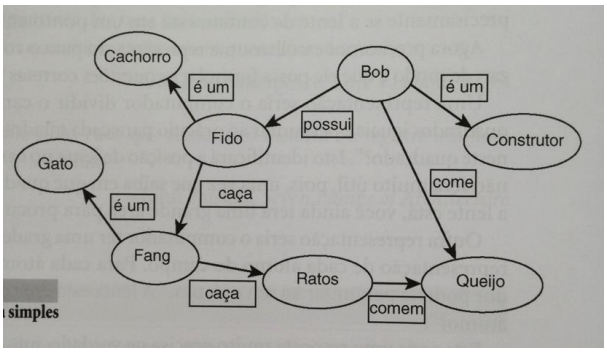
SE febre alta **E**
erupções cutâneas **E**
dores musculares
ENTÃO dengue

Este modelo é bastante recomendado em domínios nos quais o conhecimento é muito difuso, consistido de muitos fatos ou ações independentes, como é o caso da Medicina.

5.6 REDES SEMÂNTICAS

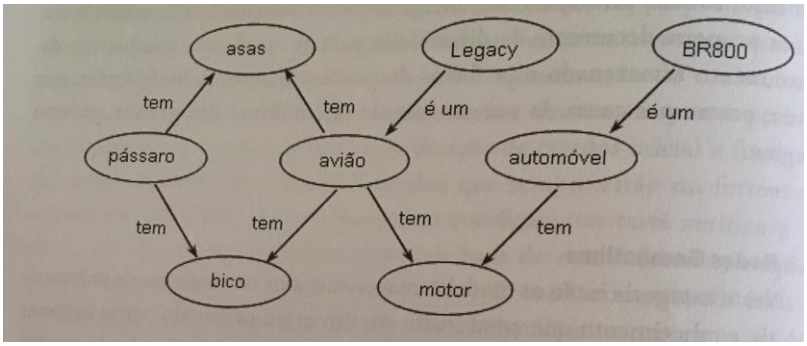
Nesta categoria estão os modelos mais avançados e complexos de representação do conhecimento que combinam, em um grafo orientado, tanto associações declarativas como procedurais. Uma rede semântica consiste, basicamente, em um conjunto heterogêneo de objetos representados por um grafo, em que os **nós**, em geral, representam **objetos** e as **arestas** representam relações binárias entre os objetos.

Originalmente as redes semânticas foram usadas como suporte ao estudo do processamento de linguagem natural, constituindo estruturas muito poderosas que conseguem representar vários tipos de conhecimento. Acredita-se que este seja o modelo mais parecido com o que se imagina acontecer no mecanismo cerebral. Em sua versão mais simples, usa nós para objetos, situações e conceitos, e arcos para representar as relações entre eles, conforme ilustra a figura:

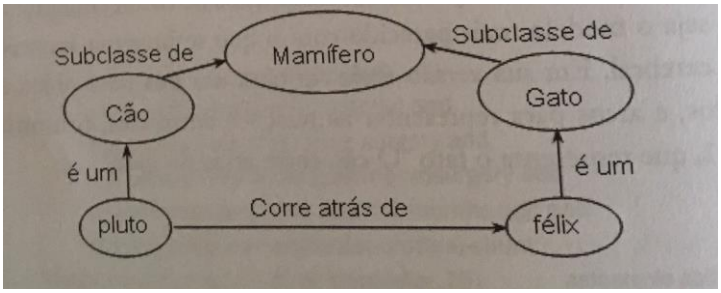


Representamos objetos específicos (Fang, Bob e Fido) que são **instâncias** de uma **classe** particular (cachorro, gato). Pelo gráfico, não fica claro se queijo é uma classe ou uma instância de classe. Por isso, Redes Semânticas devem possuir regras, por exemplo: "qualquer objeto que não tenha um relacionamento 'é-um' com uma classe representará uma classe de objetos".

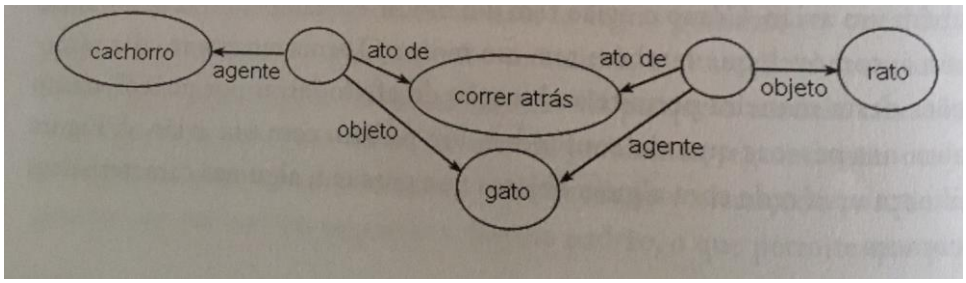
Redes semânticas constituem uma importante estrutura capaz de simular o modelo psicológico da memória associativa, permitindo associações entre objetos com características em comum. Assim, um pássaro que tem asas, lembra também um avião. Como o avião tem motor, ele também traz lembranças de um automóvel, que também tem motor. O armazenamento das informações dessa maneira permite a obtenção do efeito "errar por pouco", muito comum nas pessoas.



A figura a seguir apresenta algumas melhorias neste modelo com a inclusão do conceito de **subclasse**.



Uma rede mais avançada é a **Rede Semântica Generalizada** que procura resolver limitações das redes semânticas elementares ao representar verbos através de nós e não mais de arcos. A figura representa o conhecimento "O cachorro corre atrás do gato e o gato corre atrás do rato.



Algumas vantagens importantes do uso de redes para representar o conhecimento são: simplificam a forma de representar o problema, pois o ser humano consegue visualizar o conhecimento codificado; podem estabelecer relações de causa e efeito simples e intuitivas. Como desvantagens tem-se a dificuldade em se definir hierarquia; encontrar uma semântica exata do nó e das suas ligações e representar o fator tempo.

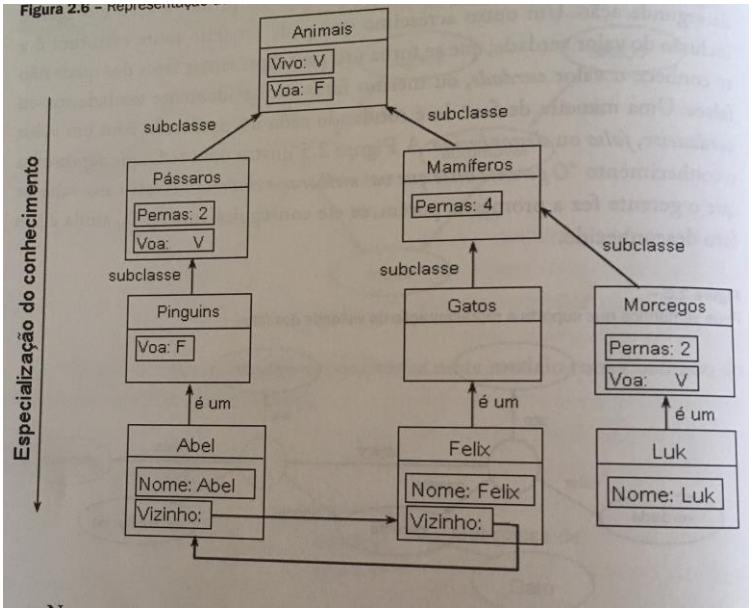
5.7 QUADROS

Esta representação se baseia na hipótese de que as pessoas usam conhecimentos adquiridos em experiências anteriores para resolver situações novas. A nova experiência também serve para incrementar o conhecimento atual, gerando maior especialização.

Assim, um quadro consiste em um conjunto de atributos valorados (compartimentos) que descrevem as propriedades do objeto representado, conforme mostra a figura a seguir, que define, inicialmente, uma classe de animais, que tem dois parâmetros vivo = verdadeiro e voa = falso.

Neste exemplo, são criadas duas subclasses (pássaros e mamíferos), sendo que para a primeira, o atributo *voa* precisa ser redefinido para verdadeiro. Nas duas subclasses também é inserido o parâmetro *número de pernas*, que no caso dos pássaros são duas e no caso dos mamíferos são quatro.

Os valores dos atributos podem ser, além dos valores do objeto em particular, valores *default*, ponteiros para outros quadros e conjuntos de regras de procedimento que podem ser implementados. Se os valores dos atributos forem apontadores para outros quadros (vizinho), cria-se uma rede de dependência entre os quadros.



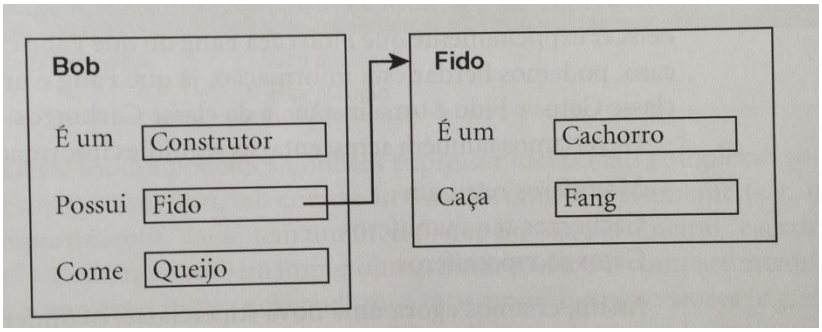
Representação baseada em quadros é um desenvolvimento de redes semânticas e nos permite expressar a ideia de herança.

Como as redes semânticas, um **sistema de quadros** consiste em um conjunto de quadros (ou vértices) que são interligados por relações. Cada **quadro** descreve uma instância (um **quadro instância**) ou uma classe (um **quadro classe**).

Cada quadro tem um ou mais **compartimentos** aos quais são atribuídos **valores de compartimento**. Em vez de ter somente ligações entre quadros, cada relacionamento é expresso por um valor sendo colocado em um compartimento:

Nome do Quadro	Compartimento	Valor do Compartimento
Bob	é um	Construtor
	possui	Fido
	come	Queijo
Fido	é um	Cachorro
	caça	Fang
Fang	é um	Gato
	caça	Ratos
Ratos	come	Queijo
Queijo		
Construtor		
Cachorro		
Gato		

Podemos também representar este sistema de quadros em uma forma diagramática utilizando representações como a mostrada a seguir:



O relacionamento "é-um" é muito importante em representações baseadas em quadros porque ele nos permite expressar pertinência a classes. Este relacionamento é também conhecido como **generalização**, já que referir-se à classe de mamíferos é mais geral que referir-se à classe de cachorros e referir-se à classe de cachorros é mais geral que em relação a referir-se a Fido.

É também útil ser capaz de falar de um objeto como parte de um outro. Por exemplo, Fido tem uma cauda e, assim, a cauda é parte de Fido. Este relacionamento é chamado de **agregação** porque Fido pode ser considerado um agregado de partes de cachorro.

Outros relacionamentos são conhecidos como **associação**. Um exemplo desse caso é o relacionamento "caça". Isto explica como Fido e Fang estão relacionados ou associados entre si. Relacionamentos de associação tem significado em duas direções. O fato de Fido caçar Fang, significa que Fang é caçado por Fido.

5.7.1 POR QUE QUADROS SÃO ÚTEIS?

Além de poder ser utilizados como uma estrutura de dados, a principal vantagem em utilizar sistemas baseados em quadros é que toda informação sobre um objeto específico é armazenada em um único lugar.

Em sistemas baseados em regras, informações sobre Fido deveriam ser armazenadas em diversas outras regras não relacionadas e, então, se Fido muda ou uma dedução precisa ser feita sobre Fido, será gasto tempo examinando regras e fatos irrelevantes no sistema, ao passo que no sistema de quadros, o quadro Fido poderia ser rapidamente examinado.

Em situações na qual objetos tenham muitas propriedades e muitos objetos estão relacionados entre si (como em muitas situações do mundo real), essa vantagem torna-se particularmente clara.

5.7.2 HERANÇA

Deveríamos estender nosso sistema de quadros com a seguinte informação adicional:

```
Cachorros caçam gatos
Gatos caçam ratos
```

Ao expressar esses fragmentos de informação, precisamos agora estabelecer explicitamente que Fido caça Fang ou que Fang caça ratos. Neste caso, podemos herdar esta informação, já que Fang é uma instância da classe Gatos e Fido é uma instância da classe Cachorros.

Deveríamos também acrescentar as seguintes informações adicionais:

```
Mamíferos respiram
Cachorros são mamíferos
Gatos são mamíferos
```

Assim, criamos agora uma nova superclasse, mamíferos, da qual cachorros e gatos são subclasses. Deste modo não precisamos expressar explicitamente que gatos e cachorros respiram porque podemos herdar esta informação. Da mesma forma, não precisamos expressar explicitamente que Fido e Fang respiram — eles são instâncias das classes Cachorros e Gatos e, portanto, eles herdam da superclasse dessas classes. Agora vamos acrescentar o seguinte fato:

```
Mamíferos têm quatro pernas
```

É claro, isto não é verdade, pois humanos não têm quatro pernas, por exemplo. Em um sistema baseado em quadros, podemos expressar que este fato é o valor-padrão e que ele pode ser sobrescrito. Imaginemos que Fido tenha tido um infeliz acidente e agora tenha apenas três pernas. Esta informação seria expressa assim:

Nome do Quadro	Compartimento	Valor do Compartimento
Mamífero	*número de pernas	quatro
Cachorro	subclasse	Mamífero
Gato	subclasse	Mamífero
Fido	é um	Cachorro
	número de pernas	três
Fang	é um	Gato

Aqui utilizamos um asterisco (*) para indicar que o valor para o compartimento "número de pernas" para a classe Mamífero é um valor-padrão e pode ser sobrescrito, como foi feito para Fido.

5.7.3 COMPARTIMENTO COMO QUADROS

É também possível expressar uma faixa de valores que um compartimento possa ter — por exemplo, o compartimento número de pernas poderia ter um número entre 1 e 4 (apesar de, para a classe insetos, poder ser 6). Um modo de expressar este tipo de restrição é permitir que compartimentos sejam quadros. Em outras palavras, o compartimento número de pernas pode ser representado como um quadro, que inclua informação sobre qual faixa de valores ele pode ter:

Nome do Quadro	Compartimento	Valor do Compartimento
Número de pernas	valor mínimo	1
	valor máximo	4

Deste modo, podemos também expressar ideias mais complexas sobre compartimentos, tais como o inverso de um compartimento (o compartimento "caça" tem um inverso que é o compartimento "caçado por"). Podemos também impor outras limitações a um compartimento, tais como especificar se ele pode ou não assumir diversos valores (o compartimento "número de pernas" deveria ter apenas um valor, ao passo que o compartimento "come" poderia assumir muitos valores).

5.7.4 HERANÇA MÚLTIPLA

É possível para um quadro herdar propriedades de mais de um quadro. Em outras palavras, uma classe pode ser uma subclasse de duas super-classes e um objeto pode ser uma instância de mais de uma classe. Isto é conhecido como **herança múltipla**.

Por exemplo, poderíamos acrescentar os seguintes quadros ao nosso sistema:

Nome do Quadro	Compartimento	Valor do Compartimento
Humano	Subclasse	Mamífero
	Número de pernas	dois
Construtor	Constrói	casas
Bob	é um	Humano

A partir disso, podemos ver que Bob é um humano, tanto quanto é um construtor. Assim, podemos herdar as seguintes informações sobre Bob:

```
Ele tem duas pernas
Ele constrói casas
```

Em alguns casos, encontraremos **conflitos**, quando a herança múltipla nos leva a concluir informações contraditórias sobre um quadro. Por exemplo, vamos considerar o seguinte simples sistema de quadros:

Nome do Quadro	Compartimento	Valor do Compartimento
Queijo	é	Fedorento
Coisa embrulhada em papel laminado	é	Não fedorento
Cheddar	é um	Queijo
	é um	Coisa embrulhada em papel laminado

(Observação: O compartimento "é" seria mais precisamente denominado “tem característica”)

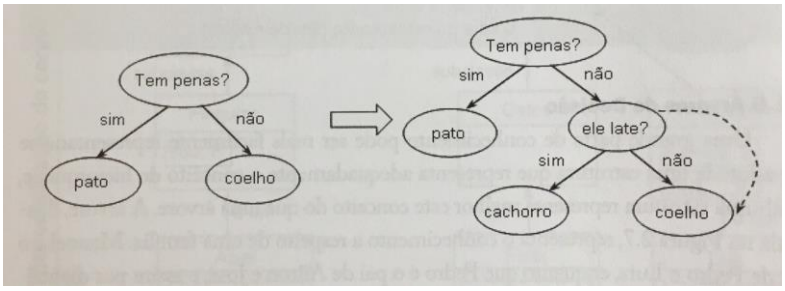
Aqui podemos ver que Cheddar é um tipo de Queijo e que ele veio embrulhado em papel laminado. Cheddar deveria herdar sua característica de cheiro da classe Queijo, mas ele também herda a falta de cheiro da classe Coisa embrulhada em papel laminado. Neste caso, precisamos de um **mecanismo para decidir qual característica herdar de cada superclasse**. Um **método elementar** é simplesmente dizer que os conflitos sejam resolvidos pela ordem em que eles aparecem. Então se um fato for estabelecido por herança e, em seguida, ele torna-se contraditório por herança, o primeiro fato é preservado por ter aparecido primeiro e a contradição é descartada.

Isto é certamente bastante arbitrário e seria bem melhor construir o sistema de quadros de modo a que conflitos desta natureza não ocorressem. Herança múltipla é uma característica-chave da maioria das linguagens de programação.

5.8 ÁRVORES DE DECISÃO

Uma grande parte de conhecimento pode ser mais facilmente representada se for adotada uma estrutura que representa adequadamente o conceito de hierarquia e, nenhuma estrutura representa melhor este conceito do que uma árvore.

Além da facilidade para representar conhecimento hierárquico, o uso de árvores para representar o conhecimento possui como vantagem a existência de várias técnicas de buscas que operam em árvores e, consequentemente, tem obtido sucesso em diversas aplicações. Por outro lado, tem como desvantagens o fato e que um mesmo conceito pode ser representando por árvores diferentes; árvores muito grandes se tornam difíceis de interpretar; dificultam a alteração (inserir e remover dados) e a representação de conhecimento não-hierárquico.



Apesar de simples, veja este programa que introduz o conceito de aprendizado, pois inicia com uma base de conhecimentos insignificante e vai acrescentando novos fatos à medida que interage com o usuário. Como ilustrado na figura: no início, o programa faz apena uma pergunta ao usuário: “o animal tem penas”? e no caso da resposta ser “sim”, dirá que o animal é um pato. Se a resposta for um “não”, dirá que o animal é um coelho. Nos dois casos, se o usuário disser que o animal não está correto, o programa pede que o usuário informe então, qual é o animal.

Se o usuário informar que o animal é um cachorro, o programa então, solicitará ao usuário que insira uma nova pergunta que poderia ser feita, para diferenciar um coelho de um cachorro, por exemplo, “ele late?”, então, a árvore é modificada como novo conhecimento acrescentado. Na próxima vez em que for executado, o sistema irá perguntar se o animal late e, se a resposta for sim, dirá que se trata de um cachorro, caso contrário, será um coelho. Caso erre novamente, irá repetir o processo de aprendizado, inserindo novas perguntas acima dos nós folhas da árvore, de modo que, após algum tempo, acumulará uma base de conhecimento a respeito de animais.

As árvores são estruturas muito importantes em IA.

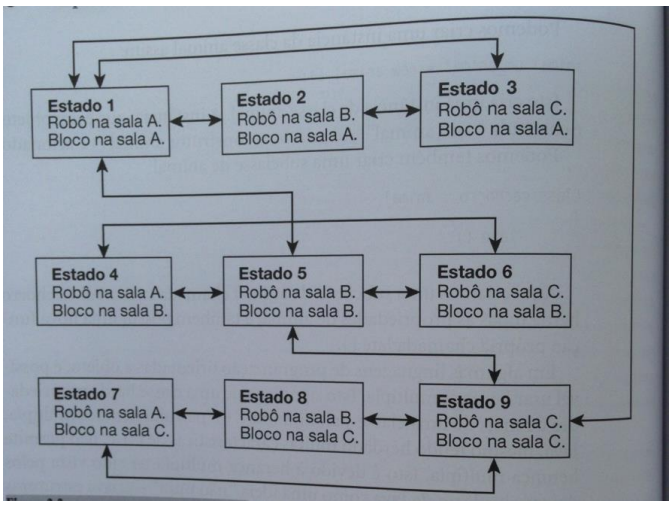
5.9 ESPAÇOS DE BUSCA

Muitos problemas em Inteligência Artificial podem ser representados como espaços de busca. De forma simples, um espaço de busca é uma representação do conjunto de possíveis escolhas de um dado problema, uma ou mais das quais é solução do problema.

Por exemplo, ao tentar localizar uma palavra específica em um dicionário com 100 páginas, um espaço de busca consistirá em cada uma das 100 páginas. A página que estiver sendo buscada é chamada de alvo e pode ser identificada verificando-se se a palavra que estamos procurando está ou não nela. (Na verdade, esta identificação seria um problema de busca em si mesmo, mas, para este exemplo, assumiremos que esta seja uma ação simples ou atômica.)

O objetivo da maioria dos procedimentos de busca é identificar um ou mais alvos e, geralmente, identificar um ou mais caminhos até estes alvos (frequentemente, o caminho mais curto ou o caminho de menor custo).

Devido a um espaço de busca consistir em um conjunto de estados, conectados por caminhos que representam ações, eles também são chamados de **espaço de estados**. Muitos problemas de busca podem ser representados por um espaço de estados onde o objetivo é começar com o mundo em um estado e terminar com o mundo em outro estado mais desejável. No problema dos Missionários e Canibais, o estado inicial tem todos os missionários e todos os canibais em uma margem do rio e o estado-alvo tem todos eles na outra margem. O espaço de estados do problema consiste em todos os possíveis estados intermediários.



A figura exibe um diagrama de espaço de estados muito simples para um robô que mora em um ambiente com três salas (sala A, sala B e sala C) e um bloco que pode ir de sala para sala. Cada estado consiste em um possível arranjo do robô e do bloco. Assim, por exemplo, no estado 1, tanto o robô quanto o bloco estão na sala A. Observe que este diagrama não explica como o robô vai de uma sala para a outra ou como um bloco é movido. Este tipo de representação assume que o robô tem a representação de diversas **ações** que ele pode realizar. Para determinar como passar de um estado para outro, o robô precisa utilizar um processo chamado **planejamento**.

Na figura as setas entre estados representam **transições de estados**. Observe que não há transições entre cada par de estados. Por exemplo, não é possível passar do estado 1 para o estado 4 sem passar pelo estado 5. Isto é devido ao bloco não poder se mover por conta própria e somente ser movido para uma sala se o robô for para lá. Assim, um diagrama de espaço de estados é um valioso modo de representar as possíveis ações que podem ser realizadas em um dado estado e então representar as possíveis soluções de um problema.

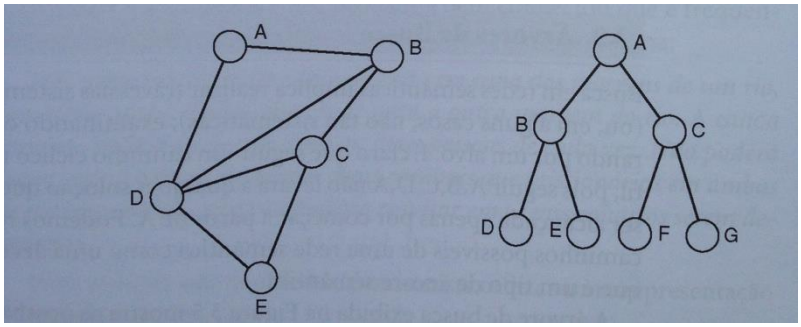
5.10 ÁRVORES SEMÂNTICAS

Uma **árvore semântica** é um tipo de rede semântica que tem as seguintes propriedades:

• Cada nó (exceto o nó raiz, descrito a seguir) tem exatamente um **predecessor** (pai) e um ou mais **sucessores** (filhos). Na árvore semântica da figura a seguir, o nó A é o predecessor do nó B: o nó A conecta-se por uma aresta ao nó B e vem antes dele na árvore. Os sucessores do nó B, nós D e E, conectam-se diretamente (por uma aresta cada) ao nó B e vêm depois dele na árvore. Podemos escrever estas relações como:

`suc(B) = D e pred(B) = A`

A natureza não simétrica deste relacionamento significa que uma árvore semântica é um grafo orientado. Em oposição, grafos não orientados são aqueles onde não há diferença entre um arco de A para B e um arco de B para A.



• Um nó não tem predecessores. Este nó é chamado de **raiz**. Em geral, ao fazer uma busca em uma árvore semântica, começa-se do nó raiz. Isto é devido ao nó raiz tipicamente representar um ponto inicial do problema. Por exemplo, ao analisar árvores de jogos, veremos que a árvore para o xadrez representa todos os movimentos possíveis do jogo, começando da posição inicial na qual ainda não foi feito movimento pelos jogadores. A posição inicial corresponde ao nó raiz na árvore de jogo.

• Alguns nós não têm sucessores. Esses nós são chamados de **folhas**. Uma ou mais folhas são chamadas de **alvos**. Estes são os nós que representam um estado no qual a busca foi bem-sucedida.

• Exceto as folhas, todos os nós têm um ou mais sucessores. Exceto a raiz, todos os nós têm exatamente um predecessor.

• Um **ancestral** de um nó é um nó acima dele em algum caminho na árvore. Um **descendente** vem depois de um nó em um caminho na árvore.

Um **caminho** é uma rota na árvore semântica, que pode consistir em apenas um nó (um caminho de comprimento 0). Um caminho de comprimento 1 consiste em um nó, um ramo que parte deste nó e o nó sucessor ao qual este ramo chega. Um caminho que parta da raiz e vá até um alvo é chamado de **caminho completo**. Um caminho que parta da raiz e vá até uma folha que não seja alvo é chamado de **caminho parcial**.

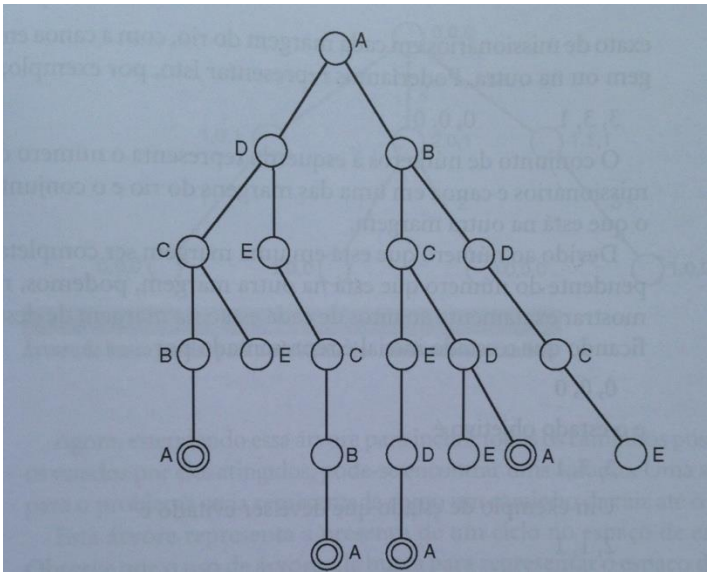
Ao comparar redes semânticas e árvores semânticas visualmente, uma das diferenças mais óbvias é que redes semânticas podem conter ciclos, mas árvores semânticas não. Um **ciclo** é um caminho pela rede que visita o mesmo vértice. A figura anterior exibe uma rede semântica e uma árvore semântica. Na rede semântica, o caminho A, B, C, D, A... é um ciclo. Nas árvores semânticas, uma aresta que conecte dois nós é chamada de **ramo**. Se um nó tiver n sucessores, diz-se que esse nó tem **fator de ramificação** de *n*. Diz-se que uma árvore tem fator de ramificação de *n* se a média dos fatores de ramificação de todos os nós for *n*.

Diz-se que a raiz de uma árvore está no nível 0 e os sucessores da raiz estão no nível 1. Os sucessores dos nós do nível *n* estão no nível *n + 1*.

5.11 ÁRVORES DE BUSCA

Busca em redes semânticas implica realizar travessias sistemáticas na rede (ou, em alguns casos, não tão sistemáticas), examinando os nós, procurando por um alvo. É claro que seguir um caminho cíclico na rede é inútil, pois seguir A, B, C, D, A não levará a qualquer solução que não pudesse ser alcançada apenas por começar

a partir de A. Podemos representar os caminhos possíveis de uma rede semântica como uma **árvore de busca**, que é um tipo de árvore semântica.



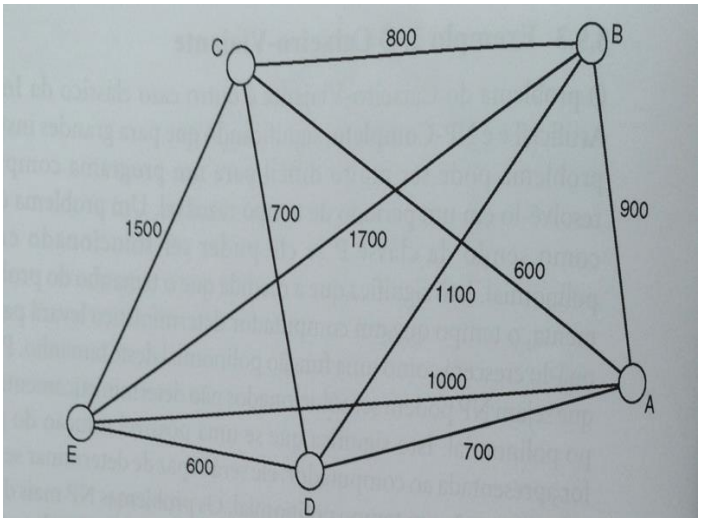
A árvore de busca exibida na figura mostra os possíveis caminhos pela rede semântica exibida na figura anterior. Cada nó na árvore representa um caminho, com sucessivas camadas na árvore representando caminhos cada vez mais longos. Observe que não incluímos caminhos cíclicos, o que significa que alguns ramos na árvore de busca terminam em folhas que não são alvos. Observe também que rotulamos cada nó na árvore de busca com uma única letra, que representa o caminho a partir da raiz até aquele nó na rede semântica.

Assim, buscar um nó na árvore de busca corresponde a buscar um caminho completo na rede semântica.

5.11.1 EXEMPLO 1: O CAIXEIRO-VIAJANTE

O problema do Caixeiro-Viajante é outro caso clássico da Inteligência Artificial e é **NP-Completo**, significando que para grandes instâncias do problema pode ser muito difícil para um programa computacional resolvê-lo em um período de tempo razoável. Um problema é definido como sendo da classe P se ele puder ser solucionado em tempo polinomial. Isto significa que à medida que o tamanho do problema aumenta, o tempo que um computador determinístico levará para solucioná-lo crescerá como uma função polinomial deste tamanho. Problemas que sejam NP podem ser solucionados não deterministicamente em tempo polinomial. Isto significa que se uma possível solução do problema for apresentada ao computador, ele será capaz de determinar se ela é uma solução ou não em tempo polinomial. Os problemas NP mais difíceis são chamados de NP-Completos. Foi demonstrado por Stephen Cook que um grupo particular de problemas pode ser transformado no problema da satisfatoriedade. Estes problemas são definidos como NP-Completos. Isto significa que se alguém puder solucionar o problema da satisfatoriedade (para o qual soluções certamente existem) poderá então resolver qualquer problema NP-Completo. Também significa que problemas NP-Completos exigem grande esforço computacional para serem resolvidos.

Eis a definição do problema do Caixeiro-Viajante:



Um caixeiro viajante deve visitar cada uma das cidades de um conjunto de cidades e retornar à cidade de partida. O objetivo do problema é encontrar o caminho mais curto que permita que ele visite cada uma das cidades.

Imaginemos que o caixeiro-viajante esteja percorrendo as seguintes cidades americanas e as distâncias podem ser mostradas em um grafo como na figura.

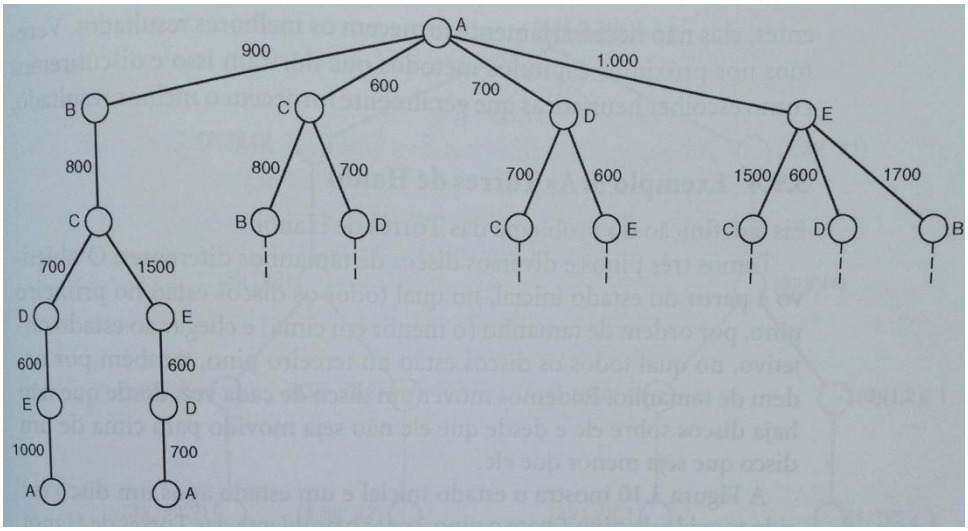
```
A Atlanta
B Boston
C Chicago
D Dallas
E El Paso
```

Ele mora em Atlanta e deve visitar todas as outras quatro cidades antes de retornar para casa. Imaginemos que este caixeiro-viajante esteja viajando de avião e que o custo de cada voo seja diretamente proporcional à distância percorrida e que haja voos diretos entre qualquer par de cidades.

O grafo exibe o relacionamento entre as cidades. Poderíamos utilizar este grafo para tentar solucionar o problema. Certamente, podemos utilizá-lo para encontrar possíveis caminhos: um possível caminho é A,B,C,E,D,A que tem um comprimento de 4.500 milhas.

Para solucionar o problema por meio de busca, uma representação será necessária, baseada neste grafo. A figura seguinte mostra uma parte da árvore de busca que representa os possíveis caminhos no espaço de estados deste problema. Cada nó é marcado com a letra que representa a cidade que foi alcançada pelo caminho até aquele ponto. Assim, na verdade, cada nó representa o caminho desde a cidade A até a cidade nomeada naquele nó. A raiz da árvore, assim, representa o caminho de comprimento 0, que consiste apenas na cidade A. Como no exemplo anterior, caminhos cíclicos foram excluídos da árvore, mas diferente da árvore para o problema dos Missionários e Canibais, esta árvore permite estados repetidos. Isto porque, neste problema, cada estado deve ser visitado uma vez e, assim, um caminho completo deve incluir todos os estados. No problema dos Missionários e Canibais o objetivo era chegar a um estado específico pelo caminho mais curto que pudesse ser encontrado. Assim, incluir um caminho como A,B,C,D se um caminho A,D já tiver sido encontrado seria perda de tempo, pois ele não poderia levar a um caminho mais curto que A,D. Já no problema do Caixeiro-Viajante, isto não se aplica e precisamos examinar cada possível caminho que inclua cada nó uma vez, com o nó inicial no começo e no fim.

A figura a seguir é apenas uma parte da árvore de busca, mas ela mostra dois caminhos completos: A,B,C,D,E,A e A,B,C,E,D,A. O gasto total desses dois caminhos é de 4.000 milhas e de 4.500 milhas, respectivamente.



Ao todo existirão (n - 1)! possíveis caminhos para o problema do Caixeiro-Viajante com n cidades. Isto porque temos a restrição da cidade inicial e, a seguir, uma escolha de qualquer combinação de (n - 1) cidades. Para problemas com um pequeno número de cidades, como 5 ou mesmo 10, isto significa que a árvore de busca completa pode ser avaliada por um programa computacional sem maiores dificuldades; mas se o problema consistir em 40 cidades, seriam 40! caminhos, cerca de 10^{48}, um número absurdamente grande. Como veremos, métodos que tentem examinar todos esses caminhos são chamados de métodos de **busca força-bruta**. Para solucionar problemas com grandes árvores, é necessário aplicar conhecimento sobre o problema em forma de **heurísticas**, que nos permitam encontrar modos mais eficientes para esta solução. Uma heurística é uma regra ou fragmento de informação que são utilizadas para tornar a busca ou um outro método de solução de problemas mais efetivo ou mais eficiente.

Por exemplo, uma abordagem de busca heurística para solucionar o problema do Caixeiro-Viajante seria: em vez de examinar cada possível caminho, simplesmente estendemos o caminho indo para a cidade mais perto da posição atual que ainda não foi visitada. Esta é chamada de **heurística do vizinho mais próximo**. No

exemplo anterior, isto levaria ao caminho A, C, D, E, B, A, que tem gasto total de 4.500 milhas. Certamente, este não é o melhor caminho possível, pois já vimos um caminho (A, B, C, D, E, A) que tem um gasto de 4.000 milhas. Isto ilustra o fato de que, apesar de heurísticas poderem tornar as buscas bem mais eficientes, elas não necessariamente fornecem os melhores resultados.

5.11.2 EXEMPLO 2: MISSIONÁRIOS E CANIBAIS

Missionários e Canibais é um problema bem conhecido que é frequentemente utilizado para ilustrar técnicas de IA. Eis o problema:

Três missionários e três canibais estão em uma das margens de um rio, com uma canoa. Todos querem ir para a outra margem do rio. A canoa somente pode transportar uma ou duas pessoas de cada vez. Não poderá haver, em qualquer momento, mais canibais que missionários em ambas as margens do rio, pois isto poderia resultar em os missionários serem devorados.

Para resolver este problema, precisamos utilizar uma representação adequada. Em primeiro lugar, para a solução do problema, podemos considerar um estado como consistindo no número exato de canibais e no número exato de missionários em cada margem do rio, com a canoa em uma margem ou na outra. Poderíamos representar isto, por exemplo, como

3, 3, 1 0, 0, 0

O conjunto de números à esquerda representa o número de canibais, missionários e canoa em uma das margens do rio e o conjunto à direita, o que está na outra margem.

Devido ao número que está em uma margem ser completamente dependente do número que está na outra margem, podemos, na verdade, mostrar exatamente quantos de cada estão na margem de destino, significando que o estado **inicial** é representado por **0, 0, 0** e o estado **objetivo** é **3, 3, 1**. Um exemplo de estado que deve ser **evitado** é **2, 1, 1**: aqui, são dois canibais, uma canoa e apenas um missionário em uma das margens do rio. Este missionário provavelmente não vai durar muito.

Para ir de um estado a outro, devemos aplicar um operador. Os operadores que temos disponíveis são os seguintes:

1. Levar um canibal para a outra margem
2. Levar dois canibais para a outra margem
3. Levar um missionário para a outra margem
4. Levar dois missionários para a outra margem
5. Levar um canibal e um missionário para a outra margem

Então se aplicássemos o operador 5 ao estado representado por 1, 1, 0 resultaria no estado 2, 2, 1. Um canibal, um missionário e a canoa deslocaram-se agora para a outra margem. Aplicar o operador 3 a este estado levaria a um estado ilegal: 2, 1, 0.

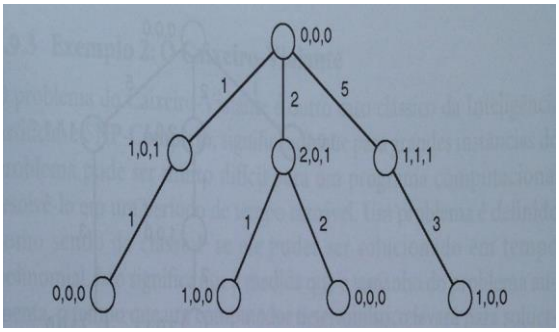
Consideramos regras desse tipo como **restrições**, que limitam os possíveis operadores aplicáveis a cada estado. Se projetarmos a nossa representação corretamente, as restrições estarão embutidas, significando que nunca precisaremos examinar estados ilegais.

Precisamos ter um teste que possa identificar se o estado objetivo foi atingido — 3, 3, 1.

Consideraremos o custo do caminho escolhido como sendo o número de passos que foram dados ou o número de vezes que um operador foi aplicado. Em alguns casos, como veremos mais tarde, é desejável encontrar uma solução que minimize o custo.

Os três primeiros níveis da árvore de busca para o problema dos missionários e canibais são exibidos na figura abaixo (os arcos estão marcados com o operador que foi aplicado).

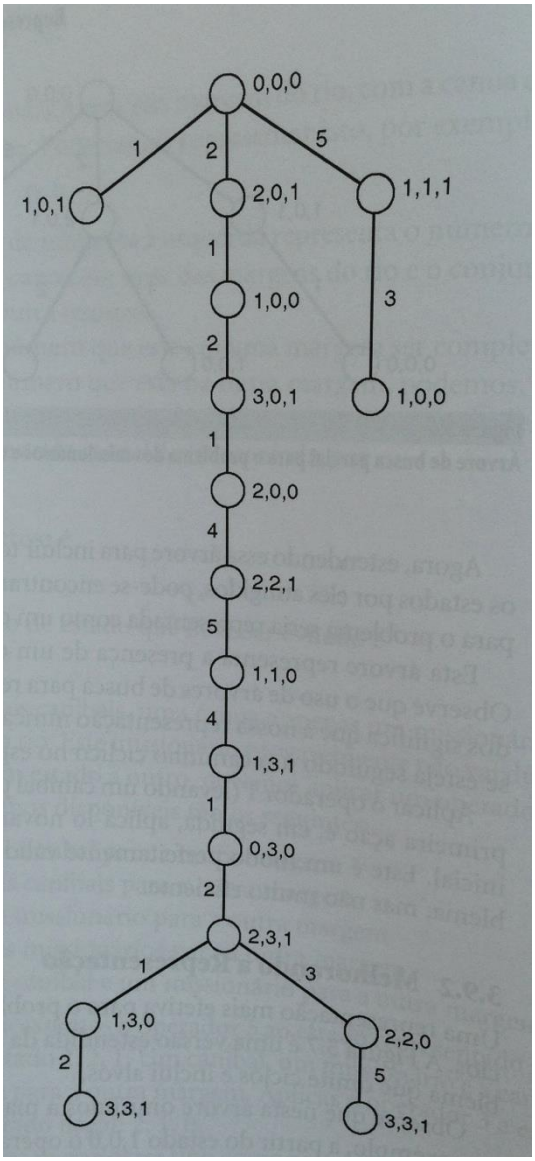
Agora, estendendo essa árvore para incluir todos os caminhos possíveis e os estados por eles atingidos, pode-se encontrar uma solução. Uma solução para o problema seria representada como um caminho da raiz até o alvo



Esta árvore representa a presença de um ciclo no espaço de estados. Observe que o uso de árvores de busca para representar o espaço de estados significa que a nossa representação nunca conterá ciclos, mesmo que se esteja seguindo um caminho cíclico no espaço de estados.

Aplicar o operador 1 (levando um canibal para a outra margem) como primeira ação e, em seguida, aplicá-lo novamente, retornará ao estado inicial. Este é um modo perfeitamente válido de tentar resolver o problema, mas não muito eficiente.

MELHORANDO A REPRESENTAÇÃO



Uma representação mais efetiva para o problema não deveria incluir ciclos. A figura a ao lado é uma versão estendida da árvore de busca para o problema que omite ciclos e inclui alvos.

Observe que nesta árvore omitimos a maioria dos estados repetidos. Por exemplo, a partir do estado 1,0,0 o operador 2 é o único exibido. Na verdade, os operadores 1 e 3 também podem ser aplicados, levando aos estados 2,0,1 e 1,1,1, respectivamente. Nenhuma dessas transições é exibida já que estes estados já apareceram na árvore.

Além de evitar ciclos, caminhos subótimos também foram removidos da árvore. Se um caminho de comprimento 2 atinge um estado específico e outro caminho de comprimento 3 também atinge aquele estado, não há interesse em seguir o caminho mais longo pois ele não levaria ao alvo por um caminho mais curto que o primeiro.

Assim, os dois caminhos que podem ser seguidos até o alvo, na árvore da figura, são as rotas mais curtas (os caminhos de menor custo) até o alvo, mas não são, de forma alguma, os únicos caminhos. Existem muitos caminhos mais longos.

Ao escolher uma representação adequada, somos capazes assim de melhorar a eficiência do método de busca. É claro, em implementações reais, as coisas podem não ser assim tão simples. Para produzir a árvore de busca sem estados repetidos, será necessária uma memória que possa armazenar os estados de modo a evitar visitá-los de novo.

É provável que para a maioria dos problemas esta exigência de memória seja uma barganha compensável pela economia de tempo, especialmente se o espaço de estados sendo explorado tiver muitos estados repetidos e ciclos.

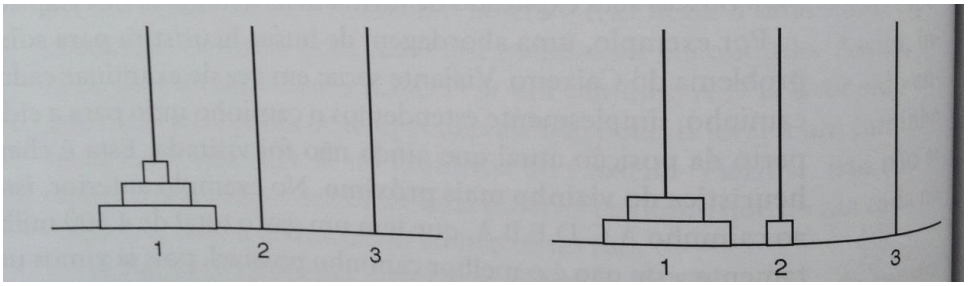
Resolver o problema dos Missionários e Canibais envolve **buscar** na árvore de busca. Como veremos, busca é um método extremamente útil para solucionar problemas e é amplamente utilizada em Inteligência Artificial.

5.11.3 EXEMPLO 3: AS TORRES DE HANÓI

Eis a definição do problema das Torres de Hanói:

Temos três pinos e diversos discos de tamanhos diferentes. O objetivo é partir do estado inicial, no qual todos os discos estão no primeiro pino, por ordem de tamanho (o menor em cima) e chegar ao estado objetivo, no qual todos os discos estão no terceiro pino, também por ordem de tamanho. Podemos mover um disco de cada vez, desde que não haja discos sobre ele e desde que ele não seja movido para cima de um disco que seja menor que ele.

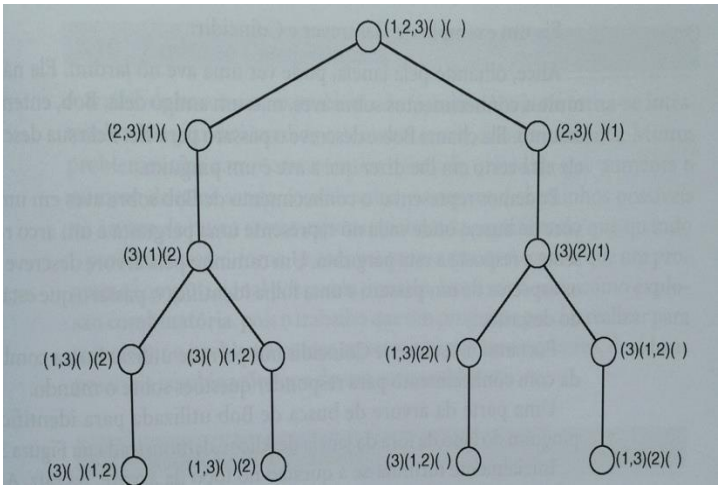
A figura abaixo mostra o estado inicial e um estado após um disco ter sido movido do pino 1 para o pino 2 para o problema das Torres de Hanói com três discos.



Agora que conhecemos os aspectos do estado inicial e do estado objetivo, precisamos apresentar um conjunto de operadores:

```
Op1   Mover disco do pino 1 para o pino 2
Op2   Mover disco do pino 1 para o pino 3
Op3   Mover disco do pino 2 para o pino 1
Op4   Mover disco do pino 2 para o pino 3
Op5   Mover disco do pino 3 para o pino 1
Op6   Mover disco do pino 3 para o pino 2
```

Também precisamos um modo de representar cada estado. Para este exemplo, utilizaremos vetores de números, nos quais 1 representa o menor disco e 3, o maior disco. O primeiro vetor representa o primeiro pino e assim por diante. Assim, o estado **inicial** é representado por **(1,2,3) ( ) ( ).** O segundo estado mostrado na figura é representado por (2,3) (1) ( ) e o estado **objetivo** é **( ) ( ) (1,2,3)**



Os primeiros cinco níveis da árvore de busca do problema das Torres de Hanói com três discos são mostrados na figura a seguir. Mais uma vez, ignoramos caminhos cíclicos. Na verdade, no problema das Torres de Hanói, a cada passo, podemos sempre escolher inverter a ação anterior. Por exemplo, tendo aplicado o operador Op1, para ir do estado inicial para o estado (2,3) (1) ( ), podemos agora aplicar o operador Op3, que inverte este movimento e nos leva de volta ao estado inicial. É claro que este comportamento sempre levará a um ciclo e, portanto, ignoramos estas escolhas na nossa representação.

A busca não é o único modo de identificar soluções para problemas do tipo das Torres de Hanói. Um método de busca encontraria uma solução examinando cada possível conjunto de ações até que um caminho fosse encontrado, do estado inicial até o estado objetivo. Um sistema mais inteligente desenvolvido, que entendesse mais sobre o problema e, na verdade, entendesse como solucioná-lo sem necessariamente ter que examinar qualquer dos caminhos alternativos poderia ser.

5.12 EXPLOSÃO COMBINATÓRIA

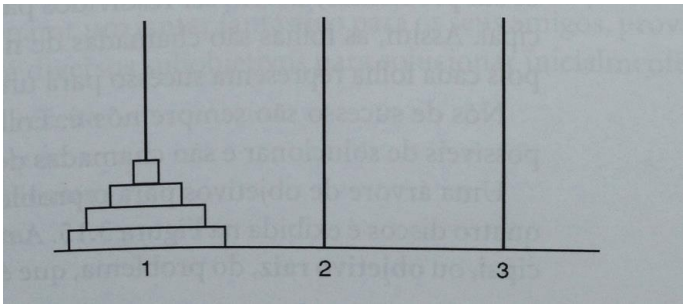
A árvore de busca para o problema do Caixeiro-Viajante torna-se intratavelmente grande à medida que o número de cidades aumenta. Muitos problemas têm como característica o fato de, à medida que aumenta o número de itens sendo considerados, o número de caminhos possíveis na árvore de busca cresce **exponencialmente**, significando que quando o problema cresce, torna-se cada vez menos razoável esperar que um programa computacional possa solucioná-lo. Isto é conhecido como **explosão combinatória**, pois o trabalho que um programa precisa realizar para solucionar o problema parece crescer sob uma taxa explosiva, devido às possíveis combinações que devem ser consideradas.

5.13 REDUÇÃO DE PROBLEMA

Em alguns casos achamos que um problema complexo pode ser efetivamente resolvido subdividindo-o em pequenos problemas. Se resolvermos todos esses **subproblemas** menores, então teremos resolvido o problema principal. Esta abordagem para a solução de problemas é chamada de **redução de objetivo**, pois envolve considerar o objetivo final da solução do problema de um modo que se possa gerar subobjetivos para aquele objetivo.

Por exemplo, para resolver o problema das Torres de Hanói com *n* discos, observa-se que o primeiro passo é solucionar o problema menor com *n* – 1 discos.

Por exemplo, vamos examinar as Torres de Hanói com quatro discos, cujo estado inicial é mostrado a seguir.



Para resolver este problema, o primeiro passo é mover o maior disco do pino 1 para o pino 3. Isto levará então a um problema das Torres de Hanói de tamanho 3, como mostrado na figura a seguir, onde o objetivo é mover os discos do pino 2 para o pino 3. Devido ao disco que está no pino 3 ser o maior disco, qualquer outro disco pode ser colocado sobre ele e, como já está na sua posição final, ele pode ser efetivamente ignorado.

Deste modo, um problema das Torres de Hanói de qualquer tamanho *n* pode ser solucionado movendo-se inicialmente o maior disco para o pino 3, aplicando-se então a solução das Torres de Hanói aos discos restantes, mas trocando o pino 1 pelo pino 2.