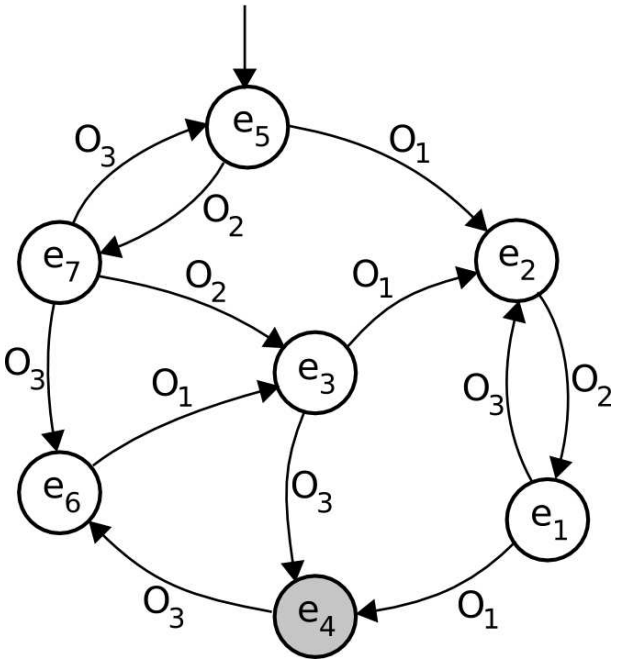
Introdução à Inteligência Artificial



Fabricio Ferrari

fabricio@ferrari.pro.br

www.ferrari.pro.br

2005

Sumário

Capítulo 1

Introdução

1.1 Faculdades Psíquicas Fundamentais

A inteligência envolve várias atividades psíquicas, cujas principais são [Na2000]:

- **sensação:** fenômeno elementar resultante de estímulos mecânicos, físicos, químicos ou elétricos sobre o organismo.
- **percepção:** tomada de conhecimento de um objeto exterior considerado real.
- **representação:** organização das imagens dos objetos.
- **conceituação:** construção simbólica a partir da essência dos objetos, agrupando-os em classes.
- **juízo:** capacidade de exprimir os vínculos entre fatos e objetos da natureza.
- **raciocínio:** concatenação ordenada dos juízos, gerando novos juízos.
- **memória:** capacidade de armazenamento de informações.
- **atenção:** capacidade de concentrar a atividade psíquica sobre um dado estímulo.
- **consciência:** conjunto de fenômenos psíquicos que permitem conhecer a si mesmo e ao mundo.
- **orientação:** consciência da situação espacial e temporal em relação a um meio.

- **afetividade:** capacidade de experimentar sentimentos e emoções, que corresponde à valorização qualitativa de determinados estados do indivíduo.
- **volição:** elemento gerador de atividades voluntárias.
- **linguagem:** mecanismo de expressão simbólica.

1.2 Inteligência

Definições[1]:

- **Capacidade mental geral que envolve a capacidade de discutir, planejar, solucionar problemas, pensar abstratamente, compreender idéias e linguagem e aprender.**
- Habilidade de compreender, de entender e de beneficiar-se da experiência.
- Habilidade de identificar similaridades e diferenças. Habilidade de reconhecer a importância relativa de algo.
- Habilidade de entender e lidar com o meio-ambiente.
- Capacidade de criar construtivamente com o objetivo de ganho evolucionário. Habilidade de reconhecer o que é útil e o que não é, nas mudanças internas e externas.
- Termo genérico de várias habilidades cognitivas: velocidade de aprendizado, memória, criatividade e raciocínio para processar conteúdo numérico, verbal e figurativo.
- Qualidade de solução e velocidade em resolver tarefas completamente novas; habilidade de aprender;

1.3 Inteligência artificial

Definições[2]:

[1] `http://www.google.com/search?q=describe:intelligence`
[2] `http://www.google.com/search?q=describe:artificial%20intelligence`

- **Área multidisciplinar englobando ciência da computação, neurociência, filosofia, psicologia, robótica e linguística, dedicada a reproduzir os métodos ou resultados do raciocínio humano e da atividade cerebral.**
- Um ramo da ciência que estuda como dotar os computadores com as capacidades da inteligência humana.
- Um ramo da ciência que tenta programar computadores para responder como se eles estivessem pensando – capazes de raciocinar, adaptar-se a novas situações e aprender novas habilidades.
- O uso de programas que capacitam maquinas a desempenhar tarefas que os humanos desempenham usando sua inteligência.
- Fazer uma máquina comportar-se com modos que seriam chamados inteligentes se um humano estivesse assim se comportando.
- Um campo de estudo cujo objetivo é entender e construir máquinas inteligentes.
- A habilidade de um computador realizar tarefas, tais como raciocinar e aprender, que a inteligência humana é capaz de fazer.
- Ferramenta que exibe inteligência e comportamento humano incluindo robôs com auto-aprendizagem, sistemas especialistas, reconhecimento de voz, tradução natural e automatizada.
- Programa de computador que mimetiza a capacidade de aprendizado humana.
- Um algoritmo pelo qual o computador dá a ilusão da capacidade de aprendizagem humana.
- O esforço para automatizar as habilidade humanas que correspondem à sua inteligência.
- Técnicas computacionais para automatizar tarefas que requerem inteligência humana e habilidade para raciocinar.
- Um ramo da ciência da computação que se ocupa de criar ou mimetizar o comportamento inteligente ou *pensamento* nos computadores.

1.4 Teste de Turing

Proposto por Alan Turing 1950 para fornecer uma definição operacional de inteligência artificial. Consiste no seguinte:

Um entrevistador, através de um meio impessoal, pergunta a um sistema (que ele não sabe o que é) uma série de perguntas de qualquer espécie. Se o entrevistador não souber discernir se o sistema que responde é homem ou máquina, este sistema é dito possuir **inteligência artificial**.

Um sistema com inteligência artificial deve possuir as seguintes características [Ru2004]:

- **Processamento de linguagem natural** para que seja possível comunicar-se num idioma natural.
- **Representação do conhecimento** para armazenar o que sabe ou aprende.
- **Raciocínio** para usar as informações armazenadas para responder perguntas e tirar novas conclusões.
- **Aprendizado** para se adaptar a novas circunstâncias, identificar e usar padrões.

1.5 Fases históricas

As principais fases históricas de desenvolvimento da IA podem ser divididas de acordo com os seus objetivos, métodos e limitações [Bt2001]:

Clássica 1956-1970

- **Objetivo:** simular a inteligência humana.
- **Métodos:** solucionadores gerais de problemas (*GPS*) e lógica.
- **Limitação:** subestimação da complexidade computacional dos problemas.

Romântica 1970-1980

- **Objetivo:** simular a inteligência humana em situações pré-determinadas.
- **Métodos:** formalismo de representação de conhecimento adaptados aos tipos de problema, mecanismos de ligação procedural visando maior eficiência computacional.

- **Limitação:** subestimação da quantidade de conhecimento necessária mesmo para tratar de problemas muito simples.

Moderna 1980-1990

- **Objetivo:** simular o comportamento de um especialista humano num domínio específico.
- **Métodos:** sistemas de regras, representação da incerteza, conexionismo.
- **Limitação:** subestimação da complexidade do problema de aquisição de conhecimento.

1.6 Aplicações

- jogo de xadrez, jogos em geral.
- prova de teoremas.
- reconhecimento de voz.
- tradução automática de textos.
- logística de transportes, hospitais, grandes organizações.
- atendimento ao cliente.
- mineração de dados.
- diagnóstico.

Capítulo 2

Lógica

2.1 Lógica proposicional

Lógica proposicional é a lógica no nível das sentenças. A menor unidade da lógica proposicional é a **sentença elementar**, **o átomo**. A lógica proposicional não discute o significa das sentenças elementares, os átomos, mas se as **sentenças complexas**, formadas por sentenças elementares, são falsas ou verdadeiras a partir das sentenças elementares [To2005].

As sentenças consideradas na lógica proposicional são aquelas que são verdadeiras ou falsas, mas não ambos. Estas sentenças são chamadas proposições. Quando uma proposição é verdadeira, diz-se que ela tem um valor verdade de *verdadeiro*; se uma proposição é falsa seu valor verdade é *falso*.

A Tabela 2.1 mostra alguns exemplos de sentenças que são e não são proposições. Observe que nas sentenças complexas não interessa a veracidade dos átomos.

O céu é azul	Proposição (V)
$2 + 5 = 5$	Proposição (F)
Feche a porta.	Não é proposição
Está chovendo?	Não é proposição
Se (chover) então a (rua está molhada)	Proposição

Tabela 2.1: Sentenças analisadas com a lógica proposicional.

2.1.1 Elementos da lógica proposicional

As sentenças elementares (átomos) são as proposições básicas. As sentenças maiores e mais complexas são construídas a partir das proposições básicas

combinando-as com **conectivos**. Proposições e conectivos são os elementos básicos da lógica proposicional. A Tabela 2.2 mostra os conectivos básicos.

Conectivo	**Símbolo**
E	$\wedge$
OU	$\vee$
NÃO	$\neg$ ou $\sim$
IMPLICA (SE_ENTÃO)	$\longrightarrow$
EQUIVALE (SE_SOMENTE_SE)	$\longleftrightarrow$

Tabela 2.2: Conectivos básicos

Para estudar as propriedades e relações comuns a todas as proposições utilizamos **variáveis proposicionais** que representam proposições e que podem ter valor *verdadeiro* ou *falso*. Um proposição adquire valor *verdadeiro* ou *falso* dependendo do valor das suas sentenças elementares e dos conectivos usados. É útil construir uma tabela com todos os valores de uma proposição complexa a partir dos valores das proposições básicas; tal tabela é chamada de **tabela verdade**.

P	Q	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$	$\neg P$
$\mathbb{V}$	$\mathbb{V}$	$\mathbb{V}$	$\mathbb{V}$	$\mathbb{V}$	$\mathbb{V}$	$\mathbb{F}$
$\mathbb{V}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{V}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{F}$
$\mathbb{F}$	$\mathbb{V}$	$\mathbb{F}$	$\mathbb{V}$	$\mathbb{V}$	$\mathbb{F}$	$\mathbb{V}$
$\mathbb{F}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{F}$	$\mathbb{V}$	$\mathbb{V}$	$\mathbb{V}$

Tabela 2.3: Tabela verdade

2.1.2 Sintaxe

A sintaxe define as regras de como as proposições complexas são construídas a partir das proposições mais simples. Na linguagem natural, combinando *"o céu é azul"* com *"o cachorro é branco"* podemos formar *"o céu é azul e o cachorro é branco"*, *"se o céu é azul, então o cachorro é branco"*, *"o céu é azul e o cachorro não é branco"*, etc. Deste modo, proposições complexas são formadas a partir de proposições simples e podem ser compostas para formar outras mais complexas ainda. Este processo de obter proposições mais e mais complexas pode ser resumido como segue.

Se X e Y são proposições arbitrárias, então

$$(\neg X), \quad (X \wedge Y), \quad (X \vee Y), \quad (X \rightarrow Y), \quad (X \leftrightarrow Y)$$

também são proposições.

Exemplos de proposições:

$$(P \wedge \neg Q), \qquad ((P \vee Q) \rightarrow (Q \wedge R), \qquad (\neg\neg P \vee Q)$$

Exemplos que não são proposições:

$$\neg(P \leftrightarrow \wedge Q) \vee Q \qquad ((Q \vee R)\neg(P \wedge Q)) \qquad (\neg(PQ \wedge R) \wedge (S \rightarrow R))$$

2.1.3 Representação lógica da linguagem natural

O raciocínio é feito com proposições usando regras de inferência. Por exemplo, das duas proposições *"se chove então a rua alaga"* e *"chove"* são verdade, então pode-se concluir que *"a rua alaga"* é verdade.

Para verificar se o raciocínio é correto, deve-se observar se as regras de inferência foram seguidas ao tirar-se conclusões a partir das premissas. Nas linguagens naturais, este cuidado nem sempre é seguido com rigor, o que pode dificultar a representação da linguagem natural numa proposição lógica.

A representação lógica do raciocínio é feita usando-se os símbolos e os conectivos. Suponha que P representa *"chove"* e Q representa *"a rua alaga"*. Então pode-se representar o argumento anterior como

$$((P \rightarrow Q) \wedge P) \rightarrow Q$$

ou alternativamente

$$\frac{\begin{array}{l} P \rightarrow Q \\ P \end{array}}{Q}$$

Estas expressões são muito mais simples de manipular do que a sentença em linguagem natural.

Para representar uma proposição de uma linguagem natural como uma proposição lógica, é preciso reescrever a sentença de forma que fique dividida em proposições elementares e conectivos da lógica proposicional (*não, e, ou, se_ então, se_ e_ somente_ se*). Atribui-se símbolos para os átomos e escreve-se a proposição como uma expressão de lógica proposicional.

Por exemplo, suponha que P representa *"está chovendo*, Q seja *"eu vou à praia"* e R *"eu tiver tempo "*. Então pode-se escrever

$$\neg P \rightarrow Q \qquad \text{(se não chover eu vou à praia)}$$

ou ainda

$$(\neg P \wedge R) \rightarrow Q \qquad \text{(se não chover e eu tiver tempo eu vou à praia)}$$

Variações equivalentes de SE_ENTÃO

Para facilitar a representação lógica de uma linguagem natural usa-se as versões logicamente equivalentes do conectivo lógico *se-então*, isto é, se uma das expressões abaixo for verdadeira, todas serão; se uma for falsa, todas serão.
$P \rightarrow Q$
Se P, então Q.
P implica Q.
Se P, Q.
Q somente se P.
P é suficiente para Q
Q se P
Q sempre que P
Q é necessário para P
É necessário para Q que P
Q somente se P

Por exemplo, leia as proposições acima substituindo P por *"estar feliz"* e Q por *"sorrir."*

2.1.4 Tipos de proposição

Tautologia é uma proposição que é sempre verdade independente do valor dos seus constituintes. Ex.: $(P \vee \neg P)$.

Contradição é uma proposição que é sempre falsa independente do valor dos seus constituintes. Ex.: $(P \wedge \neg P)$.

Contingência é uma proposição que não é nem tautologia nem contradição. Ex.: $(P \wedge Q)$.

2.1.5 Tautologias ou Identidades

A partir da definição dos conectivos, várias relações entre as proposições podem ser derivadas. São chamadas **identidades** ou **tautologias** pois tem seu valor definido independentemente do valor dos constituintes. Se duas proposições são logicamente equivalente, uma pode ser substituída pela outra na proposição em que ocorrem sem afetar o valor lógico deste da proposição.

A Tabela 2.4 mostra as principais tautologias ou identidades. São úteis principalmente para transformar expressões lógicas complexas em outras mais simples. A seguir estão exemplificadas as principais destas tautologias.

Nestas sentenças, $\Leftrightarrow$ corresponde a $\leftrightarrow$ mas **também** que a proposição é verdadeira, pois $\leftrightarrow$ pode significar equivalência entre dois valores falsos. Do mesmo modo $\mathcal{V}$ e $\mathcal{F}$ significam verdadeiro e falso, respectivamente. Os conectivos estão sublinhados para salientar a sua posição nas proposições.

1	$P \rightarrow P$	Identidade
2	$P \leftrightarrow P$	Equivalência
3	$(P \vee P) \Leftrightarrow P$	Idempotência de $\vee$
4	$(P \wedge P) \Leftrightarrow P$	Idempotência de $\wedge$
5	$(P \vee Q) \Leftrightarrow (Q \vee P)$	Comutatividade de $\vee$
6	$(P \wedge Q) \Leftrightarrow (Q \wedge P)$	Comutatividade de $\wedge$
7	$[(P \vee Q) \vee R] \Leftrightarrow [P \vee (Q \vee R)]$	Associatividade de $\vee$
8	$[(P \wedge Q) \wedge R] \Leftrightarrow [P \wedge (Q \wedge R)]$	Associatividade de $\wedge$
9	$\neg(P \vee Q) \Leftrightarrow (\neg P \wedge \neg Q)$	Lei de De Morgan
10	$\neg(P \wedge Q) \Leftrightarrow (\neg P \vee \neg Q)$	Lei de De Morgan
11	$[P \wedge (Q \vee R)] \Leftrightarrow [(P \wedge Q) \vee (P \wedge R)]$	Distributividade
12	$[P \vee (Q \wedge R)] \Leftrightarrow [(P \vee Q) \wedge (P \vee R)]$	Distributividade
13	$(P \vee \mathbb{V}) \Leftrightarrow \mathbb{V}$	
14	$(P \wedge \mathbb{F}) \Leftrightarrow \mathbb{F}$	
15	$(P \vee \mathbb{F}) \Leftrightarrow P$	
16	$(P \wedge \mathbb{V}) \Leftrightarrow P$	
17	$(P \vee \neg P) \Leftrightarrow \mathbb{V}$	
18	$(P \wedge \neg P) \Leftrightarrow \mathbb{F}$	
19	$\neg(\neg P) \Leftrightarrow P$	Dupla Negação
20	$(P \rightarrow Q) \Leftrightarrow (\neg P \vee Q)$	Implicação
21	$(P \leftrightarrow Q) \Leftrightarrow [(P \rightarrow Q) \wedge (Q \rightarrow P)]$	Equivalência
22	$[(P \wedge Q) \rightarrow R] \Leftrightarrow [P \rightarrow (Q \rightarrow R)]$	Exportação
23	$[(P \rightarrow Q) \wedge (P \rightarrow \neg Q) \Leftrightarrow \neg P]$	Absurdo
24	$(P \rightarrow Q) \Leftrightarrow (\neg P \rightarrow \neg Q)$	Contrapositivo

Tabela 2.4: Tautologias ou identidades

Exemplos

3. $(P \vee P) \Leftrightarrow P$ IDEMPOTÊNCIA DE $\vee$
É uma identidade óbvia quase nunca usada na linguagem natural. Dizer *"Pedro é feliz ou Pedro é feliz"* equivale simplesmente a *"Pedro é feliz"*. É usada para substituições nas expressões formais.

4. $(P \wedge P) \Leftrightarrow P$ IDEMPOTÊNCIA DE $\wedge$
O mesmo que a anterior.

5. $(P \vee Q) \Leftrightarrow (Q \vee P)$ COMUTATIVIDADE DE $\vee$
Dizer que *"o cachorro é branco ou o cachorro é grande"* é o mesmo que dizer *"o cachorro é grande ou o cachorro é branco"*.

6. $(P \wedge Q) \Leftrightarrow (Q \wedge P)$ COMUTATIVIDADE DE $\wedge$
O mesmo que a 5 substituindo *ou* por *e*.

7. $[(P \vee Q) \vee R] \Leftrightarrow [P \vee (Q \vee R)]$ ASSOCIATIVIDADE DE $\vee$
Esta identidade corresponde a dizer que *"Chico é magro ou calmo, ou também é grisalho"* é o mesmo que *"Chico é magro, ou também é calmo ou grisalho"*

8. $[(P \wedge Q) \wedge R] \Leftrightarrow [P \wedge (Q \wedge R)]$ ASSOCIATIVIDADE DE $\wedge$
O mesmo que a 7 trocando *ou* por *e*.

9. $\neg(P \vee Q) \Leftrightarrow (\neg P \wedge \neg Q)$ LEI DE DE MORGAN
A sentença *"Não é o caso de Chico ser magro ou calmo"* só é verdade se *"Chico não é magro e Chico não é calmo"*.

10. $\neg(P \wedge Q) \Leftrightarrow (\neg P \vee \neg Q)$ LEI DE DE MORGAN
Analogamente ao caso anterior, dizer que *"Não é o caso de Chico ser magro e calmo"* é o mesmo que dizer *"Chico não é magro ou Chico não é calmo."*

11. $[P \wedge (Q \vee R)] \Leftrightarrow [(P \wedge Q) \vee (P \wedge R)]$ DISTRIBUTIVIDADE
Esta propriedade diz que a proposição *"Chico é magro, e ele é calmo ou grisalho"* é o mesmo que *"Chico é magro e calmo, ou Chico é magro e grisalho"*

12. $[P \vee (Q \wedge R)] \Leftrightarrow [(P \vee Q) \wedge (P \vee R)]$ DISTRIBUTIVIDADE
Similar ao exemplo 9, dizer que *"Laura é bonita, ou é simpática e feliz."* é equivalente a dizer *"Laura é bonita ou simpática, e Laura é bonita ou feliz."*

13. $(P \vee \mathbb{V}) \Leftrightarrow \mathbb{V}$

14. $(P \wedge \mathbb{F}) \Leftrightarrow \mathbb{F}$
15. $(P \vee \mathbb{F}) \Leftrightarrow P$
16. $(P \wedge \mathbb{V}) \Leftrightarrow P$

A proposição 13 é sempre verdade, independente do valor de P. Estas identidades são raramente ou nunca usadas na linguagem diária, entretanto são úteis para manipular expressões no raciocínio na forma simbólica.

17. $(P \vee \neg P) \Leftrightarrow \mathbb{V}$

Uma proposição do tipo *"Laura é bonita ou Laura não é bonita"* é sempre verdade, porque contempla todas as possibilidades.

18. $(P \wedge \neg P) \Leftrightarrow \mathbb{F}$

Pelo motivo oposto ao da identidade 17, dizer que *"Laura é bonita e Laura não é bonita"* sempre será falso porque as duas proposições elementares são contraditórias.

19. $\neg(\neg P) \Leftrightarrow P$ DUPLA NEGAÇÃO

Negar duplamente uma proposição é o mesmo que afirmá-la. Por exemplo, *"Ana não vai não cantar"* significa *"Ana vai cantar"*.

20. $(P \rightarrow Q) \Leftrightarrow (\neg P \vee Q)$ IMPLICAÇÃO

Neste caso, a proposição *"se eu ganhar a loteria então te dou um milhão de Reais"* só é falsa se eu ganhar e não te der o milhão; é verdadeira em todas as outras possibilidades. Logo, equivale à sentença *"eu não ganho a loteria ou te dou um milhão de Reais."*

21.$(P \leftrightarrow Q) \Leftrightarrow [(P \rightarrow Q) \wedge (Q \rightarrow P)]$ EQUIVALÊNCIA

No caso de equivalência entre duas sentenças, a primeira implica na segunda e a segunda implica na primeira. *"Ana é feliz se e somente se é saudável"* significa que *"se Ana é feliz então é saudável e se Ana é saudável então é feliz."*

22. $[(P \wedge Q) \rightarrow R] \Leftrightarrow [P \rightarrow (Q \rightarrow R)]$ EXPORTAÇÃO

Por exemplo, a sentença *"se o sapo é grande e verde, então ele é sapo-boi"* equivale a dizer *"se o sapo é grande, então se ele é verde, então ele é sapo-boi"*

23. $[(P \rightarrow Q) \wedge (P \rightarrow \neg Q) \Leftrightarrow \neg P]$ ABSURDO

Se uma proposição implica em dois resultados contraditórios, então ela não é verdadeira. Se são verdadeiras as sentenças *"Se Ana é feliz, então é simpática"* e *"Se Ana é feliz, então não é simpática"*, então a premissa de que

"Ana é feliz" deve estar equivocada.

24.$(P \to Q) \Leftrightarrow (\neg P \to \neg Q)$ CONTRAPOSITIVO
A proposição *"se a criança brinca, então é esperta"* contém o mesmo conteúdo lógico que *"se a criança não brinca, então não é esperta"*

Proposição Dual

Considere uma proposição X que contém somente os conectivos $\wedge$, $\vee$ e $\neg$. A sua **proposição dual**, denotada X^* é construída substituindo-se $\wedge$ por $\vee$, $\vee$ por $\wedge$, $\mathbb{V}$ por $\mathbb{F}$ e $\mathbb{F}$ por $\mathbb{V}$. Assim, a proposição dual de uma proposição é construída invertendo-se os seus conectivos lógicos. Ela contém essencialmente o mesmo conteúdo que a proposição da qual foi originada.

Propriedade: se duas proposições P e Q, envolvendo somente $\wedge$, $\vee$ e $\neg$ são equivalentes, então P^* e Q^* também são equivalentes.

2.1.6 Raciocínio

Raciocínio lógico é o processo de tirar conclusões a partir de premissas usando regras de inferência. A regra de inferência mais utilizada é *Modus Ponens*[1] que diz que se $(P \to Q)$ e P são verdadeiros, então Q é verdadeiro, escrito como (veja a seção 2.1.3):

$$\begin{array}{l} P \to Q \\ P \\ \hline Q \end{array}$$

As linhas acima da barra são **premissas** e a linha abaixo é a **conclusão** deduzida das premissas.

Se as proposições *"se chove então não há festa"* e *"chove"* são ambas verdadeiras, então pode-se concluir que não há festa.

Uma proposição pode ser escrita num forma alternativa usando-se as identidades lógicas (Seção 2.1.5). Uma expressão da proposição pode ser trocada por uma expressão equivalente, sem alterar o conteúdo lógico da proposição inteira. Por exemplo, na expressão $[P \wedge (P \to Q)]$ sabemos que, veja Tabela 2.4, $(P \to Q) \Leftrightarrow (\neg P \wedge Q)$, assim podemos escrever a primeira expressão como $[P \wedge (\neg P \wedge Q)]$. Deste modo, de uma proposição é uma identidade obteve-se outra proposição, de mesmo conteúdo lógico. Este processo equivale ao **raciocínio**.

[1] Significa modo de construção.

Regra de Inferência	Forma Tautológica	Nome
P $P \to Q$ ——— Q	$[P \wedge (P \to Q)] \Rightarrow Q$	*Modus Ponens*
$\neg Q$ $P \to Q$ ——— $\neg P$	$[\neg Q \wedge (P \to Q)] \Rightarrow \neg P$	*Modus Tollens*
$P \vee Q$ $\neg P$ ——— Q	$[(P \vee Q) \wedge \neg P] \Rightarrow Q$	*Silogismo Disjuntivo*
$(P \to Q)$ $(R \to S)$ $(P \vee R)$ ——— $(Q \vee S)$	$[(P \to Q) \wedge (R \to S) \wedge (P \vee R)] \Rightarrow (Q \vee S)$	*Dilema Construtivo*
$(P \to Q)$ $(R \to S)$ $(\neg Q \vee \neg S)$ ——— $(\neg P \vee \neg R)$	$[(P \to Q) \wedge (R \to S) \wedge (\neg Q \vee \neg S)] \Rightarrow (\neg P \vee \neg R)$	*Dilema Destrutivo*

Tabela 2.5: Principais Regras de Inferência

Exemplos

Modus Ponens. Considere as proposições a seguir:

1. O casaco é preto P;
2. O casaco é branco B;
3. O casaco é meu: M;
4. O meu casaco não é branco: N.

Podemos representar a última proposição como $N \leftrightarrow (M \rightarrow \neg B)$, assim escrevemos

$$\frac{\begin{array}{l}(P \vee B)\\ M\\ N\end{array}}{\begin{array}{l}(P \vee B)\\ M\\ \underline{(M \rightarrow \neg B)} \qquad \text{substituindo } [N \leftrightarrow (M \rightarrow \neg B)]\\ P \vee B\\ \underline{\neg B} \qquad\qquad\qquad \text{substituindo } [M \rightarrow \neg B]\end{array}}{P}$$

ou seja, P, o casaco é preto.

2.2 Lógica de Predicados

Em alguns casos, a lógica proposicional não é suficiente ou prática para representar todos as sentenças que ocorrem. Por exemplo, haveria dificuldade para representar com lógica proposicional uma sentença do tipo *"todos os pássaros voam"* ou *"alguns números são primos"*.

2.2.1 Predicados

Na gramática, para que uma oração tenha sentido, basicamente deve conter sujeito e predicado. Sujeito é o termo sobre o qual a oração diz algo e predicado é o termo que contém o verbo e diz algo sobre o sujeito. Na oração *"o vento engrossava as ondas"*, *o vento* é sujeito e *engrossava as ondas* é predicado.

Um **predicado** é um formato de uma frase verbal que descreve uma propriedade de objetos ou uma relação entre objetos. Por exemplo as sentenças *"a árvore é verde"*, *"a blusa de Maria é verde"* e *"o cabelo é verde"* todas são construídas com o predicado *é verde* aplicado a algum sujeito. A frase *é verde* é um predicado e descreve a propriedade de ser verde.

Os predicados podem ser representados dando-lhes um nome, o predicado *é verde* poderia ser representado por `é_verde`, `verde` ou `V`. Assim a atribuição de ser verde poderia ser representado como `é_verde`(x) ou `verde`(x), onde x é um objeto arbitrário qualquer. As sentenças anteriores seriam representadas por `verde(árvore)`, `verde(blusa de Maria)` e `verde(cabelo)`.

Também podemos representar relações entre objetos através dos predicados. Por exemplo, *"João ama Maria"* e *"Ana ama José"* poderiam ser representados respectivamente por `ama(João,Maria)` e `ama(Ana,José)`. Sentenças mais complexas, com mais de dois sujeitos, seguem o mesmo padrão. *"Luis deu um lápis para Márcia"* pode ser escrito como `deu(Luís,lápis,Márcia)`, de modo que `deu(Julia,1/2 pão,Ana)` significaria *"Júlia deu meio pão para Ana"*.

2.2.2 Quantificação

Um predicado com variáveis não é uma proposição. Por exemplo, uma sentença do tipo $x > 1$ com a variável x não especificada não é nem verdadeira nem falsa, pois não se sabe o valor de x, podendo ser verdadeira ou falsa dependendo de x. Para transformar a sentença numa proposição ou substitui-se x por um valor específico, perdendo a generalidade, ou então escreve-se *"existe um x para o qual $x > 1$"*. Assim, a frase *"existe um x para o qual $x > 1$"* é verdadeira; de fato, existem valores de x que são maiores que 1. De maneira geral, um predicado com variáveis, chamada de **fórmula atômica**, pode ser transformada numa proposição realizando uma das operações:

1. Atribuir uma valor à variável.
   Ex.: $(x > 1)$ se torna $(3 > 1)$ [verdadeiro] ou $(0 > 1)$ [falso].

2. Quantificar a variável usando um quantificador.
   Ex.: *"existe um número x tal que $x > 1$"* [verdadeiro] ou *"para todo x tem-se que $x > 1$"* [falso].

Em geral a quantificação é executada em fórmulas de lógica de predicados, usando os quantificadores. Existem dois tipos de quantificadores: o quantificador universal e o quantificador existencial.

Universo do discurso

O universo do discurso, chamado simplesmente de **universo**, é o conjunto de objetos de interesse. O universo é o domínio das variáveis em questão. As proposições na lógica de predicados são sentenças sobre os objetos do universo do discurso. Exemplos são o conjunto dos números inteiros, todos as formigas num formigueiro, o conjunto de máquinas numa fábrica, os estudantes de uma sala de aula, etc. Quando o universo é deixado implícito na prática, deve ser óbvio a partir do contexto.

Quantificador Universal – $\forall$

O **quantificador universal**, simbolizado por $\forall$, generaliza uma fórmula atômica $P(x)$ para todos os objetos do universo. O quantificador universal transforma uma sentença do tipo $(x > 1)$ em *"para cada objeto x no universo,* $x > 1$*"*, que é expresso como $\forall x\ x > 1$. Esta nova proposição é verdadeira ou falsa dependendo do universo de discurso. Assim, é uma proposição uma vez que o universo é especificado.

A expressão $\forall x\ P(x)$ pode ser traduzida em linguagem natural como *"para todo x,* $P(x)$ *vale"* ou *"para cada x,* $P(x)$ *vale."* Por exemplo a expressão *"todos os cachorros tem dentes"* poderia ser transformada na forma proposicional $\forall x\ D(x)$, onde $D(x)$ é o predicado que significa *"ter dentes"* e x é um universo composto somente por cachorros.

Se todos os elementos do universo de discurso podem ser listados, então o quantificador universal pode ser substituído pelo conectivo "e". A expressão $\forall x\ P(x)$ pode ser substituída por $P(x_1) \wedge P(x_2) \wedge \ldots \wedge P(x_n)$, onde o universo é composto pelos objetos $\{x_1, x_2, \ldots, x_n\}$.

Quantificador Existencial – $\exists$

De maneira análoga, o **quantificador existencial**, simbolizado por $\exists$, garante a validade de uma forma atômica $P(x)$ para pelo menos um elemento do universo do discurso. O quantificador existencial transforma uma sentença do tipo $(x > 1)$ em *"existe um x tal que* $P(x)$*"* ou *"existe pelo menos um x tal que* $P(x)$*"*.

Uma sentença do tipo *"algumas plantas tem flores"* pode ser escrita como $\exists x\ F(x)$, onde $F(x)$ é o predicado que significa *"ter flores"* e x é o universo de todas as plantas de uma região, por exemplo.

Se todos os elementos do universo de discurso podem ser listados, então o quantificador existencial pode ser substituído pelo conectivo "ou". A expressão $\exists x\ P(x)$ pode ser substituída por $P(x_1) \vee P(x_2) \vee \ldots \vee P(x_n)$, onde o universo é composto pelos objetos $\{x_1, x_2, \ldots, x_n\}$.

Escopo de variáveis

Uma variável é dita **variável limitada** se um valor é atribuído a ela ou se é quantificada. Se ela não é limitada então é dita **livre**. A extensão do efeito do quantificador é chamado de **escopo** do quantificador e indicado por colchetes [ ]. Se não há colchetes, então o escopo do quantificador é o menor átomo que segue à quantificação.

Por exemplo, na expressão $\exists x\ P(x, y)$ a variável x é limitada e y é livre. Na expressão $\forall x[\exists y\ P(x, y) \wedge Q(x, y)]$, x é limitado tanto em $P(x, y)$ como em $Q(x, y)$, mas y é limitado somente em $P(x, y)$ e em $Q(x, y)$ é livre; $\forall x$ se aplica para o x de $P(x, y)$ e de $Q(x, y)$ por causa dos colchetes, mas $\exists y$ se aplica somente a $P(x, y)$.

Interpretando fórmulas quantificadas

Ao ler fórmulas quantificadas, lê-se da esquerda para direita. $\forall x$ pode ser lido como *"para todos os objetos x no universo o que segue é válido"* e $\exists x$ como *"existe um objeto x no universo que satisfaz"*.

Por exemplo, suponha que o universo é composto por todas as motocicletas e que o predicado $R(x, y)$ significa *"x mais rápido que y"*. Deste modo, $\forall x\ \forall y\ R(x, y)$ pode ser escrito como *"para cada moto x o seguinte vale: x é mais rápido que cada y"* ou, de outro modo, *"que cada moto é mais rápida que cada moto"* (incluindo a si mesma.)

$\forall x \exists y\, R(x, y)$ significa que *"para cada moto x o seguinte vale: para alguma moto y, x é mais rápido que y."*

$\exists x \forall y\, R(x, y)$ significa que *"existe uma moto x para a qual o seguinte vale: para cada moto y, x é mais rápido que y."*

$\exists x \exists y\, R(x, y)$ significa que *"existe uma moto x para a qual o seguinte vale: para alguma moto y, x é mais rápido que y."*

Ordem de aplicação dos quantificadores

Os quantificadores são aplicados na fórmula a partir do que está mais próximo da fórmula. Assim, na proposição $\exists y\ \forall x\ P(x, y)$ deve ser interpretado como $\exists y\ [\forall x\ P(x, y)]$, ou seja, *"existe um y tal que, para todo x, $P(x, y)$ vale"* ou *"para algum y, $P(x, y)$ vale para todo x."*

As posições relativas de quantificadores de mesmo tipo podem ser trocadas sem afetar o significado da proposição, desde que não haja quantificadores de outro tipo no meio dos que serão intercambiados. Assim, $\exists x \exists y \exists z P(x, y, z)$ pode ser escrito como $\exists y\ \exists z\ \exists x\ P(x, y, z)$ ou $\exists z\ \exists x\ \exists y\ P(x, y, z)$. O mesmo vale para o quantificador universal.

Entretanto, a posição de diferentes tipos de quantificadores **não** pode ser trocada. Por exemplo, $\forall x \, \exists y \, P(x, y)$ **não** é equivalente a $\exists y \forall x \, P(x, y)$. Para esclarecer, suponha que $M(x, y)$ represente $(x < y)$ para os números de um dado universo. Assim, $\forall x \, \exists y \, M(x, y)$ significa que *"para cada x, existe um y que é maior que x"*, o que é verdade. Entretanto $\exists y \, \forall x \, M(x, y)$ significa que *"existe um y que é maior que cada número x"*.

2.2.3 Fórmulas bem formadas (FBF)

Sem toda sentença representa uma proposição na lógica de predicados. Aquelas que representam uma proposição são chamadas de **fórmulas bem formadas** (do inglês *well-formed formulas – wff*) da lógica de predicados de primeira ordem.

Regras de sintaxe para construção de FBF

A expressão de um predicado e suas variáveis como em $P(x, y)$ é chamado de **fórmula atômica**. Para construir uma FBF, deve-se obedecer as seguintes regras:

1. $\mathbb{V}$ e $\mathbb{F}$ são FBF;
2. Cada constante e cada variável proposicional é uma FBF;
3. Cada fórmula atômica (predicado e variáveis) é uma FBF;
4. se A e B são FBF, então também são $\neg A$, $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$ e $(A \leftrightarrow B)$;
5. Se x é uma variável e A é uma FBF, então também são $\forall x \, A$ e $\exists x \, A$.

Por exemplo, *"Crusty é um cachorro"* é uma proposição específica, logo é uma FBF pela regra 2. Seja $V(x)$ o predicado *"ser vermelho"* e x uma variável; a fórmula atômica $V(x)$ é FBF pela regra 3. Da mesma forma, $\forall x \, V(x)$ e $\exists x \, V(x)$ também é FBF pela regra 5. Se o predicado $R(x)$ representa *"ser redondo"*, então, pela regra 4, $V(x) \wedge R(x)$ e $V(x) \vee R(x)$ são FBF. Por outro lado, $B(\exists x)$, $V(R(x))$ ou $\forall x \, V(x) R(x)$ **não** são FBF. De maneira geral, se a fórmula pode ser interpretada numa linguagem natural, então provavelmente é uma FBF.

Interpretação de FBF

Um FBF nem sempre é uma proposição. Por exemplo, a FBF $\forall x\ P(x)$ significa *"número não negativo"*; esta FBF é verdadeira se o universo é o conjunto $\{1, 3, 5\}$, o conjunto $\{2, 3, 4\}$ ou então todos os números naturais $\{0, 1, 2, \ldots\}$; entretanto é falsa se o universo é o conjunto $\{-1, 2, 7\}$, por exemplo. Além do mais, a FBF $\forall x\ S(x, y)$, onde $S(x, y)$ significa *"menor que"* pode ser verdadeira ou falsa para o universo $x = \{1, 2, 3\}$ dependendo do valor de y.

Logo, o valor verdade de uma FBF depende tanto da fórmula atômica (variáveis e predicados) como também do universo de discurso. A especificação do universo e predicados, e a atribuição de uma valor para cada uma das variáveis livres em uma FBF é chamado de uma **interpretação** de uma FBF.

Deste modo, se o predicado $D(x, y)$ significa *"densidade de x maior que de y"*, então atribuindo o universo como $\{x = \text{água, mercúrio, aço}\}$ e atribuindo o valor *"óleo"* a y, então a fórmula $\forall x\ D(x, y)$ é verdadeira, que é lida como *"cada elemento do conjunto {água, mercúrio, aço} é mais denso que óleo"*. Uma FBF se torna uma proposição quando recebe uma interpretação.

Satisfazibilidade

Uma FBF é dita **satisfazível** se existe um interpretação (um modelo) que a torna verdadeira, isto é, se há um universo, predicados e valores das variáveis livres que tornam a FBF verdadeira.

Em contrapartida, uma FBF é dita **insatisfazível** ou **incoerente** se não existe uma interpretação que a torna verdadeira. Por exemplo, $\forall x\ [N(x) \wedge \neg N(x)]$ nunca pode ser verdadeira independentemente do universo, predicados e valores das variáveis livres.

Uma FBF é dita **válida** se é verdadeira para qualquer interpretação. Por exemplo, a FBF $\forall x\ P(x) \vee \exists x\ \neg P(x)$ é válida para qualquer predicado $P(x)$ pois $\exists x\ \neg P(x)$ é a negação de $\forall x\ P(x)$

Equivalência de quantificadores

Duas FBF F_1 e F_2 são **equivalentes** se e somente se $F_1 \leftrightarrow F_2$ é válido, isto é se e somente se $F_1 \leftrightarrow F_2$ é verdadeiro para todas as interpretações.

Por exemplo, $\forall x\ P(x)$ e $\neg\exists x\ \neg P(x)$ são equivalentes para qualquer predicado P. Assim também o são $\forall x\ [P(x) \wedge Q(x)]$ e $[\forall x\ P(x) \wedge \forall x\ Q(x)]$ para qualquer predicado P e Q.

$$\forall x\ P(x) \Leftrightarrow \neg\exists x\ \neg P(x)$$

$$\exists x P(x) \Leftrightarrow \neg \forall x \, \neg P(x)$$

Capítulo 3

Representação do conhecimento

A representação do conhecimento consiste em colocar o conteúdo do aprendizado numa forma que possa ser armazenada e tratada de maneira sistemática. Representar o conhecimento é necessário não somente para organizar o conhecimento adquirido, mas também para que se possa realizar operações com esta informação armazenada. Uma boa representação do conhecimento permite que a informação seja manipulada de maneira eficiente, ampliando a aplicação do conhecimento armazenado. As vantagens mais importantes que se obtém ao representar o conhecimento são as seguintes:

- **Função dedutiva:** obtenção de novos resultados a partir dos dados armazenados.
- **Função consulta:** manipulação eficiente dos dados.
- **Função organização:** a informação organizada é mais facilmente compreensível não só por máquinas como por humanos.
- **Função interpretação:** comparação dos dados com modelos, resultando em interpretações.

Uma representação de conhecimento deve ser projetada em função dos objetivos a atingir. Os fatos importantes ou cruciais para o sistema em questão devem estar óbvias na representação; ao mesmo tempo que a representação deve ser concisa para poupar espaço e facilitar o processamento, deve ser o mais completa possível, contendo todos os fatos pertinentes para o sistema. De um modo geral, podemos classificar as características de uma representação de conhecimento através das seguintes propriedades:

- **Granularidade:** é o nível de detalhamento da descrição;

- **Disponibilidade:** natureza implícita ou explícita do conhecimento armazenado; compromisso entre eficiência e armazenamento;
- **Credibilidae:** o grau de confiança nas informações;
- **Modularidade:** facilidade de modificar a estrutura que representa o conhecimento;
- **Transparência**[1]**:** clareza na representação do conhecimento;
- **Eficiência:** facilidade de obter uma informação representada;
- **Uniformidade:** regras de representação escritas com o mesmo padrão;
- **Naturalidade:** semelhança da representação com o formato natural humano;

3.1 Sistemas de produção

Os **sistemas de produção** são compostos por um conjunto de regras que reúnem condições e ações. As **regras de produção** são estes conjuntos de condição e ação . A condição é formada por um padrão que determina se se deve aplicar a regra correspondente. Estes conjunto de regras pode ser representado da seguinte maneira:

$$\begin{array}{ccc} \text{CONDIÇÃO1} & \rightarrow & \text{AÇÃO1} \\ \text{CONDIÇÃO2} & \rightarrow & \text{AÇÃO2} \\ \vdots & \vdots & \vdots \\ \text{CONDIÇÃON} & \rightarrow & \text{AÇÃON} \end{array}$$

As condições são chamadas de *lado esquerdo*, premissas ou antecedentes; as ações são o *lado direito* ou conseqüente. Um exemplo muito simples de um sistema de produção seria:

[1] O oposto da transparência é a opacidade

```
se [ dia() ∧ chove() ∧ venta() ] então
   fechar(janelas)
   abrir(cortinas)
se [ dia() ∧ sol() ∧ ¬venta() ] então
   fechar(cortinas)
   abrir(janelas)
se [ ¬dia() ] então
   fechar(cortinas)
   fechar(janelas)
se [ dia() ∧ noite() ] então
   fim_do_mundo()
   ⋮
```

A regras de produção tem a vantagem de ser prático editá-las (modularidade), interpretá-las (naturalidade) e que são todas escritas da mesma forma (uniformidade). Por outro lado, tem a dificiência de que é difícil verificar a completeza do sistema (opacidade) e de serem necessárias muitas buscas e comparações (ineficiência).

Em geral sua aplicação é mais adequada para sistemas onde o conhecimento é difuso, com muitos fatos, ou em sistemas em que os fatos podem ser representados de maneira independente. Exemplos de aplicação são sistemas de diagnóstico em geral, monitoração de pacientes, algoritmos de classificação.

3.2 Redes semânticas

Numa **rede semântica** o conhecimento é representado por meio de uma rede de **vértices** e **arestas**. Os vértices representam os objetos, situações ou conceitos, enquanto as arestas representam as relações entre eles. Por exemplo considere a sentença: *Ane tem um bonito livro "Veleiros", onde há um texto sobre barcos. O livro faz parte da sua importante biblioteca, que fica no edifício Bento Gonçalves.*

Existem vários **tipos de ligações** entre os vértices. Os mais comuns são *é_um*, *é*, *parte_de*. As relações podem ser classificados nos tipos:

- **Propriedade:** relação com um vértice que é uma característica do objeto.
- **Subparte:** quando um vértice é componente de outro; *parte_de*.

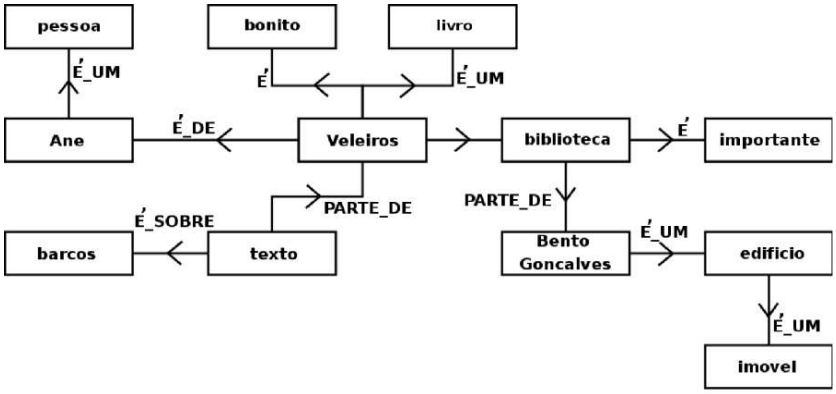


Figura 3.1: Exemplo de rede semântica simples.

- **Subclasse:** quando os vértices podem ser agrupados e relacionados em uma classe com outros vértices.
- **Relacionamento:** quando os vértices se relacionam de alguma forma.

Os conceitos representados na rede semântica podem ser representados também por meio da lógica de predicados (Seção 2.2). Por exemplo, a rede da Figura 3.1 poderia ser representada como:

```
é_de("Veleiros", Ane)
é_um(Ane, pessoa)
parte_de(texto, "Veleiros")
é_um("Veleiros", livro)
é_um(biblioteca, edifício)
parte_de("Veleiros", biblioteca)
          ⋮
```

Numa rede semântica, os vértices herdam as propriedades dos elementos de mais alto nível, economizando espaço de armazenamento e processamento. Por exemplo, "Veleiros" herda as características do conceito *livro*. No exemplo abaixo, *pardal* e *pomba* são *AVES* e, portanto, herdam as características de *voar* e *por ovos*.

3.3 Quadros e roteiros

Um **quadro** representa um conjunto de informações sobre uma certa situação ou objeto. Os quadros descrevem as classes de objetos. Considere o quadro geral `livro`:

QUADRO: `livro`

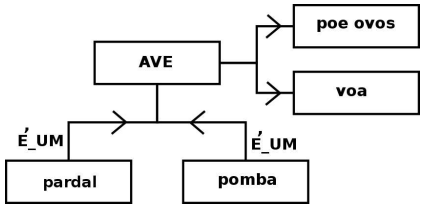


Figura 3.2: Herança de propriedades

ATRIBUTO: `título` caracter
ATRIBUTO: `autor` caracter
ATRIBUTO: `páginas` inteiro
ATRIBUTO: `formato` **retangular**, redondo, irregular
ATRIBUTO: `material` plástico, **papel**, tecido

Num quadro, cada atributo armazena um tipo de informação. Alguns atributos tem seu conteúdo pré-definido e possuem um valor padrão, caso uma opção não seja feita (valores em negrito em `livro`). Podemos usar o quadro geral `livro` para especificar um livro em específico, o `livro` "Veleiros". O `livro` "Veleiros" tem os mesmo atributos que `livro`, constituindo uma instância de `livro`.

QUADRO: `livro` "Veleiros"
ATRIBUTO: `título` "Veleiros"
ATRIBUTO: `autor` "Pedro Cabral"
ATRIBUTO: `páginas` 925

Os atributos que não são explicitamente especificados são herdados e assumem o valor padrão, por exemplo os atributos `formato` (retangular para "Veleiros") e `material` (papel.) Quando o sistema tem de criar uma instância de um quadro, o fato de ter de procurar pelas informações dos atributos não especificados é uma característica do processo de busca e raciocínio.

Os **roteiros** são semelhantes aos quadros, mas descrevem eventos ao invés de objetos. As entradas do roteiro são as condições que devem ser satisfeitas para o roteiro ocorrer e os resultados são as condições que serão satisfeitas após o roteiro ocorrer. Além disso podem haver apoios, que são objetos envolvidos nos eventos descritos no roteiro.

Por exemplo, para o roteiro `tomar_banho`, teríamos as seguinte características a seguir. Entradas: estar sujo, haver água, haver sabonete, haver toalha, haver roupa limpa, não estar vestido; Resultados: estar limpo, estar seco, estar vestido; Apoios: sabonete, banheiro, água, toalha, roupas.

Capítulo 4

Espaço e solução de problemas

O processo de solução de um problema em especial com IA é feito basicamente através de três etapas que serão discutidas nesta seção. São elas:

1. Definir o problema precisamente. Definir que variáveis são importantes para descrever o sistema e determinar estados iniciais e finais.
2. Analisar o problema. Considerar as regras e as características relevantes para o sistema.
3. Escolher a técnica que melhor se adapta ao sistema e aplicá-la.

4.1 Espaço de estados

Cada configuração do sistema é chamado de **estado do sistema**. O conjunto de todas as configurações possíveis para o sistema é chamado de **espaço de estados** do sistema. Em geral, determinar um solução para um problema é descobrir um caminho no espaço de estados que leva da situação inicial para a situação final, passando por vários estágios intermediários. A solução se dá através dos seguintes passos:

1. Definir um ou mais estados relevantes $\mathcal{E} = \{e_0, e_1, \ldots, e_N\}$
2. Definir um ou mais **estados iniciais** e_i a partir de $\mathcal{E}$.
3. Definir um ou mais **estados finais** e_f a partir de $\mathcal{E}$.
4. Descrever um conjunto de **regras possíveis** $\mathcal{R} = \{r_0, r_1, \ldots, r_M\}$ para o problema na forma de **condições** $\mathcal{C} = \{c_0, c_1, \ldots, c_M\}$, que quanto satisfeitas serão seguidas pela aplicação dos **operadores** $\mathcal{O} = \{o_0, o_1, \ldots, o_M\}$.

5. Aplicar as regras ao sistema para tentar obter e_f de e_i ou vice-versa.

4.2 Características dos problemas

Ao determinar que métodos aplicar a uma determinada situação, é importante avaliar características gerais que são comuns a todos os problemas.

4.2.1 Decomponibilidade

É a características de o problema poder ou não ser decompostos em problemas mais simples. Por exemplo, um problema do tipo $[3\times5+2^2+\log(40)]$ pode ser decomposto em 3×5, 2^2 e $\log(40)$. Por outro lado, o problema de empilhar várias caixas não pode ser decomposto em empilhar duas de cada vez porque as primeiras já devem ter sido empilhadas para que se possam colocar as últimas. De maneira geral, quando as etapas do problema são idependentes, o problema pode ser decomposto.

4.2.2 Histórico das etapas

Ao resolver um problema é importante saber se a sequência das operações realizadas no sistema podem, a história de evolução do sistema, pode ser ignorada ou desfeita, em parte ou completamente.

Dependendo da características do sistema, pode-se guardar o histórico das etapas do sistema, para voltar as etapas realizadas ou para determinar características da trajetória. Se as etapas não podem ser desfeitas, o planejamento dos passos deve ser muito mais elaborado.

Quanto ao tipo de história de etapas do sistema, podemos classificá-las como:

- **Ignorável:** os passos podem ser ignorados. É solucionável por uma estrutura de controle que nunca retrocede. Por exemplo, na prova de teoremas, não interessa a ordem das regras que foram utilizadas, desde que se chegue numa solução.

- **Recuperável:** os passos realizados podem ser desfeitos. É solucionável por uma estrutura de controle que eventualmente comete erros e precisa retroceder. Por exemplo, o quebra-cabeças de 8 ou a saída de um labirinto.

- **Irrecuperável:** os passos da solução não podem ser desfeitos. A solução utiliza uma estrutura de controle que não deve errar, pois cada

passo é final. Necessita planejamento (simulação antes de agir.) Por exemplo, o xadrez, em que jogadas erradas não poderão ser desfeitas.

Em geral sistemas que tem interação com o meio ambiente, em que este tenha participação ativa no sistema, são irrecuperáveis.

4.2.3 Previsibilidade

Num sistema qualquer, é importante saber em que nível o conhecimento dos estados podem ser determinados. Podemos classificar os sistemas quanto a previsibilidade em três grupos:

- **Estados determinados exatamente** – os resultados de cada operação são exatamente conhecidos e a sequência de estados pode ser conhecida exatamente. São deste tipo os problemas sem interação com o meio-ambiente, como o quebra-cabeças 8.
- **Estados probabilísticos** – as probabilidades das várias sequências de estados podem ser estimadas; não há determinismo sobre o resultado de uma operação. São sistemas em que há interação com o meio ambiente. Planejamento possível com análise de vários conjuntos de soluções, necessitando contínuo planejamento de acordo com a resposta do meio-ambiente. São deste tipo os jogos de carta, movimento de um robô num ambiente fechado.
- **Estados indeterminados** – os estados do sistema não podem ser listados e nem as suas probabilidades relativas. Por exemplo, o movimento de um robô num ambiente aberto e desconhecido em larga escala; o auxílio a um advogado a defender um cliente.

De forma genérica, juntando as características da história das etapas e da previsibilidade, os problemas mais fácies são os recuperáveis e de estados determinados. Nestes casos, é possível saber exatamente que resultado advém da aplicação de um operador e todas as etapas podem ser revertidas se necessário. Em contrapartida, o tipo de sistema mais difícil é o irrecuperável e de estados indeterminados, em que não se pode nem retroceder ao cometer um erro e também não se sabe o resultado de uma ação.

4.2.4 Satisfazibilidade

Satisfazibilidade é a característica que determina em que nível a solução do sistema satisfaz os requisitos propostos. Pode ser dividade em:

- **Local:** qualquer caminho que leve ao resultado é aceitável, p.ex. prova de teoremas. Busca heurística é aplicável.
- **Global:** é necessário determinar o melhor caminho de todos. Necessita uma busca heurística muito mais abranjente. Por exemplo, movimento do robô e viajante.

4.3 Grafos e árvores

O espaço de estados do problema pode ser representado em um gráfico orientado, chamado **grafo**, em que os estados do sistema $\{e_0, e_1, \ldots, e_N\}$ são representados por vértices e os operadores $\{o_0, o_1, \ldots, o_M\}$ são representados pelas arestas, como mostra a figura abaixo [Bt2001].

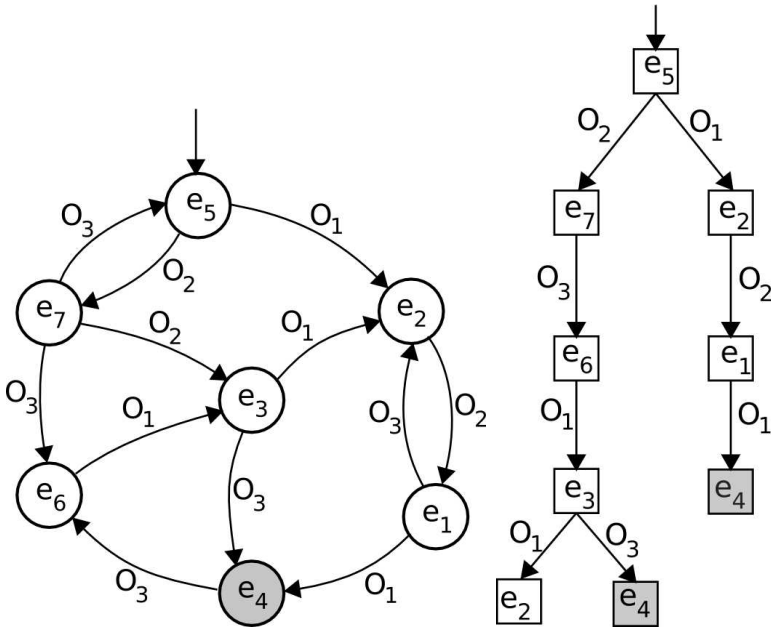


Figura 4.1: Um grafo orientado (direita) e um árvore (esquerda) representando os estados $\{e_1, e_2, \ldots, e_N\}$ e os operadores $\{o_1, o_2, \ldots, o_M\}$

Quando o grafo é num só sentido é também chamado de árvore.

4.4 Métodos de Busca

Uma solução de um problema consiste em determinar uma sequêcia de operações que são realizadas no/s estado/s inicial/s a fim de obter o/s estado/s

final/s. Um método de busca é um algoritmo que determina esta sequência de acordo com determinado critério.

Existem vários métodos de busca cujo desempenho depende da situação. Além das características mencinadas acima é importante considerar no processo de busca:

- A direção da busca (direta ou retrógrada);
- A topologia do processo de busca, i.e., o fator de ramificação;
- A representação dos vértices;
- A aplicação das regras;
- A guiagem da busca (função heurística).

4.4.1 Buscas Cegas

As buscas cegas são assim chamadas porque simplesmente explorar o espaço de estados do problema sem nenhum critério que oriente a pesquisa. Apostam o seu sucesso na probabilidade de que uma solução seja encontrada no caminho explorado. Em geral são muito dispendiosas computacionalmente e em muitos casos não podem ser aplicadas.

Força bruta

Çonsiste em testar todas as soluções possíveis para um determinado problema. Percorre todo o espaço de estados. É o método mais banal e em muitos casos impraticável, pois o número de operações cresce exponencialmente com o problema. Todos os métodos que não se utilizam de uma forma de estimar o melhor caminho no espaço de fases são métodos de **busca cega**, pois simplesmente geram todos os espaço de estados aplicando os operadores possíveis em cada estado. Os métodos de busca em profundidade ou busca em largura, aborados a seguir, são métodos de busca cega. Em muitos problemas não é possível gerar todo o espaço de fases. Nestes caso é necessário qualificar cada um dos caminhos possíveis como mais ou menos promissor; a função de qualificação é chamada **função heurística** e os métodos que procuram os caminhos mais promissores são **métodos heurísticos**.

Busca em profundidade

A busca em profundidade explora um ramo da árvore de estados até o final antes de explorar o ramo seguinte. Um operador é aplicado ao estado inicial,

depois aplicado ao estado resultante da primeira operação e assim sucessivamente até atingir o estado final ou um estado que não permite mais mudança. Na etapa seguinte, volta-se ao vértice anterior não expandido e gera-se daí outro caminho na árvore que vai ser expandido o máximo possível. Em resumo, aplica-se um operador sucessivamente num estado para percorrer um determinado caminho no espaço de estados.

Se o problema em questão possui várias soluções e caminhos que levam à solução, o método de busca em profundidade produz resultados rápidos e satisfatórios. Em comparação com outros métodos requer menos memória porque precisa armazenar somente o caminho que está sendo percorrido.

As desvantagens é ficar perdido em caminhos muito longos ou infinitos. Este problema pode ser resolvido limitando-se a profundidade de busca.

Assim, procura em profundidade é boa quando há muitas soluções possíveis e só se quer uma, não importa qual. É menos apropriada se há uma solução ou se é necessário achar o caminho mais curto.

A figura abaixo mostra a sequência de estados que são explorados na busca em profundidade.

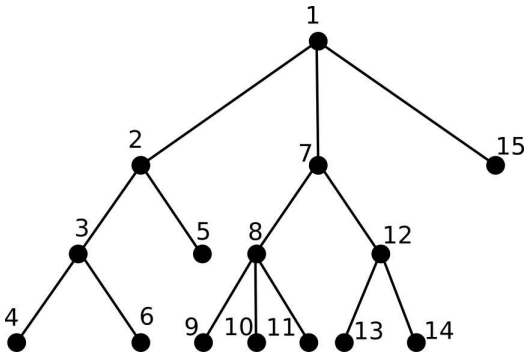


Figura 4.2: Sequência de estados numa busca em profundidade

Busca em largura

A busca em largura consiste em explorar todos os vértices de um determinado nível antes de expandir o próximo nível. Nesta caso, vários operadores são aplicados num mesmo estado, gerando um nível completo do espaço de estados.

A busca em largura usa mais memória, pois a árvore inteira tem de ser armazenada, entretanto não fica presa em caminhos infinitos. Outra vantagem é sempre achar o caminho com o menor número de passos. É mais apropriada para explorar espaços de estados muito amplos onde uma solução que involve

um número pequeno de passos é esperada. Também é útil quando se deseja conhecer todas as soluções.

A figura abaixo mostra a sequência de estados que são explorados na busca em largura.

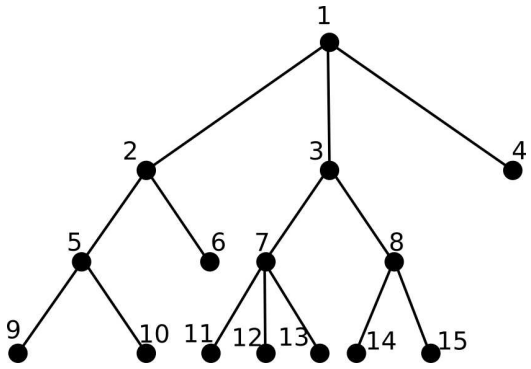


Figura 4.3: Sequência de estados numa busca em largura.

4.4.2 Buscas Heurísticas

Os algortimos anteriores podem ser usados para procurar em todo o espaço de estados, entretanto nem sempre isto é possível, dado a complexidade e extensão do espaço de estados. Neste casos, ao invés de realizar a procura em todos os estados, é necessário focalizar o algoritmo em estados ou caminhos que provavelmente levarão ao estado final ou a um estado desejado. Este é o caso da busca heurística.

O termo **heurístico** refere-se a descoberta, a encontrar algo, neste caso alguma solução. A busca heurística é feita associando-se algum método de verificação se a solução está rumando para um estado mais desejável ou não. A função que faz esta avaliação é chamada **função heurística**. Por exemplo, no caso do caixeiro viajante, a função heurística determinaria a menor trajetória entre dois caminhos possíveis, ou num jogo de xadrez, qual das jogadas possíveis proporcionaria a maior vantagem material ao ser realizada. Um algoritmos heurístico não garante que uma solução será encontrada, mas proporciona uma maneira de avaliar quais soluções são satisfatórias de acordo com determinado critério

Morro-acima

A idéia do algoritmo de morro-acima é sempre partir para um estado que é melhor, no sentido da função heurística, do que o estado atual. O algoritmo

não tenta procurar o melhor caminho possível, mas avalia o trajeto somente a partir do estado atual, logo nenhum registro do caminho percorrido precisa ser mantido.

Entretanto, se existir um elo no espaço de estados o algoritmo pode ficar preso, andando em círculos. O algoritmo termina quando não há estados sucessores melhores que o estado atual, mesmo que o estado em questão não seja o objetivo. Estes problemas estão relacionados com máximos ou mínimos locais no espaço de estados associados com a função heurística escolhida. Uma das formas de resolver é manter certo histórico das ações (estados e operadores) de modo a poder retroceder no caso de uma armadilha. Outra forma é avaliar não só o próximo estado mas um certo número de estados seguintes.

Melhor-primeiro

A busca em profundidade é boa porque pode encontrar uma solução sem calcular todos os vértices. Já a busca em largura é boa porque não fica presa em caminhos sem fim e encontra todas as soluções. A busca do melhor-primeiro permite se beneficiar das características de ambos. A busca do melhor-primeiro é como a busca do morro-acima, porém a do melhor-primeiro é exaustiva, no sentido em que vai percorrer todo o espaço de estados.

O método do melhor-primeiro consiste em expandir primeiro o vértice que é mais promissor, mas posteriormente expande toda a árvore. Inicia-se no estado inicial, expande-se os estados sucessores e atribui-se um valor a cada estado, de acordo com a função heurística. Em seguida, escolhe-se o melhor estado dentre estes; este estado é expandido e o processo repetido até encontrar uma solução ou um ramo sem fim; quando isto acontece, volta-se ao segundo melhor estado mais antigo que não foi expandido e prossegue-se dali com o mesmo processo.

A vantagem da busca melhor-primeiro em relação à busca morro acima é que a morro-acima pode ficar presa num mínimo ou máximo local, isto é, um estado que não tem nenhum estado subsequente que tenha um valor melhor que o estado atual, assim o algoritmo não sairia nunca deste estado. No caso da busca do melhor-primeiro, este estado vai ser expandido primeiro, mas todos os outros estados também serão explorados. Se a função heurística é bem formulada, o método do melhor-primeiro é bastante eficiente em achar uma solução para grande parte dos problemas. Se não se deseja explorar todo o espaço de estados, pode-se estipular a quantidade de caminhos que o algoritmos vai procurar.

A*

A busca do melhor-primeiro é útil em muitas situações, mas não leva em conta o custo do caminho usado para chegar até determinado estado. Assim, pode-se encontrar uma solução que não é tão boa. O algoritmo $A*$ é uma variação do melhor-primeiro que tenta minimizar o custo do caminho utilizado para chegar a uma determinada solução. Ele combina a vantagem da busca em largura, em que o menor caminho é encontrado primeiro, com o benefício da busca do melhor-primeiro em que o estado mais próximo da solução é explorado primeiro.

No algoritmo $A*$, o valor associado a cada estado é uma combinação do custo do caminho até o presente estado e o custo estimado até a solução. Este valor é expresso com um funçao $f(e_i)$, que é a soma da função $g(e_i)$ que estima o custo do caminho desde o estado inicial e a função $h(e_i)$ que estima o custo até o estado final:

$$f(e_i) = g(e_i) + h(e_i)$$

Deste modo, ao tentar escolher um caminho entre os estados, o algoritmo tenta minimizar a função $f(e_i)$ para determinar o caminho com menor custo que une o estado inicial e final.

4.4.3 Decomposição de Problemas

Quando o objetivo é encontrar uma sequências de ações que levem de um estado inicial a um estado final, pode-se simplesmente realizar uma procura no espaço de estados, onde cada vértice representa um estado do sistema e cada aresta uma ação que transforma um estado em outro. Quando representamos o espaço de estados num grafo ou árvore deste tipo (como na Figura 4.6) estamos representando um estrutra chamada de **grafos OU** pois em cada nível o sistema estará num estado *ou* noutro.

Uma outra alternativa para solucionar um problema é representá-lo através de uma estrutura de **grafo E**, em que o problema é resolvível pela decomposição em problemas menores, cuja soluções são unidas (por isso o **E**) para formar a solução geral. No grafo ou árvore **E** as arestas que tem ligação lógica **E** são indicadas por um arco ligando-as, como na Figura 4.4.

Uma estado num grafo **E** pode apontar para qualquer número de vértices sucessores, devendo ser todos resolvidos para que o arco aponte para uma solução. Muitas vezes um grafo é construído misturando-se grafos **E** e grafos **OU**, chamados grafos **E-OU**. O exemplo da Figura 4.4 é um caso em que pode-se realizar uma ação **OU** (outra **E** outra).

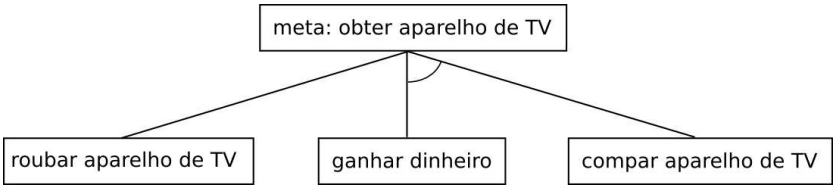


Figura 4.4: Árvore **E-OU**: *"obter um aparelho de TV"* significa *"roubar aparelho de TV"* **OU** (*"ganhar dinheiro"* **E** *"comprar aparelho de TV"*).

Por exemplo, para solucionar o problema de viajar da Uergs em Novo hamburgo (NH) até o Mercado Público em Porto Alegre (PoA) não é preciso explorar todo o espaço de estados desde a saída da Uergs até chegada no Mercado Público. Se o processo consiste em 1) ir até a rodoviária de NH, 2) pegar um ônibus de NH até PoA, 3) ir da rodoviária de PoA até o Mercado Público, o espaço de estados de cada uma das estapas é indepente da outras e o problema pode ser decomposto em 3 problemas mais simples (veja Figura 4.5).

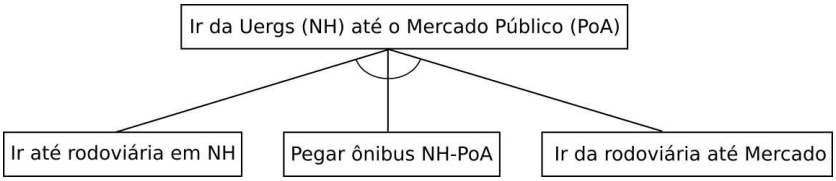


Figura 4.5: Árvore **E**: *"Ir da Uergs-NH até o Mercado Público-PoA"* significa *"ir até rodoviária em NH"* **E** *"pegar ônibus NH-PoA"* **E** *"ir da rodoviária PoA até o Mercado"*.

4.4.4 Satisfação de Restrições

Muitos problemas podem ser interpretados como de **satisfação de restrições**, ou seja, encontar uma solução que satisfaça uma série de vínculos. A maioria do problemas reais trabalho limitado por um conjunto de restrições, sejam elas de tempo, de movimento, de massa, de material e outras. Os métodos de satisfação de restrição podem ser resolvidos pelos métodos de busca explorados até agora. Porém a estrutura destes problemas requer o aumento da complexidade na descrição do estado do problema, com uma lista de restrições que se modifica na medida em que o sistema avança no espaço de estados. O mecanismo de busca deve manipular esta lista do mesmo modo que manipula a lista dos estados e a procura por uma solução final também acontece na lista de restrições. Deste modo, o processo de busca acontece

com duas buscas concomitântes: uma que opera no espaço de estados do problema e outra que opera na lista de restrições vinculadas ao problema. A solução deverá ser encontrada em ambos espaços.

4.5 Aplicação - Problema dos Baldes

Um problema interessante de abordar com a busca é o problema dos baldes: suponha dois baldes, um de 4 litros e outro de 3 litros, ambos sem marcações; é possível enchê-los, esvaziá-los e tranferir o conteúdo de um para o outro. Como se pode colocar 2 L no balde de 4 L, terminando com o de 3 L vazio?

Vamos representar o estado do sistema como um par ordenado $(B4, B3)$ onde $B4$ indica o conteúdo do balde de 4 litros e $B3$ o conteúdo do balde de 3 litros. Deste modo, o estado inicial é $(0, 0)$ e o estado final $(2, 0)$. As regras possíveis estão descritas abaixo; o formato é `{nome [condição]: ação obs}`.

C4	$[x < 4]:$	$(x, y) \mapsto (4, y)$	balde de 4 cheio
C3	$[y < 3]:$	$(x, y) \mapsto (x, 3)$	balde de 3 cheio
V4	$[x > 0]:$	$(x, y) \mapsto (0, y)$	balde de 4 vazio
V3	$[y > 0]:$	$(x, y) \mapsto (x, 0)$	balde de 3 vazio
T43	$[(x > 0) \wedge (y < 3)]:$	$(x, y) \mapsto (x - (3 - y), ?)$	transfere do 4 para o 3
T34	$[(y > 0) \wedge (x < 4)]:$	$(x, y) \mapsto (?, y - (4 - x))$	transfere do 3 para o 4

Assim, partindo de um estado podemos aplicar cada um dos 6 operadores acima, contanto que as condições sejam satisfeitas. Por exmplo, iniciando de $(0, 0)$ somente duas regras podem ser aplicadas, C4 e C3; deste modo são gerados os estados $(4, 0)$ e (0,3) respectivamente. Procedendo deste modo, podemos gerar todos os estados possíveis para o problema e determinar que caminho da solução, isto é, que sequência de estados e operadores que leva do estado inicial ao estado final.

Este procedimento está representado na Figura 4.6. Todos os estados estão mostrados, assim como os operadores que levam de um estado ao outro. Para simplificar o diagrama, alguns estados foram rotulados (I,A,B,C,D). É importante notar que existem pelo menos dois caminhos que levam do estado inicial ao final:

CAMINHO 1: $(2, 0) =$ T34 V4 T34 C3 T34 C3 $(0, 0)$
CAMINHO 2: $(2, 0) =$ V3 T43 C4 T43 V3 T43 C4 $(0, 0)$

Os caminhos estão representados em termos da notação dos operadores, por exemplo o caminho 1 deve ser lido *operador C3 aplicado em* $(0, 0)$*, depois operador T34 aplicado no resultado, depois ...*, ou seja, a sequência de operadores estão da direita para esquerda.

Cada caminho tem características próprias. O caminho 1 só utiliza os operadores $\{C3, T34, V4\}$ e é composto de 6 etapas. Por outro lado, o caminho 2 utiliza os peradores $\{C4, V3, T43\}$ ao longo de 7 etapas. Deste modo, dependendo do caminho utilizado, os aparatos necessários seriam distintos. Neste caso o conhecimento das duas soluções pode ser relevante para o planejamento da estrutura.

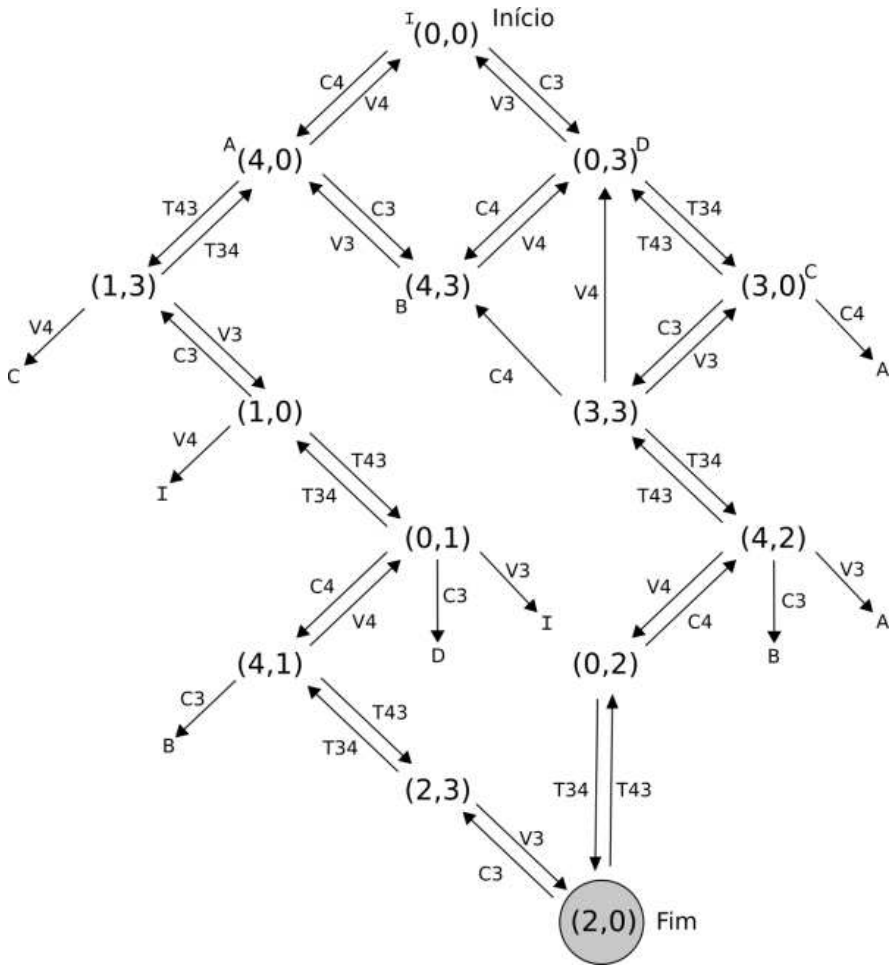


Figura 4.6: Este grafo representa todo o espaço de estados do problema dos baldes. Há dois caminhos possíveis para chegar de $(0, 0)$ até $(2, 0)$.

Capítulo 5

Agentes

5.1 Agente e ambiente

Um agente é todo sistema capaz de perceber o seu ambiente por meio de sensores e agir sobre este ambiente por meio de atuadores.

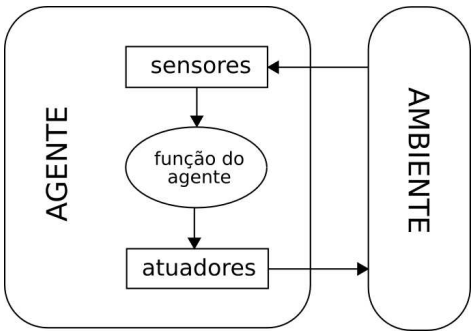


Figura 5.1: Estrutura do agente e interação com ambiente.

As **percepções** são entradas sensoriais e perceptivas do agente. A sensação (conteúdo do sensor) é a medida física de uma quantidade e a percepção é atribuir determinado valor operacional a esta medida. A **sequência de percepções** é a história completa de tudo que o agente já percebeu.

As **ações** são as mudanças que o agente realiza no meio-ambiente por meio de seus atuadores. As ações do agente podem depender de toda a sequência de percepções até o momento.

A **função do agente** é uma função que mapeia uma sequência de percepções numa sequência de ações; é o programa do agente.

Exemplo: Mundo de um alimentador de pássaros.

Características:

- Ambiente: duas gaiolas A e B;
- Percepções: se há comida em cada gaiola;
- Ações: colocar comida em cada gaiola e ir de uma gaiola a outra.

Conjunto de percepções e ações:

PRECEPÇÕES	AÇÕES
`tem(`A`, ração)`	`ir_para(`B`)`
`¬tem(`A`, ração)`	`encher(`A`)`
`tem(`B`, ração)`	`ir_para(`A`)`
`¬tem(`B`, ração)`	`encher(`B`)`

5.2 Características do agente

Racionalidade

Um **agente racional** é aquele que age no sentido do maior sucesso. A **medida de desempenho** do agente é um critério para medir o sucesso do comportamento do agente, uma medida objetiva do seu desempenho. Uma medida de desempenho efetiva deve ser construída de acordo com o resultado desejado no ambiente e não de acordo com o comportamento esperado do agente.

A racionalidade depende dos seguintes fatores:

- medida do desempenho que define o critério de sucesso;
- conhecimento anterior que o agente tem do ambiente;
- ações que o agente pode executar;
- sequência de percepções até o momento.

Para cada sequência de percepções possível, um agente racional deve selecionar uma ação que venha a maximizar a sua medida de desempenho.

Onisciência

Onisciência é ter conhecimento de tudo exatamente. Um agente onisciente sabe o resultado real das suas ações. No mundo real, o agente tenta maximizar o desempenho baseado na sequência de precepções até o momento e com a sua medida do seu desempenho. Se algum fato que não pode ser previsto acontece modificando o desempenho do agente, mesmo assim ele é considerado racional, pois não poderia levar tal fato em conta. Em problemas restritos e artificiais, por exemplo jogos solitários e quebra-cabeças, o agente pode ser onisciente pois sabe exatamente o fruto das suas ações.

Aprendizagem e autonomia

A **aprendizagem** refere-se ao fato de o agente usar as informações coletadas para melhorar o seu desempenho no ambiente. Por exemplo, uma máquina de pipocas foi projetada originalmente para produzir 50% de pipocas doces e 50% salgadas; ao longo do processo, nota que vende-se 80% doces e 20% e ajusta a sua produção para refletir este fato aprendido.

A **autonomia** refere-se ao fato de que o agente deve aprender para compensar um conhecimento prévio parcial ou incorreto.

5.3 Características do Ambiente

O ambiente ou **ambiente de tarefas** é o que caracteriza conjunto de problemas que o agente deve solucionar. Constitui-se de:

- medidas de desempenho;
- ambiente físico;
- atuadores;
- sensores.

O ambiente de tarefas deve ser classificado de acordo com o modo como o agente pode interagir com ele, isto é, mesmo que o ambiente tenha características gerais próprias, ele deve ser classificado de acordo com a percepção que o agente tem dele.

Completamente ou parcialmente observável

O ambiente é **completamente observável** se os sensores são suficientes para determinar as características do sistema completamente. Um ambiente

parcialmente observável é aquele em que se tem acesso somente a certas medidas ou então as medidas contém ruído ou erro.

Determinístico ou estocástico

Num ambiente **determinístico**, o próximo estado é completamente determinado pelo estado atual e pela ação do agente. No ambiente **estocástico**, a sequência de estados pode depender de fatos que não podem ser medidos (não são acessíveis). Se o ambiente é determinístico exceto pela ação de outros agente, diz-se que é **estratégico.**

Episódico ou sequencial

No ambiente **episódico**, a experiência do agente acontece em eventos atômicos, que se constituem de uma percepção e uma ação. Os eventos episódicos não são interdependentes. Em ambientes sequenciais, a decisão atual pode afetar todas as decisões futuras e é necessário fazer planejamento.

Estático ou dinâmico

Se o ambiente se altera enquanto o agente está trabalhando, é um ambiente **dinâmico**, caso contrário é **estático**. Se muda somente o nível de desempenho é **semi-dinâmico**.

Discreto ou contínuo

Esta classificação refere-se ao modo como as ações e percepções são tratadas no tempo. No ambiente **discreto** cada percepção ou ação é bem definida no tempo, enquanto que no ambiente **contínuo** as percepções e ações variam continuamente.

Agente único ou múltiplo

Refere-se ao fato de o ambiente poder ser descrito com de um ou mais agentes. Os eventos do ambiente são considerados agentes se trabalham no sentido de maximizar o sucesso. Se os agentes têm medidas de sucesso que são conflitantes, então o ambiente é multiagente **competitivo**. Se as medidas de sucesso são concordantes, então o ambiente é multiagente **cooperativo**.

Tipo de agente	Medida de desempenho	Ambiente	Atuadores	Sensores
Sistema de diagnóstico médico	paciente saudável, minimizar custos	paciente, hospital, equipe	exibir perguntas, testes, diagnósticos	entrada das informações
Sistema de análise de imagens	classificação correta da imagem	representação da imagem	exibir a categoria da imagem	matriz que representa a imagem
Robô de seleção de peças	fração de peças nos lugares certos	correia de transporte, peças	braços e mãos articulados	câmera e sensores de presença
Controlador de refinaria	maximizar pureza, rendimento e segurança	refinaria, operadores	válvulas, bombas	sensores de temperatura, pressão e produtos químicos

Tabela 5.1: Exemplos de ambientes de tarefas

Capítulo 6

Tópicos Especiais

6.1 Algoritmos genéticos ou evolutivos

Os algoritmos genéticos ou evolutivos (AG) baseiam-se nos princípios da **Teoria Evolucionista** de Charles Darwin para estabelecer uma solução apta a resolver um determinado problema. A Teoria Evolucionista diz que a evolução dos seres vivos acontece por meio da seleção natural, que é a sobrevivência dos mais aptos. As características são passadas de geração a geração pela reprodução e as novas características surgem por mutação genética.

Os algoritmos evolutivos são utilizados no caso de o espaço de estados ser muito extenso para ser percorrido inteiramente e quando uma procura aleatória não produzirá uma solução razoável num espaço de tempo hábil. Suas principais características são:

- Algoritmo de otimização e busca baseado nos mecanismos da seleção natural e da genética.
- Avalia várias possíveis soluções simultaneamente; trabalha com uma população de soluções candidatas ao mesmo tempo.
- Associa uma função de avaliação ou custo com cada possível solução.
- Trabalha com a codificação (representação) dos parâmetros e não com os parâmetros em si.
- Utiliza regras de transição probabilística e não determinística.

Alguns termos são uilizados para definir as características dos AG's:

Indivíduo ou cromossomo: solução candidata.

População: conjunto de indivíduos.

Grau de aptidão: capacidade de adaptação do indivíduo (função de adaptação) que mede a capacidade de solução do indivíduo.

Genótipo: é o conjunto das características do indivíduo.

Fenótipo: é o resultado da combinação das caraterísticas do genótipo. A aptidão do fenótipo determina a sua sobrevivência.

O processo de busca consiste num processo iterativo em que cada geração é avaliada e os indivíduos mais aptos são escolhidos para dar origem a nova geração; os menos aptos são descartados. Os membros restantes estarão sujeitos à mutação ou cruzamento, gerando os descendentes.

6.1.1 Seleção

A seleção determina quais os indivíduos participarão da fase de reprodução. Depois de várias gerações a população deve reproduzir as características dos indivíduos mais aptos.

A seleção é feita com base na função de aptidão. A **função de aptidão** determina a aptidão daquele indivíduo em proporcionar uma solução para o problema. No caso da maximização de uma função $\mathcal{F}(x_1, \ldots, x_N)$ dependente de várias características (genes) $x_1, \ldots, x_N$, a função de aptidão seria a própria $\mathcal{F}$, ou seja, determinar um genótipo (conjunto dos valores de $x_1, \ldots, x_N$) que façam $\mathcal{F}$ ser o máximo. No caso de minimização da função $\mathcal{F}$, a função de aptidão poderia ser $1/\mathcal{F}$, isto é, quanto menor $\mathcal{F}$, maior será a aptidão $1/\mathcal{F}$. Para problemas gerais, a função de aptidão pode reunir vários custos e benefícios de cada uma das características (gene) de uma solução (genótipo).

A seleção acontece no **espaço de seleção**, onde os indivíduos mais aptos ocupam uma parcela maior, tendo maior chance de serem escolhidos

Método da Roleta

Neste método de seleção, os indivíduos serão escolhidos de forma aleatória, mas ponderados pela sua aptidão: constrói-se uma roleta (um selecionador aleatório) com todas os indivíduos presentes; a área coberta por cada indivíduo é proporcional a sua aptidão relativa, que é a sua nota normalizada pelo total das notas. Assim, os indivíduos mais aptos terão mais changes de serem escolhidos. A roleta é acionada N vezes, seleckionando N indivíduos para reprodução.

Métodos do Ordenamento

No método do ordenamento os indivíduos ocuparão uma parcela fixa do espaço de seleção de acordo com a sua ordem geral de aptidão ou notas. Por exemplo, se escolhe-se a fração de 0.45, o primeiro da lista fica com uma fração de 0.45 do espaço de seleção; sobra 0.55; o segundo colocado ocupará 0.45 dos 0.55 restantes, ou 0.2475; sobra 0.45+0.2475; terceiro 0.45 dos 0.3025 restantes, ou 0.136125; e assim por diante.

6.1.2 Reprodução

Os indivíduos selecionados tomarão parte na reprodução. Na reprodução são combinados e modificados, produzindo os indivíduos da próxima geração. A reprodução é realizada pelos **operadores genéticos** que deverão produzir uma população que reuna as melhores características das gerações anteriores.

Operador de cruzamento

O operador de cruzamento combina as características dos pais durante a reprodução. É o operador predominante, aplicado com uma taxa (taxa de cruzamento) maior que os outros. Pode ser:

cruzamento de 1 ponto: um ponto no genótipo é escolhido e a partir daí as características dos progenitores serão trocadas.

cruzamento multipontos: são utilizados vários pontos para a troca das características dos progenitores.

cruzamento uniforme: uma máscara é utilizada para determinar que genes serão herdados de cada progenitor.

Operador de mutação

O operador de mutação introduz novas características na população, assim a change de se chegar a qualquer ponto no espaço de estados é não nula, fazendo com que qualquer solução seja acessível.

Operador de elitismo

O operador de elitismo garante que os indivíduos mais aptos sejam passados diretamente para a outra geração, a fim de que o seu conjunto de características não se perca.

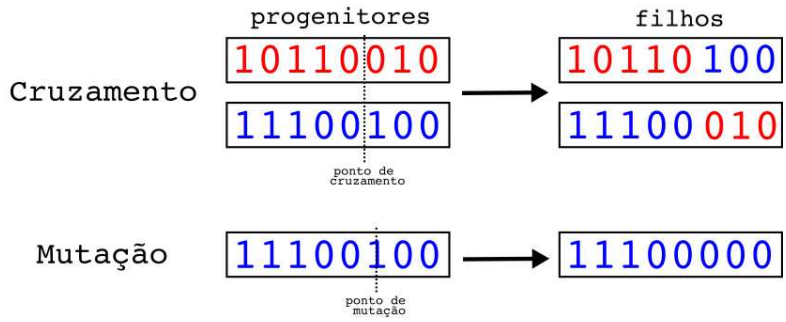


Figura 6.1: Exemplos de reprodução

6.1.3 Parâmetros genéticos

Parâmtros que controlam as características do algoritmo.

Tamanho da população: deve ser escolhida de acordo com um compromisso entre o habilidade do algoritmo em percorrer o espaço de estados e, logo, determinar um solução, e a sua eficiência computacional. Uma população grande tem eficiência de busca mas é cara computacionalmente.

Taxa de cruzamento: controla a introdução de novas combinações das características existentes na população. Se a taxa for alta, bons indivíduos podem ser retirados mais rapidamente que outros melhores tomem os seus lugares; se for muito baixa a busca pode estagnar.

Taxa de mutação: esta taxa introduz novas características na população, impedindo que a busca fique restrita a uma parte do espaço de estados. Entretanto, se a taxa de mutação for muito alta a busca se torna puramente aleatória.

6.1.4 Algoritmo

O algoritmo genético consiste do seguinte:

1. gerar um conjunto de genótipos iniciais;
2. repetir {
   (a) definir aptidão de cada genótipo
   (b) selecionar genótipos mais aptos e descartar resto;
   (c) aplicar operadores de reprodução;

} até obter genótipo adequado ou produzir N gerações.

Apêndice A

Exercícios

A.1 Lógica

1. Quais das seguintes sentenças são proposições? Qual o valor verdade das que são proposições?

1. Brasília é a capital do Brasil.
2. $2 + 3 = 7$
3. Abra a porta!
4. $5 + 7 < 10$
5. A Lua é um satélite da Terra.
6. $x + 5 = 7$
7. $x = 5 > 9$ para cada número real x

2. Qual a negação para cada uma das seguintes proposições?

1. Montevidéu é a capital do Uruguai.
2. A comida não é cara no Brasil.
3. $3 + 5 = 7$
4. O verão em Porto Alegre é quente e ensolarado.

3. Considere as proposições:
P : *o seu carro sem gasolina*
Q : *você não pode dirigir seu carro*
Escreva as seguintes proposições usando P e Q e os conectivos lógicos:

1. o seu carro não está sem gasolina.
2. você não pode dirigir seu carro se ele estiver sem gasolina.
3. seu carro não está sem gasolina se você puder dirigí-lo.
4. se você não puder dirigir seu carro, então ele está sem gasolina.

4. Determine se cada uma das seguintes implicações é verdadeira ou falsa.

1. Se $0,5$ é inteiro então $1 + 0,5 = 5$.
2. Se os carros podem voar, então $1 + 1 = 3$.
3. Se $5 > 2$, então porcos voam.
4. Se 3×5, então $1 + 2 = 3$

5. Escreva o reverso e o contrapositivo de cada uma das seguintes sentenças:

1. Se chover hoje, ficarei em casa.
2. Nos jogaremos se estiver ensolarado.
3. Se um inteiro é primo, então ele não tem outro divisor que ele mesmo.

6. Construa a tabela verdade para cada um das proposições compostas:

1. $P \wedge \neg P$
2. $(P \wedge \neg Q) \rightarrow Q$
3. $(P \rightarrow Q) \leftrightarrow (\neg Q \rightarrow \neg P)$

7. Escreva cada uma das sentenças na forma *"se P, então Q"*:

1. O jornal não sai se há um palmo de chuva na rua.
2. Chove sempre que o vento sopra do sul.

3. Se os preços sobem implica que os estoques estão cheios.

4. é necessário ler o livro para entender o curso.

8. Use a tabela verdade para verificar as seguintes equivalências:

1. $P \wedge \mathbb{F} \Leftrightarrow \mathbb{F}$

2. $P \vee \mathbb{V} \Leftrightarrow \mathbb{V}$

3. $P \vee P \Leftrightarrow P$

4. $P \vee (Q \vee R) \Leftrightarrow (P \wedge Q) \vee (P \wedge R)$

9. Mostre que as seguintes implicações são tautologias:

1. $P \rightarrow (P \vee Q)$

2. $(P \wedge Q) \rightarrow (p \rightarrow Q)$

3. $\neg(P \rightarrow Q) \rightarrow \neg Q$

10. Encontre a proposição dual das seguintes proposições:

1. $P \wedge \neg Q \wedge \neg R$

2. $(P \vee Q \vee R) \wedge S$

3. $(P \wedge \mathbb{F}) \vee (Q \wedge \mathbb{V})$

A.2 Representação do Conhecimento

11. Represente a seguinte frase numa rede semântica: *"A cadeira preta de encosto curvo, que pertence a Júlia está em cima do caminhão novo de Paulo"*

12. Represente a seguintes sentenças numa rede semântica: *pessoas tem duas pernas; Clara é uma professora; Clara tem 1,63 m de altura; professores são um tipo de pessoa; professores trabalham com educação; Ricardo é um programador; Ricardo tem 1,70 m de altura; programadores são um tipo de pessoa; programadores trabalham com computadores;*

13. Represente uma rede semântica com a seguintes informações: *Chevette é um carro; carros tem quatro rodas; Jonas é o dono do Chevette; O Chevette está no estacionamento da Uergs; o tanque de gasolina está vazio; Liberato é uma escola em Novo Hamburgo; a Uergs e a Liberato estão no mesmo prédio.*

14. Represente as informações dos problemas 11, 12 e 13 na forma de predicados.

15. Escreva uma representação de quadros que contenha as principais características de uma cadeira.

16. Escreva as regras de um sistema de produção para caracterizar o tipo de fogo em questão de acordo com o material. sabe-se que:

- Classe A: materiais combustíveis como papel, madeira e tecido;
- Classe B: materiais inflamáveis e combustíveis líquidos, graxas e similares;
- Classe C: equipamentos elétricos;
- Classe D: materiais inflamáveis como magnésio, sódio e potássio.

O tipo de extintor de incêndio que deve ser usado depende da classe do incêndio.

- Classe A: água ou químicos secos;
- Classe B: dióxido de carbono ou espuma;
- Classe C: dióxido de carbono bromotrifluorometano.
- Classe D: rimetoxyboroxina;

Descreva os fatos usando as regras. A entrada do algoritmo são os tipos de materiais. A saída deve indicar que tipo de extintor usar ou outras providências como interromper a energia ou cortar o fluxo de combustível.

A.3 Solução de problemas

17. Baldes – Suponha dois baldes, um de 4 litros e outro de 3 litros, ambos sem marcações; é possível enchê-los, esvaziá-los e tranferir o conteúdo de

um para o outro. Determine como se pode colocar 2 L no balde de 4 L, terminando com o de 3 L vazio.

18. Caixeiro viajante – Um vendedor tem um lista de cidades que ele deve visitar exatamente uma vez. Há estradas diretas entre cada dupla de cidades. Encontre a rota para o vendedor viajar a menor distância possível, iniciando em qualquer uma das cidades e terminando nela.

19. Quebra-cabeças de 8 – As peças do quebra-cabeça abaixo podem movimentar-se dentro do quadrado ocupando o lugar vazio adjacente. Que passos são necessários para colocar as peças organizadas como no lado esquerdo da figura abaixo na forma organizada do lado direito.

2	8	3
1	6	4
7		5

1	2	3
4	5	6
7	8	

20. O lobo, o carneiro e as alfaces – Numa margem de um rio há um lobo, um carneiro e uma cesta de alfaces. É preciso atravessá-los, um de cada vez, para a outra margem do rio, sem deixar o lobo sozinho com o carneiro, nem o carneiro sozinho com as alfaces em nenhuma das margens. Determine como fazer isso.

Referências Bibliográficas

[Bt2001] Bittencourt, Guilherme *Inteligência Artificial – Ferramentas e Teorias*, Editora da UFSC: Florianópolis, 2001.

[Na2000] Nascimento Jr., C. L., Yoneyama, T. *Inteligência Artificial em Controle e Automação*, Editora Edgard Blücher: São Paulo, 2000.

[Ru2004] Russel, S., Norvig, P. *Inteligência Artificial*, Editora Campus: Rio de Janeiro, 2004

[To2005] Toida, S. *Discrete Structures Web Course Material* `http://www.cs.odu.edu/~toida/nerzic/content/web_course.html` visitado em Ago/2005.

[Tu1964] Turing, A. M. *Computing Machine and Intelligence.* Mind, 59: 433-460, 1950.

[Al2005] Alison Cawsey, *Essence of Artificial Intelligence* `http://www.cee.hw.ac.uk/~alison/ai3notes` visitado em Out/2005.

[Da2005] A. Dave Marshall, *Artificial intelligence*, Cardiff University `http://www.cs.cf.ac.uk/Dave/AI2/` visitado em Out/2005.