6 Sistemas Especialistas

"Deus não joga dados."
(Einstein)

6.0 INTRODUÇÃO

Os sistemas especialistas (SE) constituem uma das mais importantes áreas da IA e, diferente de algumas outras áreas de I.A., que ainda continuam sendo apenas promessas para o futuro, os sistemas especialistas têm sido usados comercialmente, há bastante tempo. Isto foi possível, sem dúvida alguma, porque os sistemas especialistas constituem uma proposta muita bem colocada. Ele consiste num sistema computacional destinado a representar o conhecimento (usando alguns dos modelos apresentados), de um ou mais especialistas humanos, sobre um domínio bem específico e, a partir do processamento da base de conhecimento, busca soluções para problemas que, em geral, requerem grande volume de conhecimento especializado.

6.1 COMPONENTES DOS SISTEMAS ESPECIALISTAS

Para serem capazes de ajudar na tomada de decisões, os sistemas especialistas devem ter algumas características, como:

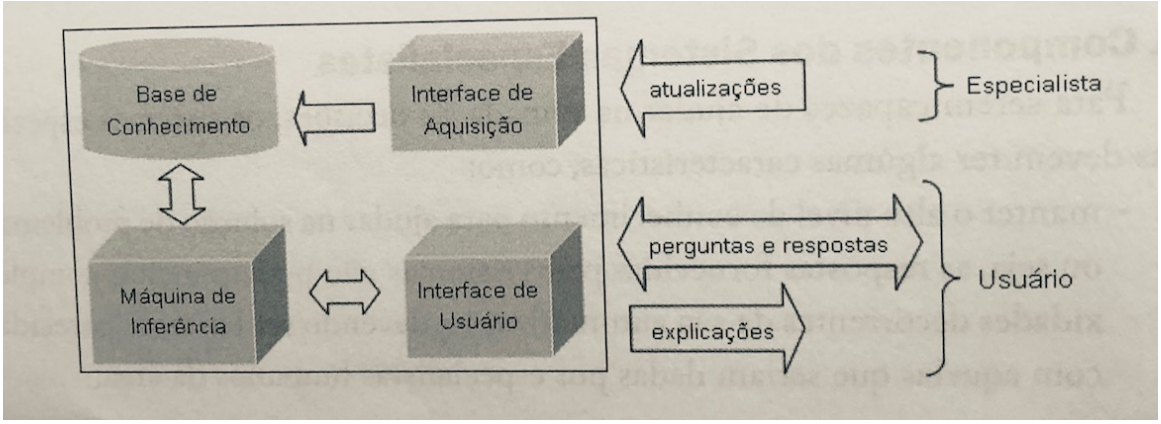
- **manter o alto nível de conhecimento** para ajudar na solução de problemas, ou seja, as respostas fornecidas pelos sistemas não podem incluir complexidades decorrentes da sua automatização, devendo ser bastante parecidas com aquelas que seriam dadas por especialistas humanos da área;
- **contemplar todas as atividades relacionadas ao conhecimento**, desde a sua aquisição até a sua representação, ou seja, o sistema especialista deve auxiliar tanto a tarefa em que a base de conhecimento é alimentada por especialistas da área, quanto a tarefa de processamento, quando um usuário do sistema interage com ele;
- **fornecer explicações referentes às conclusões alcançadas** ou as linhas de raciocínio utilizadas para alcançar uma possível conclusão.

Também é desejável que o sistema seja **flexível**, de modo a facilitar a **atualização**, **visualização** e **compreensão do conhecimento**. O uso de regras **heurísticas** também é recomendado, pois em geral, melhora o seu desempenho, evitando que se tenha que processar toda a base de conhecimento. Outro recurso desejável é que os usuários possam interagir com o sistema usando uma **linguagem natural**, como o português ou o inglês. Espera-se, também, que o sistema especialista possa funcionar com **informações incompletas ou incertas**, o que pode ser feito usando técnicas estatísticas e a lógica nebulosa, que será abordada posteriormente. Por fim, é desejável que o sistema apresente um **desempenho comparável** com um especialista humano em termos de **velocidade, confiabilidade e precisão** de suas recomendações.

Todos os sistemas de computador sempre embutem algum tipo de inteligência e conhecimentos de alguma área em seus códigos e, consequentemente, são capazes de apoiar a resolução de problemas em áreas bem específicas, porém, muitos destes sistemas não podem ser considerados sistemas especialistas.

A **caracterização** de um sistema especialista começa com uma completa **separação** entre os métodos de **solução do problema** e o **conhecimento codificado**. Na prática, tem-se um programa executável que vai buscar em um **arquivo**, a parte, o **conhecimento sobre o seu domínio**. Isto significa que a sua base de conhecimento pode ser completamente alterada e, mesmo assim, o programa irá funcionar normalmente, adotando o conhecimento da nova base.

A Figura ilustra a arquitetura de um Sistema Especialista



6.1.1 Base de Conhecimento (BC)

A base de conhecimento (BC) é o componente responsável pelo **armazenamento do conhecimento**, e deve usar algum dos **modelos de representação** do conhecimento discutidos anteriormente, como: a representação por Lógica Matemática, a representação por Regras de Produção, a representação por Redes Semânticas, a representação por Quadros e Roteiros ou a representação usando Árvores.

Embora todos estes modelos possam ser usados individualmente ou em combinações, o uso de **Regras de Produção** é, sem dúvida alguma, a estratégia mais adotada, o que se deve, principalmente, por causa de duas de suas características: a **modularidade** e a **uniformidade**.

A **modularidade** se dá porque cada regra define um pequeno e independente pedaço do conhecimento, permitindo grande facilidade para adicionar novas regras. Além disso, regras incorretas podem ser facilmente alteradas ou mesmo excluídas da base de conhecimento.

A **uniformidade** se refere ao fato de todas as regras usarem um mesmo padrão de representação, o que permite que mesmo pessoas não acostumadas com o sistema consigam entender o conteúdo do conhecimento armazenado nas regras.

Por fim, os sistemas de produção são interessantes porque também conseguem apoiar, com facilidade, a implementação da habilidade de explicar as decisões e soluções obtidas, pois, quando o usuário do sistema solicita alguma explicação, basta que o sistema indique quais foram as regras usadas para se obter a conclusão apresentada.

As informações para a base de conhecimento podem ser obtidas de várias fontes, como por exemplo: livros, estudos de casos, relatórios, dados empíricos, processos de aprendizagem de máquina e a própria experiência de especialistas. Porém, a tendência atual é incorporar módulos para realizarem a aquisição automática de conhecimento, por meio da implementação de métodos de aprendizado de máquina.

6.1.2 Interface de Aquisição

A interface de aquisição é o componente do SE que permite ao especialista definir e manipular as regras. Assim, a interface de aquisição deve apoiar a construção inicial da base de conhecimentos e também permitir que sejam feitas atualizações, podendo ser correções de regras já existentes ou a adição de novas regras para o tratamento de novos conhecimentos sobre o mesmo domínio.

6.1.3 Interface de Usuário

Esta parte do SE é responsável pela interação do usuário com o sistema e, por meio dela, o usuário utiliza o conhecimento armazenado na base de conhecimento para obter as respostas às suas perguntas e também explicações referentes às linhas de raciocínio que o sistema usa para alcançar uma conclusão. Como em qualquer sistema computacional, a interface de comunicação com o usuário é sempre responsável pelo grau de satisfação do usuário com o sistema, devendo ser sempre muito eficiente e amigável.

6.1.4 Máquina de Inferência

A máquina de inferência é a parte do SE responsável pelo processamento das perguntas do usuário,

processamento dos fatos armazenados na base de conhecimento e pela obtenção das conclusões e explicações que serão fornecidas ao usuário. Os processos de inferência procuram gerar novos conhecimentos a partir de fatos, suposições e conhecimento já estabelecidos na base de conhecimento.

Deste modo, a máquina de inferência transforma uma situação dada (estado inicial) em uma situação desejada (estado final), usando um conjunto de operadores. O processo para a resolução de problemas resume-se em encontrar a sequência de operadores que levam do estado inicial ao estado final. No caso mais comum, usando **regras de produção** para representar o conhecimento, a máquina de inferência pode operar usando o encadeamento **direto** ou o encadeamento **reverso**.

6.1.4.1 Encadeamento Direto (Prova Direta)

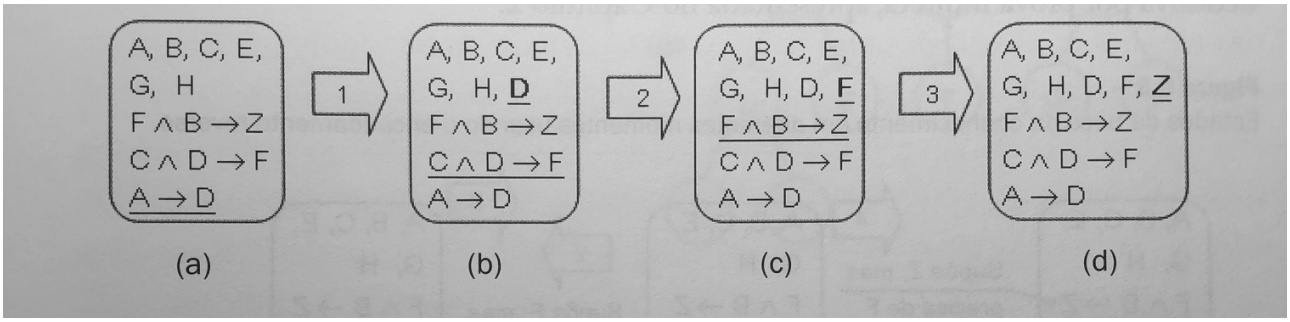
Usando esta modalidade de processamento, a memória de trabalho **recebe dados** sobre o problema e o processo de inferência **deduz outros dados a partir das regras**, comparando os dados da memória de trabalho com as premissas das regras relevantes.

Em seguida, **adiciona** à memória de trabalho os **dados inferidos** (conclusão de regras válidas), além de gerar perguntas ao usuário para confirmar as premissas adicionais de regras com boa possibilidade de aplicação.

Este processo termina quando ocorre a confirmação de todas as premissas de uma regra cuja conclusão possa ser aceita como uma resposta final. Neste caso, a máquina de inferência vai percorrer as regras sequencialmente e, usando os fatos presentes na base de conhecimento, vai tentar provar tudo o que for possível até obter a conclusão que deseja provar.

A Figura mostra um exemplo de processamento usando o encadeamento para frente, que supõe verdadeiras as premissas A, B, C, E, G e H, e deseja provar que Z é verdadeira. A base de conhecimento também possui as regras F ^ B → Z, C ^ D → F e A → D.

Em (a), a única regra que o encadeamento vai conseguir provar é A → D, pois A é verdadeira e, assim, D passa a ser verdadeira (Modus Ponens) e, então, é incluída no conjunto de premissas em (b). Em seguida, o processamento consegue provar que F é verdadeira usando a regra C ^ D → F. Assim, F também é incluída no conjunto de premissas em (c). Agora, a base de conhecimentos cm (c) permite provar a regra F ^ B → Z, finalizando o processo, pois se conclui Z.



Estados da base de conhecimento em diferentes momentos usando o encadeamento direto.

Quando algum fato é desconhecido ou não pode ser provado com as regras existentes na base de conhecimento, a máquina de inferência deve perguntar ao usuário se ele conhece o fato e, no caso de uma resposta positiva, o fato é assumido como verdadeiro e o processamento prossegue.

6.1.4.2 Encadeamento Reverso (Prova Indireta)

Esta estratégia procura usar somente as regras que são relevantes a um problema em questão. Desta maneira, a máquina de inferência vai partir da conclusão a ser provada, tentando provar a validade de suas premissas. Este é o modo de processamento usado pelo Prolog quando tenta obter uma resposta em sua base de conhecimentos. No encadeamento reverso, o processo termina com a conclusão verdadeira quando todas as premissas são provadas. Caso contrário, a conclusão não é verdadeira.

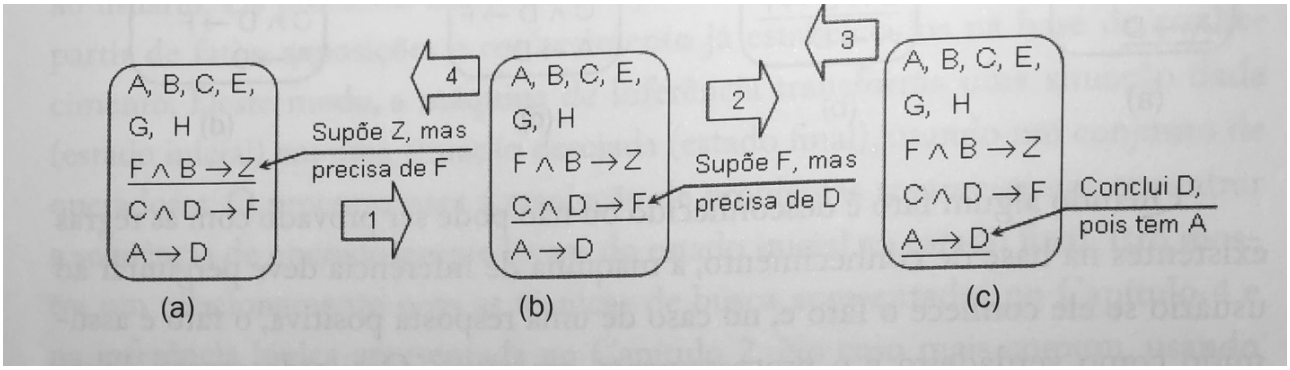
A Figura mostra um exemplo de processamento usando o encadeamento reverso, em que supõe-se as mesmas premissas e regras usadas anteriormente A, B, C, E, G e H, e deseja-se provar que Z é verdadeira, usando-se as regras F ^ B → Z, C ^ D → F e A → D.

Para provar que Z é verdadeira, o encadeamento reverso vai admitir que Z é temporariamente verdadeira e procurar, em (a), a regra que tem Z em sua conclusão, no caso, F ^ B → Z, Porém para prová-la, precisa de B e F, mas apenas B está na base de conhecimento, assim, precisará provar que F é verdadeira, o que também é admitido temporariamente, e uma regra contendo F como conclusão deverá ser avaliada.

Em (b) a regra C ^ D → F é então processada, porém, apenas o C está na base de conhecimento, assim, D também precisa ser provado usando alguma regra. Novamente, a conclusão é admitida como verdadeira e em (c), o processamento vai procurar a regra que tem D como conclusão, ou seja, A → D será avaliada.

Finalmente, esta regra tem como premissa apenas o A que está na base de conhecimento (verdadeira); deste modo, o processo termina com Z sendo realmente verdadeira, pois todas as suposições realizadas até o momento foram satisfeitas.

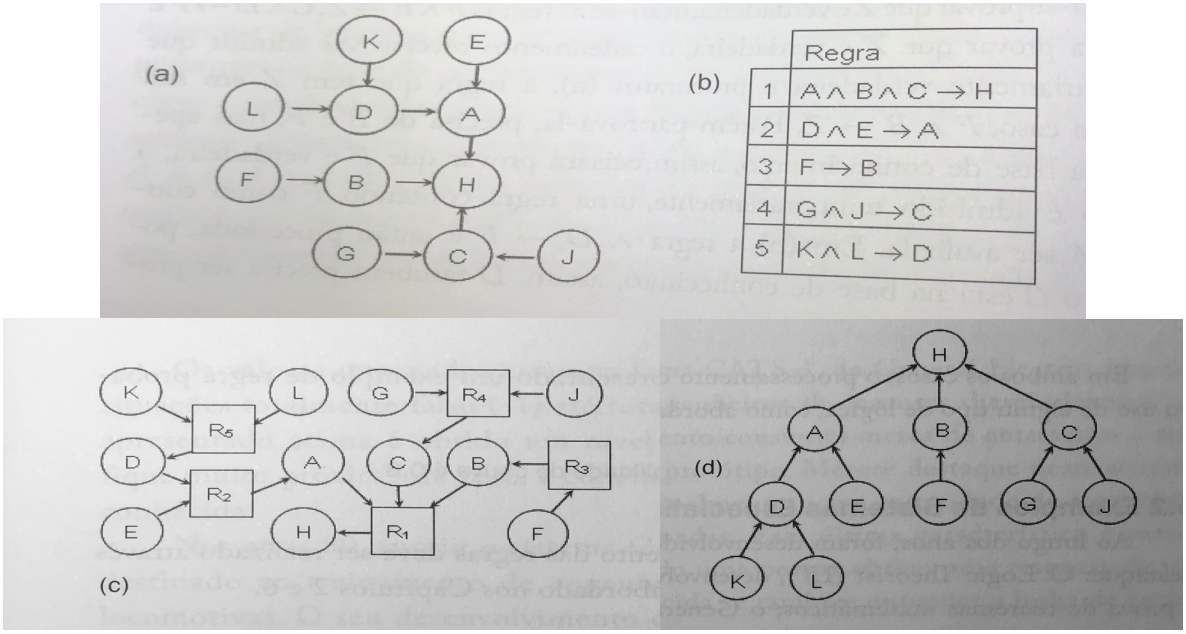
Por outro lado, se alguma das regras avaliadas falhasse, a suposição inicial de que Z é verdadeira seria descartada. O encadeamento reverso corresponde à técnica dedutiva por prova indireta.



Estados da base de conhecimento em diferentes momentos usando o encadeamento reverso.

6.1.4.3 Máquina de Inferência usando Redes Semânticas

A forma de processamento usada na máquina de inferência depende do modelo de representação de conhecimento adotado na base de conhecimento. Conforme mencionado anteriormente, as redes semânticas são estruturas extremamente poderosas, capazes de representar, de diferentes maneiras, o conhecimento embutido em regras e sentenças lógicas. A Figura (a) exibe uma rede que representa o conhecimento embutido nas sentenças lógicas apresentadas em (b). Em (c) as mesmas regras também são representadas por nós, porém, as regras também são incluídas no grafo. Por fim, em (d) tem-se uma outra



Representação do conhecimento usando diferentes modelos.

Usando a árvore ilustrada na Figura (d), a prova de *H* (raiz da árvore) pode ser feita aplicando uma busca em profundidade, neste caso, obtendo o trajeto *H, A, D, K, L, E, B, F, C, G* e *J*. Usando esta sequência, *H* será verdade se A é verdade; o que é provado investigando *D* e *E*. Mas, para provar *D, é* preciso avaliar *K e L*. A busca em profundidade indica que o próximo a ser investigado é *B*, que depende de *F*. Por fim, *C* deverá ser provado, o que é feito a partir de *G* e *J*. A prova de *H* também pode ser feita com uma busca em amplitude e, neste caso, o trajeto adotado seria *H, A, B, C, D, E, F, G, J, K e L*.

6.1.4.4 Outros Aspectos da Implementação da Máquina de Inferência

A máquina de inferência pode ser implementada de forma a trabalhar **deterministicamente** ou **probabilisticamente**. No primeiro caso, deverá fornecer uma resposta única e bem definida a um problema, pois suas regras trabalham de forma exata. Isto está relacionado com o tipo de problema a ser resolvido e das regras usadas. Em seguida é apresentado um exemplo de regra determinística.

Se um átomo tem dois elétrons **Então** é um átomo de Hélio

No modo probabilístico, a máquina de inferência deve ser capaz de fornecer respostas que tenham uma certa possibilidade de ocorrer, o que é comum em um grande número de aplicações. Neste caso, deve-se considerar o peso das variáveis envolvidas. Em seguida é apresentado um exemplo de regra probabilística.

Em janeiro a probabilidade de chuva é 0.8

Em ambos os casos, o processamento das regras deve ser realizado através do uso de algum tipo de lógica.

6.2 EXEMPLOS DE SISTEMAS ESPECIALISTAS

Ao longo dos anos, foram desenvolvidos vários SE e, entre eles merecem destaque: O **Logic Theorist** (LT), desenvolvido nos anos 50 e 60 e destinado a prova de teoremas matemáticos; o **General Problem Solver** (GPS), também destinado a prova de teoremas, geometria e jogos e o **Dendral**, destinado à inferência de estruturas moleculares desconhecidas. O Dendral usava uma representação procedimental do conhecimento e obteve bastante sucesso em publicações científicas. Um exemplo de regra usada pelo Dendral é:

If the spectrum for the molecule has two peaks at masses x1 e x2, such that:
1. x1 + x2 = M + 28 **and**
2. x1 - 28 is a high peak **and**
3. x2 28 is a high peak **and**
4. at least one of x1 or x2 is high **then** the molecule contains a ketone group

O sucesso dos SE nos anos 70 levou ao desenvolvimento de vários deles, merecendo destaque o **Mycin**, que foi construído para auxiliar profissionais da área médica, auxiliando o diagnóstico de doenças infecciosas (bacteremia, meningite e cistite infecciosa). Além do diagnóstico, o sistema também sugeria as terapias mais adequadas em cada caso. Representando o conhecimento médico através de um conjunto com 450 regras de produção, este sistema conseguia obter um sucesso acima de 90% em suas respostas. Um exemplo de regra usada no Mycin é apresentado a seguir:

If 1. the infection is primary-bacteremia **and**
2. the site of the culture is one of the sterile sites **and**
3. the suspeoted portal of entry of the organism is gastrointestinal tract
then there is suggestive evidence (0.7) that the identity of the organism is bactericides

Os valores retornados estão na faixa de "-1 até +1" e correspondem as situações totalmente falso (-1)

até totalmente verdadeiro (+1). No exemplo apresentado acima é obtido um nível de certeza de 70%. Caso este valor fique muito próximo de zero, a conclusão obtida deve ser considerada desconhecida.

Nos anos 80, surgiu o sistema CATS-1, da General Electric, que era destinado ao treinamento de engenheiros de motores diesel-elétricos de locomotivas. O seu desenvolvimento consumiu meses de entrevistas e três anos para se chegar ao primeiro protótipo. Merece destaque neste sistema a sua interface amigável e, principalmente, a sua capacidade de apresentar explicações para as decisões tomadas. Esta última característica é extremamente importante, pois quando uma pessoa obtém uma resposta de um especialista, ela fica mais convencida se também entender a linha de raciocínio adotada.

O sucesso contínuo dos SE tem proporcionado o surgimento de várias empresas que utilizam os seus princípios de funcionamento nas mais diferentes aplicações. Os editores de texto atuais, como o Word, da Microsoft, e vários outros, podem ser classificados como SE, pois o código existente no programa (Word.exe) é capaz de abrir uma base de conhecimentos sobre uma certa língua (idioma) e, a partir daí, o programa se comporta baseado nas regras da língua selecionada. Como os conhecimentos sobre a língua escolhida estão guardados em bases separadas do código, a troca de idioma a ser usado é simplificada. Além da análise sintática, na maioria das vezes, os editores também realizam com sucesso a análise semântica das frases. Por fim, também existe a possibilidade de se acrescentar palavras novas às bases de conhecimento, o que, no caso, é feito pelo próprio usuário, que deve ser responsável pela qualidade do conhecimento que é acrescentado na base. Desta maneira o usuário atua como o usuário do sistema e também como o especialista que alimenta a base de conhecimentos, quando manda adicionar ao dicionário (base de palavras do Word) uma nova palavra.

6.3 APLICAÇÃO

Um exemplo simples de base de conhecimento é apresentado a seguir. Trata-se de um conjunto com 17 regras a respeito de animais, que por causa da sua simplicidade, tem sido usado em várias publicações.

R_1 **Se** tem pelos **Então** mamífero
R_2 **Se** dá leite **Então** mamífero
R_3 **Se** tem penas **Então** ave
R_4 **Se** voa e ovíparo **Então** ave
R_5 **Se** come carne **Então** carnívoro
R_6 **Se** tem dentes pontiagudos, garras e olhos à frente **Então** é carnívoro
R_7 **Se** mamífero, tem cascos **Então** ungulado
R_8 **Se** mamífero e rumina **Então** ungulado
R_9 **Se** mamífero, tem manchas negras, cor fulva e é carnívoro, **Então** puma
R_{10} **Se** mamífero, tem listras negras, cor fulva e é carnívoro **Então** tigre
R_{11} **Se** ungulado, tem pescoço comprido, pernas compridas, manchas negras **Então** girafa
R_{12} **Se** é ungulado e tem listras negras **Então** zebra
R_{13} **Se** ave, não voa, preto e branco e tem pescoço comprido **Então** avestruz
R_{14} **Se** ave, não voa, nada e é preto e branco **Então** pinguim
R_{15} **Se** ave e voa **Então** albatroz
R_{16} **Se** mamífero e voa **Então** morcego
R_{17} **Se** mamífero e não tem pelos **Então** baleia

O desafio neste caso é construir um programa capaz de manipular estas regras, usando o encadeamento direto ou reverso, para identificar os animais, a partir de premissas que devem ser informadas pelo usuário. Deve-se observar que o código fonte não pode codificar as regras, que devem estar dentro de um arquivo à parte, como por exemplo, um TXT. Deste modo, caso as regras sejam alteradas, o programa ainda deverá operar normalmente.

6.4 REFERÊNCIAS BIBLIOGRÁFICAS

[1] Levine, R.I.; Drang, D.E.; Edelson, B. *Inteligência Artificial e Sistemas Especialistas - aplicações e*

exemplos práticos. McGraw-Hill, São Paulo, 1986.

[2] Artero, A.O. *Inteligência Artificial: Teórica e Prática*. ,Livraria da Física, São Paulo, 2009.

[3] Passos, E.L. *Inteligência Artificial e Sistemas Especialistas ao Alcance de Todos*, LTC-Livros técnicos e científicos, Rio de Janeiro, 1989.