UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE UM SISTEMA ESPECIALISTA PARA A SELEÇÃO DE MICROCOMPUTADORES UTILIZANDO A FERRAMENTA JESS

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

RONALDO CÉSAR SCHORK JÚNIOR

BLUMENAU, JUNHO/2002

2002/1-68

PROTÓTIPO DE UM SISTEMA ESPECIALISTA PARA A SELEÇÃO DE MICROCOMPUTADORES UTILIZANDO A FERRAMENTA JESS

RONALDO CÉSAR SCHORK JÚNIOR

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Roberto Heinzle — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Roberto Heinzle

_____

Prof. Alexander Roberto Valdameri

_____

Prof. Luiz Bianchi

DEDICATÓRIA

Dedico este trabalho aos meus pais Ronaldo e Daisy, aos meus irmãos Diogo e Juliana e a minha namorada Carla.

AGRADECIMENTOS

Agradeço principalmente aos meus pais Ronaldo César Schork e Daisy Schork, pela presença em todos os momentos de minha vida e pelo apoio e confiança em mim depositados.

Aos meus irmãos Juliana Cristina Schork de Borba e Diogo César Schork, pela amizade e companheirismo, e também a minha namorada Carla Cassiana Brandão pelo apoio e incentivo dedicado nas horas mais difíceis.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema especialista para auxiliar na escolha de microcomputadores para uso pessoal, utilizando a ferramenta Jess. Para a representação do conhecimento foi utilizada a técnica de regras de produção.

ABSTRACT

This work presents the development of an expert system to auxiliary in the choice of microcomputers for personal use, using the tool Jess. For the knowledge representation the technique of production rules was used.

1 INTRODUÇÃO

A Inteligência Artificial (IA) vem conquistando um espaço cada vez maior ao longo dos últimos anos, e tem se consolidado como uma poderosa ferramenta para elevar a produtividade nas empresas. A IA tem como objetivo tornar os computadores mais úteis para compreender os princípios que tornam a inteligência possível.

Para a utilização de IA no desenvolvimento de aplicações que visam atender uma necessidade específica, apresenta-se a técnica de Sistemas Especialistas (SE), que segundo Lia (1995), são programas de computador que procuram atingir soluções de determinados problemas do mesmo modo que se supõe que os especialistas humanos resolvam, se estiverem sob as mesmas condições.

Para auxiliar na construção de SE foram criadas ferramentas de Inteligência Artificial orientadas para engenharia do conhecimento e construção de SE denominadas *shells*, que estão aptas a realizar muito do trabalho necessário para transpor um SE para o computador. Essas ferramentas permitem que o desenvolvedor do sistema preocupe-se somente com a representação do conhecimento do especialista, deixando para a *shell* a tarefa de interpretar o conhecimento representado e executá-lo em uma máquina. A principal função de uma *shell* é simplificar ao máximo o trabalho de implementação de um sistema especialista e permitir seu uso por qualquer pessoa sem conhecimentos de informática (Heinzle, 1995).

Neste trabalho será utilizada a *shell* Jess, que possui um mecanismo de regras e um ambiente de desenvolvimento escrito em Java. Foi originalmente inspirada pela *shell* CLIPS, mas cresceu com forte influência do Java. Com esta ferramenta é possível construir *applets* e aplicações que tem a capacidade de raciocinar usando o conhecimento fornecido através de regras declarativas.

Com esta ferramenta, será desenvolvido um protótipo de um SE para auxiliar na escolha de um microcomputador para uso pessoal, visando facilitar ao vendedor e ao comprador chegar a uma solução que satisfaça ambos.

1.1 MOTIVAÇÃO

Percebe-se o crescente número de trabalhos que utilizam técnicas de inteligência artificial, oriundos da necessidade de se resolver problemas do mundo real que necessitam da interpretação de um especialista. Estes problemas são resolvidos através de um modelo computacional de raciocínio de um especialista, permitindo que o sistema chegue as mesmas conclusões que este especialista humano chegaria.

As ferramentas para o desenvolvimento de sistemas especialistas mais conhecidas no meio acadêmico são o *Expert SINTA*, o *Arity Prolog*, o *SPIRIT* e o *CLIPS*. Diante disto, busca-se conhecer novas ferramentas que possibilitem o desenvolvimento de sistemas baseados no conhecimento de um especialista, e no caso específico deste trabalho foi estudada a ferramenta Jess.

1.2 OBJETIVOS

Desenvolver um protótipo de um SE para auxiliar na seleção de microcomputadores para uso pessoal, utilizando a ferramenta Jess.

Os objetivos específicos do trabalho são:

a) construir uma base de conhecimento dos microcomputadores e periféricos disponíveis;

b) definir regras que auxiliem na tomada de decisão;

c) disponibilizar um sistema que aplique as regras utilizando a ferramenta Jess.

1.3 ESTRUTURA DO TRABALHO

O primeiro capítulo apresenta uma introdução sobre o trabalho, os motivos que o levaram a ser desenvolvido, e os objetivos a serem alcançados.

O segundo capítulo abrange o conceito de Sistemas Especialistas, um breve histórico, características, a sua estrutura e a formalização do conhecimento.

No terceiro capítulo é apresentada a ferramenta Jess. São descritas as suas principais características e como pode-se construir a base de conhecimentos.

O quarto capítulo apresenta pontos relevantes que devem ser considerados na escolha dos componentes de um computador.

O quinto capítulo demonstra a especificação e implementação do protótipo utilizando a ferramenta Jess.

O sexto capítulo apresenta as conclusões, limitações e sugestões.

2 SISTEMAS ESPECIALISTAS

Os Sistemas Especialistas (SE), são sistemas computacionais projetados e desenvolvidos para solucionarem problemas que normalmente exigem especialistas humanos com conhecimento na área de domínio da aplicação. Tal como um especialista o sistema deve ser capaz de emitir decisões justificadas acerca de um determinado assunto a partir de uma substancial base de conhecimentos. Para tomar uma decisão o especialista busca em sua memória conhecimentos prévios, formula hipóteses, verifica os fatos que encontra e compara-os com as informações já conhecidas e então emite a decisão (Heinzle, 1995).

Um sistema convencional é baseado em um algoritmo, emite um resultado final, normalmente correto e processa um volume de dados de maneira repetitiva, enquanto que um SE é baseado em uma busca heurística e trabalha com problemas para os quais não existe uma solução convencional organizada de forma algorítmica disponível (Souza, 2001a).

Um SE é inicialmente projetado e desenvolvido para atender a uma aplicação determinada e limitada do conhecimento humano. É capaz de emitir um decisão, tal qual um especialista humano emitiria, apoiando-se em conhecimento justificado, a partir de uma base de conhecimentos (Souza, 2001a).

2.1 CONCEITOS

Alguns autores apresentam definições formais sobre sistemas especialistas. Segundo Ribeiro (1987), “um sistemas especialista é aquele que é projetado e desenvolvido para atender a uma aplicação determinada e limitada do conhecimento humano. É capaz de emitir uma decisão, com apoio em conhecimento justificado, a partir de uma base de informações, tal qual um especialista de determinada área do conhecimento humano”.

Para Lia (1995), “sistemas especialistas são programas de computador que procuram atingir soluções de determinados problemas do mesmo modo que especialistas humanos se estiverem sob as mesmas condições”.

Rabuske (1995) escreve: “sistemas especialistas devem, também, ter habilidade para aprender com a experiência e explicar o que estão fazendo e porque o fazem. Esta última é

uma das principais características que distinguem estes sistemas dos tradicionais sistemas de informações".

2.2 ABORDAGEM HISTÓRICA

No início da década de 1960 começaram os primeiros trabalhos que originariam os SE. A idéia inicial era de construir máquinas inteligentes com grande poder de raciocínio para a solução de problemas, e segundo Heinzle (1995), "imaginava-se que a partir de um pequeno conjunto de normas ou regras de raciocínio introduzidas num poderoso computador criaríam-se sistemas de capacidade superior a humana. Não tardou para que os pesquisadores observassem o engano e verificassem as reais dimensões do trabalho".

Um dos primeiros sistemas especialistas a ser construído foi o DENDRAL, em 1964, por Joshua Lederberg da Universidade de Stanford. O DENDRAL deduz a possível estrutura de um determinado composto químico, a partir de um determinado conjunto de dados como massa espectográfica e ressonância magnética, mas sua aplicabilidade ficou restrita ao meio acadêmico.

Em 1968, surge no MIT – *Massachusetts Institute of Technology*, o MACSYMA, destinado a auxiliar matemáticos na resolução de problemas complexos. O programa foi originalmente elaborado por Carol Engleman, William Martin e Joel Mores. O MACSYMA é um sistema ainda hoje amplamente utilizado em universidades e laboratórios de pesquisa (Pereira, 2000).

Na década de 1970, surgiram importantes e complexos SE como o MYCIN e o PROSPECTOR. O MYCIN é um sistema que atua área médica para detectar doenças infeciosas e recomendar um tratamento de acordo com a sua base de conhecimentos. O MYCIN deu origem a *Shell* EMYCIN. O PROSPECTOR é um sistema para auxiliar geólogos na exploração do mineral. Nesta década também surgiu o CASNET, para diagnosticar glaucomas. O CASNET deu origem a ferramenta para construção de SE *Expert* (Cunha[1], apud Pereira, 2000).

[1]CUNHA, Horácio da e Ribeiro, Souza. **Introdução aos sistemas especialistas**. São Paulo, Livros Técnicos e Científicos, 1987.

2.3 CARACTERÍSTICAS

Os SE são também conhecidos como sistemas baseados em conhecimento. O processo de construção de um SE é geralmente chamado de engenharia do conhecimento, a qual tipicamente envolve uma forma especial de interação entre o construtor do sistema, chamado de engenheiro do conhecimento, e um ou mais especialistas. O engenheiro do conhecimento "extrai" do especialista humano, seus procedimentos, estratégias e regras para a solução do problema, e constrói seu conhecimento dentro do SE (Waterman, 1986).

Os SE podem atuar como uma teoria de processamento de informação ou modelo de solução de problemas em um dado domínio, fornecendo as respostas desejadas para um dado problema e mostrando como eles poderiam se ajustar a novas situações.

Outra característica dos sistemas especialistas é a utilização de técnicas de inferência para manipular informações visando uma solução. O mecanismo de inferência utiliza estratégias genéricas para adquirir conhecimento, processá-lo, tirar conclusões próprias e dar explicações acerca do processo de raciocínio. Essa abordagem baseada em conhecimento oferece a possibilidade de separar o conhecimento que descreve o domínio do problema do código de procedimentos que examina este conhecimento. Este mecanismo distingue os sistemas especialistas dos tradicionais (Heinzle, 1995).

O SE pode explicar em detalhes como uma nova situação conduz a mudanças. Ele permite ao usuário avaliar o efeito de novos fatos ou dados, e entender o relacionamento deles com a solução, avaliar os efeitos de novas estratégias ou procedimentos aplicados à solução.

Conforme descreve Souza (2001a), um SE é uma fonte prática capaz de lidar com problemas complexos, resolver questões que requerem um alto nível de juízo e perícia humana, e comunicar-se com seu usuário através de um diálogo eficaz. Os SE fazem perguntas, fornecem conclusões e as justificam.

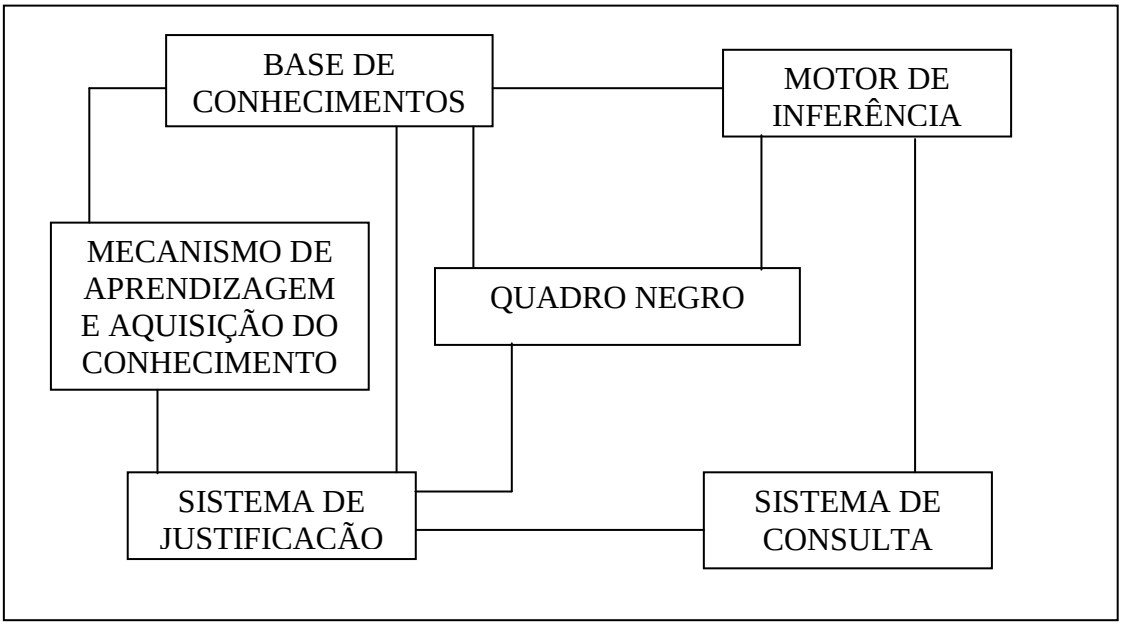
De acordo com Souza (2001a), os SE são caracterizados por:

a) utilizar lógica simbólica ao invés de cálculos numéricos;
b) incorporar uma base de conhecimento;
c) fazer inferências e deduções a partir de uma informação disponível;
d) ter capacidade para explicar suas conclusões.

2.4 ESTRUTURA DE UM SISTEMA ESPECIALISTA

Segundo Rabuske (1995), “os componentes de um SE sofrem influências, as mais variadas, desde a generalidade pretendida, os objetivos que motivaram a sua construção, a representação interna do conhecimento e as ferramentas usadas na implementação”. O modelo geral da arquitetura de um sistema especialista apresentada por vários autores é mostrado na figura1. Especificamente, porém, a sua arquitetura depende da forma de representação do conhecimento e implementação adotada.

Figura 1 - ESTRUTURA DE UM SISTEMA ESPECIALISTA



Fonte: (Heinzle, 1995)

2.4.1 BASE DE CONHECIMENTO

A base de conhecimento é o local onde estão armazenadas as informações de um SE, fatos e regras. Este conhecimento é passado ao sistema pelo especialista e armazenado de uma forma própria que permitirá ao sistema fazer posteriormente o processo de inferência. Um novo fato pode modificar todo o processo de inferência de acordo com as regras existentes sobre ele que estão sendo aplicadas e também sobre os novos fatos gerados pela avaliação dessas regras (Ribeiro, 1987).

2.4.2 MÁQUINA OU MOTOR DE INFERÊNCIA

A máquina de inferência é o mecanismo que procura as respostas na base de conhecimento. Busca as regras necessárias a serem avaliadas e ordena-as de uma maneira lógica. A partir daí, direciona o processo de inferência. Funciona como um supervisor, que dirige a operação sobre o conhecimento contido no SE. Uma máquina de inferência toma decisões e julgamentos baseados em dados simbólicos contidos na base de conhecimento. As funções básicas da máquina de inferência são inferência e controle. Depois de iniciado o sistema, a máquina de inferência busca na base de conhecimento, fatos e regras, e compara estes fatos com a informação fornecida pelo usuário. A operação da máquina de inferência é baseada em algoritmos que definem a busca específica e a unificação de regras. Basicamente, a máquina de inferência compara a entrada fornecida pelo usuário comas regras contidas na base de conhecimento buscando combinações (Souza, 2001a).

Na máquina de inferência são implementados modos de raciocínio, técnicas e estratégias de busca, resolução de conflitos e tratamento de incerteza. Conforme Weiss (1988), os SE geralmente adotam uma das seguintes estratégias de raciocínio:

a) raciocínio para frente (*forward*): o sistema é dirigido pelos dados, parte de fatos conhecidos e tenta deduzir novos fatos, através da máquina de inferência, até chegar a solução;

b) raciocínio para trás (*backward*): dirigido pela meta. O sistema faz o caminho inverso partindo da solução do problema (meta) e tenta verificar se é verdadeira através de suas condições, que passam a ser então submetas a serem também provadas. Isto ocorre sucessivamente até se chegar a um conjunto de condições verificáveis.

Outra técnica que pode ser utilizada é a de encadeamento lateral (*sideways*), cujo procedimento é baseado em prioridades para os itens envolvidos no processo de busca.

O tratamento de incerteza é uma ativa área de pesquisa em SE, pois os domínios adequados a sua implementação, caracterizam-se por não serem modelados por nenhuma teoria geral, o que implica descrições incompletas, inexatas ou incertas.

Graus de confiança são freqüentemente atribuídos as suas respostas, principalmente quando existe mais de uma. Este, sem dúvida, é um dos mais fortes pontos críticos na elaboração de uma representação computacional do saber humano. Conforme Lia (1995),

existem duas correntes de pensamento: aquela que utiliza fórmulas estatísticas rigorosas, com teoria das probabilidades, e aquela que utiliza uma abordagem da teoria das possibilidades sobre os fatores de certeza, ou seja, mais generalizada e sem uma base matemática forte.

2.4.3 SISTEMA DE CONSULTA

O usuário é, geralmente, alguém que não participou da elaboração do sistema, sendo, portanto, natural que não conheça as estruturas do sistema e, que, provavelmente, não esteja familiarizado com as formas de representação do conhecimento adotadas. Para que os potenciais usuários possam acessar com proveito e sem maiores dificuldades o sistema especialista, é preciso muni-los de recursos para consulta (Alexandre, 2000).

A maioria dos sistemas existentes usam técnicas simples de interação com o usuário, quase sempre utilizando perguntas já pré-formatadas e respostas tipo múltipla escolha. Outra técnica é a definição de uma sintética simples com um vocabulário restrito e limitado, própria para utilização do sistema. Recentemente, entretanto, intensas pesquisas tem sido feitas no sentido de tornar o computador capaz de entender a linguagem natural humana. Esta tecnologia é todavia um outro campo de estudo da inteligência artificial cujo desenvolvimento será de extrema valia para toda a área da computação. (Heinzle, 1995).

2.4.4 SISTEMA DE JUSTIFICAÇÃO

A justificação é um requisito importante dos SE. Tem a função de esclarecer o usuário a respeito de uma conclusão apresentada pelo sistema ou ainda explicar porque uma pergunta está sendo feita. Em muitos domínios nos quais os sistemas operam, as pessoas não aceitam resultados se esses não estiverem devidamente justificados (Heinzle, 1995).

Segundo Ribeiro (1987), “este módulo interage com o usuário esclarecendo-o de como o sistema chegou a determinada conclusão, ou por que está fazendo determinada pergunta. Utiliza diversos recursos e estruturas próprias para atender ao seu objetivo, mostrando que regra e que fatos foram usados da base de conhecimento, sempre que isso for solicitado por quem usa o sistema”.

2.4.5 QUADRO NEGRO

O quadro-negro é a área de trabalho que o sistema utiliza durante o processo de inferência. Nesta área são armazenadas informações de apoio e suporte ao funcionamento do sistema quando este está raciocinando. Este lugar na memória é destinado para fazer avaliações das regras que são recuperadas da base de conhecimento para se chegar a uma solução. As informações são gravadas e apagadas de um processo de inferência até se chegar à solução desejada. Embora todos os SE usem o quadro-negro, nem todos o explicitam como componente do sistema (Rabuske, 1995).

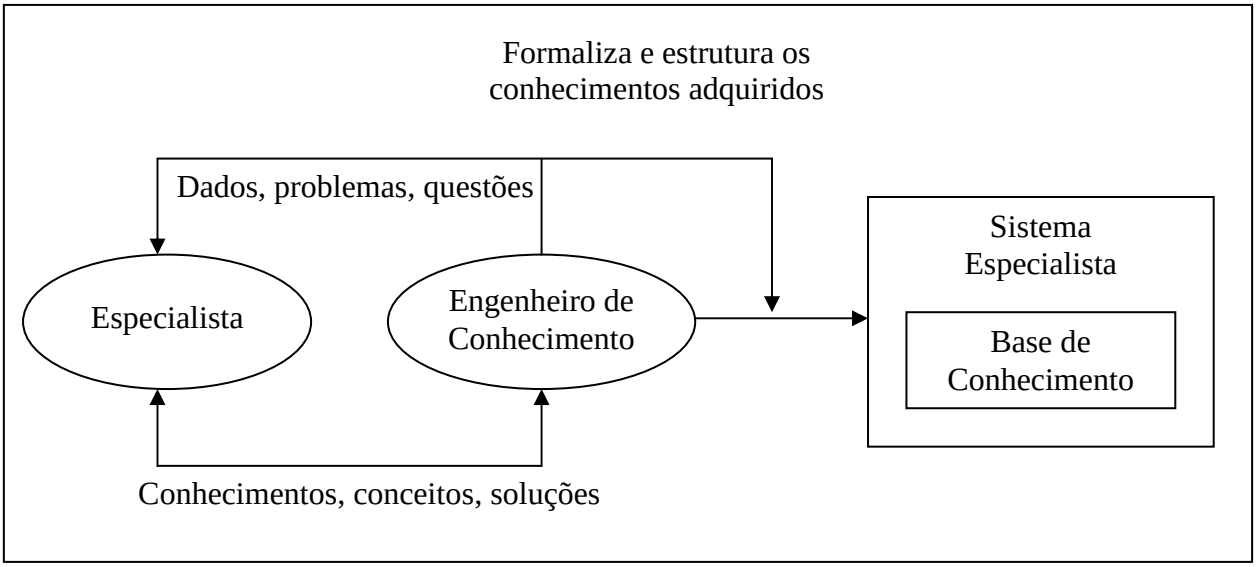
2.5 AQUISIÇÃO DO CONHECIMENTO

O processo de aquisição do conhecimento tem por objetivo compor o corpo do conhecimento sobre o problema de interesse que pode ser sistematizado em um SE. Este processo de aquisição refere-se à transferência de conhecimento de alguma fonte, freqüentemente humana, para um programa de computador. No contexto dos SE, é o processo de captar procedimentos, regras, métodos, enfim, o raciocínio do especialista no que tange a como ele resolve o problema para posteriormente transferí-lo para o sistema. (Lopes, 1997).

Esta etapa pode ser considerada o "gargalo" do processo de desenvolvimento de um SE, pois está relacionado com a dificuldade de transmissão do conhecimento por parte do especialista; ou porque o conhecimento não é bem definido, ou porque é difícil expressar este conhecimento em palavras. Da mesma forma, fatores não previstos podem afetar o processo.

A figura 2 demostra o processo de aquisição do conhecimento:

Figura 2 - PROCESSO DE AQUISIÇÃO DE CONHECIMENTO



Fonte: (Souza, 2001a)

2.6 REPRESENTAÇÃO DO CONHECIMENTO

Representação do conhecimento são métodos utilizados para modular os conhecimentos de especialistas em algum campo, de forma eficiente, e colocá-los prontos para serem acessados pelo usuário de um sistema inteligente e pela máquina de inferência. Ou seja, representação do conhecimento é uma combinação de estruturas de dados e procedimentos interpretativos, que, se usados corretamente em um programa, terão uma conduta inteligente (Luchtenberg, 2000).

A maioria das pesquisas atuais sustentam amplamente que é inútil considerar uma representação, sem considerar o raciocínio que será realizado sobre a mesma. Assim, a área de representação do conhecimento tem sido claramente padronizada para “Representação de Conhecimento e Raciocínio”. Como resultado, as pesquisas estão enraizadas no estudo das lógicas em geral, onde sintaxes formais de linguagens são acompanhadas por regras de inferência e interpretações. É importante considerar que uma linguagem de representação de conhecimento não deve ser caracterizada somente em termos de sua adequação mas também em termos de sua eficácia computacional. Assim, uma representação não deve meramente prescrever como trechos individuais de informações são representados, mas deve especificar como a totalidade da informação é estruturada e organizada de modo que as informações

relevantes possam ser recuperadas e que as inferências adequadas apresentem um nível aceitável de eficiência (Chaiben, 1999).

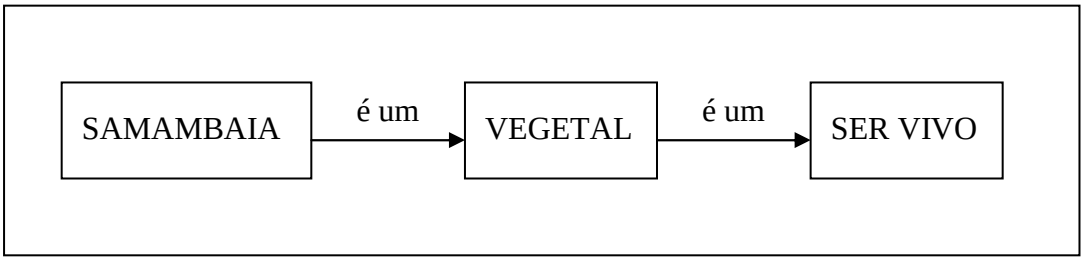
Existem algumas técnicas que permitem modelar o conhecimento de forma eficiente, e mais conhecidas serão explanadas a seguir:

2.6.1 REDES SEMÂNTICAS

As redes semânticas são representações gráficas do conhecimento e foram originalmente desenvolvidas para modelagem psicológica da memória humana, constituindo-se agora num método de representação padrão. Podem ser ilustradas por diagramas que contém nós e arcos. Os nós representam objetos, ações e eventos, os arcos que os ligam representam relações sobre os nós. Arcos comuns para representações hierárquicas são “é um” e “parte de”.

Um exemplo poderia ser citado para definir redes semânticas, onde usamos as ligações para significar “é-um” apontando do nó representando VEGETAL para o nó representado SER VIVO. A ligação “é-um” partindo da SAMAMBAIA para VEGETAL, permite ao sistema inferir que SAMAMBAIA é um subconjunto de VEGETAL, que por sua vez é um subconjunto de SER VIVO, então que SAMAMBAIA é um SER VIVO (Souza, 2001b).

Figura 3 - REPRESENTAÇÃO DE UMA REDE SEMÂNTICA



Fonte: (Souza, 2001b)

2.6.2 QUADROS OU FRAMES

É uma estrutura de dados que inclui informação declarativa e procedural em relações internas pré-definidas. Um *frame* é organizado de maneira muito parecida com uma Rede Semântica, sendo uma rede de nodos e relações organizados numa hierarquia, onde os nodos do topo representam conceitos gerais e os nodos mais baixos representam instâncias mais

específicas destes conceitos. Isto parece uma rede, mas em um sistema de *frame* o conceito de nodo é definido por uma coleção de atributos e valores destes atributos, onde os atributos são chamados de *slots*. Cada *slot* tem um número qualquer de procedimentos anexados a si, que são executados automaticamente quando a informação contida no *slot* é recuperada ou alterada. (Luchtenberg, 2000).

Um *frame* é constituído por um nome, uma coleção de atributos, chamados de escaninhos ou *slots* e valores associados a eles (Rabuske, 1995). Através do *frame* "Cadeira-do-Renato", abaixo, tem-se um exemplo de como se apresenta conforme Rabuske (1995):

Quadro: Cadeira-do-Renato

Slot: número-de-pernas – 4

Slot: tipo-de-assento – anatômico

Slot: número-de-braços – nenhum

Slot: cor – incolor

2.6.3 LÓGICA DAS PREPOSIÇÕES E DOS PREDICADOS

Na lógica das preposições, será atribuído o valor lógico verdadeiro se as informações disponíveis permitirem tirar esta conclusão a respeito de uma preposição, caso contrário é atribuído o valor falso. Para se trabalhar com várias proposições utiliza-se operadores de conexão para assim obter as chamadas proposições compostas e aumentar a capacidade de expressão. Estes operadores são: AND, OR, NOT, IMPLIES, EQUIVALENT (Heinzle, 1995).

O cálculo dos predicados é uma extensão da lógica proposicional. A unidade de elementos da lógica dos predicados é um objeto, onde as afirmações sobre o objeto chamam-se predicados. Exemplo: A "Bola" é vermelha, é uma afirmação assertativa que diz que a bola é vermelha. Essa assertativa é verdadeira ou falsa (Harmon, 1988). Para representar apropriadamente o conhecimento do mundo real com algum formalismo, deve-se poder expressar não somente proposições verdadeiras ou falsas, mas também expressar ou descrever

objetos e generalizações sobre classes de objetos. A lógica dos predicados satisfaz estes objetivos (Pereira, 2000).

2.6.4 REGRAS DE PRODUÇÃO

O termo “regras de produção” é usado para descrever uma família de sistemas, que tem em comum o fato de serem constituídos de um conjunto de regras, que reúnem condições e ações. A condição é constituída por um padrão que determina a aplicabilidade da regra, enquanto a ação indica o que será realizado quando a regra for aplicada.

Um sistema de produção poderá ser formado por uma ou mais bases de regras, separadas segundo as conveniências de processamento. Complementa ainda, o sistema de produção, uma estratégia de controle estabelecendo as prioridades em que as regras serão aplicadas, bem como critérios de desempate quando houver mais regras candidatas á aplicação a um tempo só (Rabuske, 1995).

Sua estrutura constitui-se basicamente de uma premissa, ou conjunto de premissas, e uma conclusão, ou conjunto de conclusões. As regras são armazenadas como uma coleção de declarações SE-ENTÃO (SE <premissas> ENTÃO <conclusões). Onde a parte condicional consiste de uma expressão proposicional ou simplesmente um termo (Alexandre, 2000).

Além da naturalidade para a interpretação humana a utilização de regras de produção apresenta outros aspectos positivos, como a modularidade e uniformidade. As regras podem ser manipuladas como peças independentes e novas regras podem ser incluídas a qualquer tempo, o que é uma característica importante, pois o conhecimento de qualquer SE tende a aumentar com o passar do tempo. A uniformidade fica caracterizada como padrão único utilizado para todas as regras do sistema (Heinzle, 1995).

3 A FERRAMENTA JESS

Jess, denominada, *Java Expert System Shell*, é uma ferramenta para a construção de sistemas especialistas, que trabalha com uma linguagem de regras declarativas. Possui um ambiente de desenvolvimento interativo e para processar os fatos e regras utiliza o algoritmo *Rete*, que é um mecanismo para resolver o problema de “muitos-para-muitos” acertos.

A ferramenta Jess foi desenvolvida por Ernest Friedman-Hill na Sandia National Laboratories em Livermore, Canadá, como parte de um projeto de pesquisa. A sua primeira versão foi terminada no final do ano de 1995. Jess foi originalmente inspirada na *shell* CLIPS, e cresceu com forte influência da linguagem Java (Herzberg, 2002).

O software CLIPS (*C Language Integrated Production System*) foi desenvolvido pela NASA/Johnson Space Center, e tinha como finalidade gerar soluções que apresentassem alta portabilidade, baixo custo e fácil integração com sistemas externos. Problemas que afetavam os projetos da NASA em função da diversidade de aplicações e tecnologias utilizadas (Riley, 2002).

A linguagem utilizada no Jess é compatível com o CLIPS, sendo possível utilizar *scripts* feitos para CLIPS e Jess. Tal qual o CLIPS, o Jess usa o algoritmo *Rete* para processar as regras. Além disso, foram adicionadas algumas capacidades como *backwards chaining* (encadeamento para trás) e a habilidade para manipular e raciocinar diretamente sobre objetos Java. Jess permite criar e chamar métodos Java sem compilar código Java (Herzberg, 2002).

3.1 A LINGUAGEM JESS

Nesta seção serão demonstradas as principais características da linguagem Jess, conforme Herzberg (2002).

3.1.1 ÁTOMOS

O átomo é a essência do conceito da linguagem Jess, e são como identificadores em outras linguagens. Um átomo Jess pode conter letras, números e a seguinte pontuação: $*=+/<>_?#. . Um átomo não deve começar com um número.

Átomos em Jess são sensíveis ao caso, ou seja, difere as letras minúsculas das maiúsculas. Por exemplo: processador, Processador e PROCESSADOR são átomos diferentes.

Existem três átomos que são interpretados de forma especial no Jess: *nil*, o qual é similar ao *null* em Java; e os átomos *TRUE* e *FALSE*, que são valores *booleanos*.

3.1.2 NÚMEROS

O Jess usa as funções Java java.lang.Double.parseInt e java.lang.Integer.parseDouble para analisar números inteiros e ponto flutuante, respectivamente. Os seguintes números são válidos: 3, 4., 5.643, 6.0E4 e 1D.

3.1.3 STRINGS

Strings em Jess são denotadas usando aspas duplas (“). O caracter barra invertida (\), pode ser usado para inserir o caracter aspas (“) dentro do texto, para que o Jess não interprete como o final da *string*. Por exemplo, o comando: (printout t “Processador: \“Athlon 1000\”” crlf), irá mostrar na tela:

Processador: "Athlon 1000"

Para inserir uma nova linha no texto que será mostrado, basta criar uma nova linha no corpo da string. Exemplo:

(printout t “Processador

Athlon 1000\”” crlf).

3.1.4 LISTAS

A lista é uma das unidades fundamentais na sintaxe do Jess. A lista consiste em um conjunto de parênteses e zero ou mais átomos, números, *strings* ou outras listas. O primeiro elemento da lista é chamado de cabeça de lista. As seguintes listas são válidas: (+ 3 2), (a b c), ("Hello, World"), (), (deftemplate foo (slot bar)).

3.1.5 COMENTÁRIOS

O comentário nos programas Jess começa com o ponto e vírgula (;), e se estende até o final da linha. Abaixo temos um exemplo de comentário:

```
; Esta é uma lista

(a b c)
```

3.1.6 FUNÇÕES

Como em LISP, todo o código em Jess, como estruturas de controle, atribuições e chamadas de *procedures*, tomam a forma de uma chamada de função.

Chamadas de função em Jess são simplesmente listas, que usam uma notação de prefixo. Uma lista cuja cabeça é um átomo como o nome de uma função existente, pode ser a chamada função. Por exemplo, uma expressão que usa a função (+) para somar os números 2 e 3, seria escrita (+ 2 3). Quando interpretado, o valor desta expressão será o número 5, e não uma lista que contém o único elemento 5.

Jess possui funções embutidas para a execução de cálculos matemáticos, manipulação de strings, controle de programas, fornecendo acesso as *APIs* do Java. Uma das funções mais utilizadas é o *printout*, que serve para enviar o texto para a saída padrão do Jess, ou para um arquivo. Exemplo:

```
Jess> (printout t "Processador: P4 1.5 GHz" crlf))

Processador: P4 1.5 GHz
```

Outra função bastante usual é a *batch*, que serve para carregar um arquivo de código Jess. A função *batch* é chamada da seguinte forma: (batch regras.txt).

3.1.7 VARIÁVEIS

As variáveis em Jess são átomos que começam com o caracter ponto de interrogação (?), sendo que este é parte do nome da variável. Uma variável normal pode se referir a um único átomo, número ou string. Uma variável cujo primeiro caracter é o cifrão $, como por

exemplo, $?X, é uma variável multivalorada, que pode se referir a um tipo especial de lista, que permite mais de um campo. A variável pode ser definida através da função *bind*:

Jess> (bind ?x "Processador")

"Processador"

Estas listas com vários campos são geralmente criadas utilizando funções especiais como *create$*, e depois podem ser atribuídas a uma multivariável:

Jess> (bind $?computador (create$ Processador “Placa mãe” “Placa de vídeo”))

(Processador “Placa mãe” “Placa de vídeo”)

3.1.7.1 VARIÁVEIS GLOBAIS

As variáveis criadas pelo comando *bind*, são variáveis temporárias, que serão excluídas quando o comando *reset* for acionado, e não podem ser definidas como variáveis globais em sua essência. A criação de variáveis globais é feita pelo comando *defglobal*, sendo que estas não são excluídas pelo comando *reset*. Os nomes das variáveis globais devem começar e terminar com um asterisco, como por exemplo, ?*placa_mae*, e deve ser criada de acordo com a seguinte estrutura:

(defglobal [?<global-name> = <value>]+)

Quando a variável global é criada, é inicializada para o valor determinado. Esta variável pode ter o seu conteúdo alterado, mas ao ser executado comando *reset*, o seu valor voltará a ser o que foi definido na sua criação, como no exemplo abaixo:

Quadro 1: EXEMPLO DE VARIÁVEIS GLOBAIS

```
Jess> (defglobal ?*Processador* = “Athlon 1000”)
TRUE
Jess> (bind ?*Processador* “P4 1.5 GHz”)
 “P4 1.5 GHz”
Jess> ?*Processador*
“P4 1.5 GHz”
Jess> (reset)
TRUE
Jess> ?*Processador*
“Athlon 1000”
```

3.1.8 DEFFUNCTION

Está é uma das funções principais do Jess já que ela permite definir funções próprias. Um função *deffunction* pode ser construída de acordo com a seguinte estrutura:

(deffunction <function-name> [<doc-comment>] (<parameter>*)

<expr>*

[<return-specifier>])

O nome da função deve ser um átomo e cada parâmetro deve ser um nome de uma variável. Podem ser definidas várias expressões no bloco da função. O controle de fluxo pode ser feito por funções de controle, como o *foreach*, *if*, e *while*. Como exemplo, abaixo é demonstrado uma *deffunction* que devolve o maior de dois números passados como parâmetros:

```
Jess> (deffunction maior (?a ?b)
   (if (> ?a ?b) then
      (return ?a)
   else
      (return ?b)))
TRUE
```

Esta função pode agora ser chamada da seguinte forma:

```
Jess> (printout t "O maior número entre 3 e 5 é o " (maior 3 5) "." crlf)
O maior número entre 3 e 5 é o 5.
```

3.2 BASE DE CONHECIMENTOS

Conforme descrito na seção 2.4.1, um sistema baseado em regras de produção como o Jess, contém uma série de definições chamadas de fatos, que compõem a sua base de conhecimentos. É um pouco parecido com um banco de dados relacional, especialmente porque os fatos tem que ter uma estrutura específica. Em Jess, existem os fatos ordenados e fatos não ordenados (Herzberg, 2002).

3.2.1 FATOS ORDENADOS

Fatos ordenados são simplesmente listas onde o primeiro campo, a cabeça da lista, descreve uma categoria para o fato.

(computador “Athlon 1000” “ASUS A7VL133-VM”)

(pessoa "João da Silva" Masculino 21)

(pai-de Maria Pedro)

Os fatos ordenados podem ser acrescentados à base de conhecimento com a função *assert*:

(assert (pai-de Maria Pedro))

3.2.2 FATOS NÃO ORDENADOS

Os fatos ordenados não são estruturados, e em linguagens orientadas a objeto, os campos nos quais os dados serão inseridos, são identificados por um nome. Estes campos são chamados de *slots*.

(computador (Processador “Athlon 1000”) (Placa-mae “ASUS A7VL133-VM”))

(pessoa (nome " João da Silva") (idade 21) (sexo Masculino))

(automovel (montadora Ford) (modelo Focus) (ano 2001))

Antes dos fatos serem inferidos, devem ser definidos os *slots* que serão utilizados, utilizando a função *deftemplate*:

(deftemplate <deftemplate-name> [extends <classname>] [<doc-comment>]

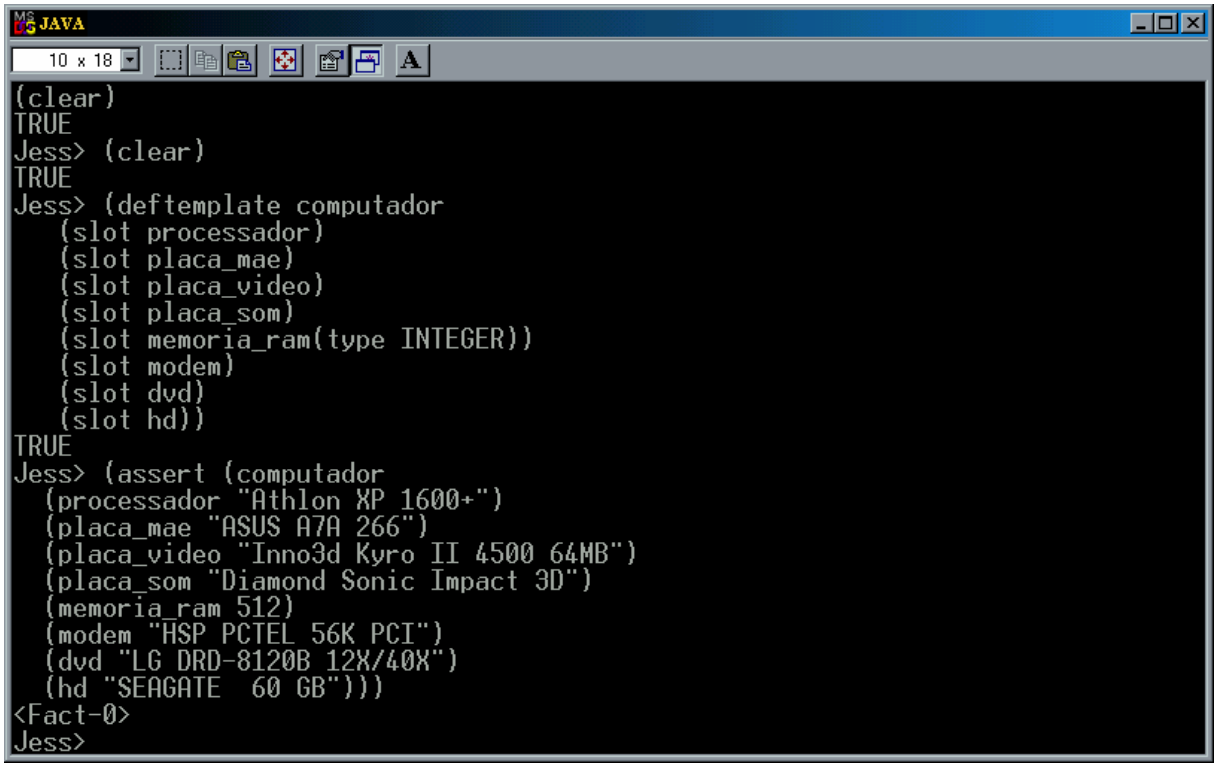
[(slot <slot-name> [(default | default-dynamic <value>)]

[(type <typespec>))]*)

A definição <deftemplate-name> é a cabeça dos fatos que serão criados usando este modelo. Pode existir um grande número de *slots*, onde cada <slot-name> deve ser um átomo. O valor padrão de um *slot* em um fato novo é determinado pelo campo identificado como <value>, e tem como valor padrão o átomo *nil*. A opção 'defaul-dynamic' atribuirá o valor determinado a cada vez que um novo fato que utiliza este modelo for afirmado. Valores aceitáveis são: ANY, INTEGER, FLOAT, NUMBER, ATOM, STRING, LEXEME, and OBJECT.

Na figura 4 é demonstrado a definição do *deftemplate* computador, e a aferição de um fato.

Figura 4 - EXEMPLO DE CRIAÇÃO DE UM *DEFTEMPLATE*



Este é um exemplo de definição de um *deftemplate* automovel:

```
Jess> (deftemplate automovel
(slot montadora)
(slot modelo)
(slot ano (type INTEGER))
(slot cor (default branco)))
```

Para inserir um fato na base, utiliza-se então o comando assert:

```
Jess> (assert (automovel (montadora Fiat) (modelo Palio) (ano 1997)))
```

O carro tem o atributo cor setado com o valor branco como padrão. Se não for definido um valor padrão para o *slot*, e não for informado um valor para este *slot* quando se for afirmar o fato, será atribuído o valor *nil*.

Um *slot* pode receber somente um valor, mas pode ser definido um *slots* multi-valorado, utilizando a palavra *multislot*. Pode-se utilizar também a cláusula *extends*, que permite que você defina um novo modelo baseado em um já existente. Por exemplo, pode-se definir um modelo carro_usado com um tipo de automovel, herdando os slots já definidos em automovel, e adicionando outros, como no exemplo abaixo:

```
Jess> (deftemplate carro_usado extends automovel
(slot quilometragem)
(slot valor_de_mercado)
(multislot donos))
```

3.2.3 DEFFACTS

A construção *deffact* é simplesmente uma lista nomeada de fatos, que permite inserir na base vários fatos em uma mesma construção, ao invés de inserir os fatos um a um

utilizando o comando *assert*. Após definir os fatos com o *deffact*, estes serão afirmados na base de conhecimentos assim que o comando *reset* for executado:

```
Jess> (deffacts Fatos

(Jogos_pesados nao)

(Ferramentas_graficas sim)

(Custo_beneficio alto))

Jess> (reset)

TRUE
```

3.2.4 DECLARAÇÃO DE REGRAS

Conforme descrito na seção 2.6.4, uma regra, é algo parecido com a declaração “se...então” em uma linguagem procedural, mas não é utilizado da mesma forma no Jess. Enquanto em uma linguagem procedural a declaração “se...então” é executada em um momento específico, de acordo com ordem em que está no programa, uma regra é executada sempre que a parte condicional, chamada de *left-hand-side* (LHS), for satisfeita.

Uma regra em Jess é declarada pela construção *defrule*. Para declarar-se uma regra que irá imprimir na tela o processador “Athlon 1000” se as condição de jogos pesados for igual a sim e custo benefício for igual a baixo, pode ser feita da seguinte forma:

```
(defrule imprime_athlon1000

(Jogos_pesados sim)

(Custo_beneficio baixo)

=>

(printout t "Processador: Athlon 1000" crlf))
```

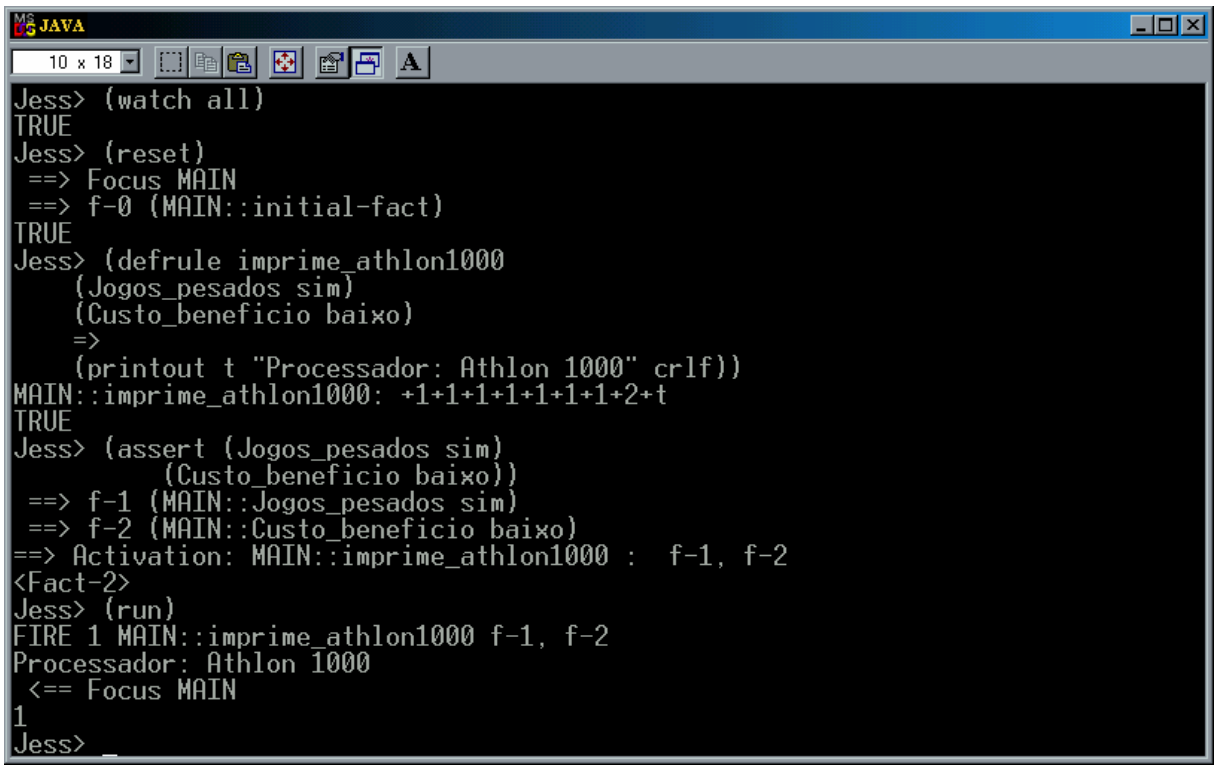
Esta regra tem duas partes, separadas pelo símbolo “=>”, que pode ser lido como “então”. A primeira parte consiste na condição que deve ser satisfeita, LHS, que irá procurar combinar os fatos com a base de conhecimentos, e a segunda parte é a ação a ser tomada se a primeira parte for satisfeita, através de chamadas de função.

Neste exemplo, a regra será ativada assim que o fato for encontrado na base de conhecimentos, disparando a função *printout* para imprimir na tela “Processador: Athlon 1000". O fato pode ser aferido na base da seguinte forma:

(assert (Jogos_pesados sim) (Custo_beneficio baixo))

Para demonstrar como esta regra pode ser disparada, apresenta-se na figura 5 um programa que irá imprimir o processador na tela assim que os fatos forem encontrados na base. Utiliza-se a função *watch all* que irá mostrar na tela algumas informações importantes enquanto o programa for digitado.

Figura 5 - EXEMPLO DE UM PROGRAMA JESS



O primeiro comando é o *reset*, que afirma na base o fato inicial (*initial-fact*). O comando *reset* deve ser utilizado sempre que se trabalhar com regras. Quando a regra é declarada, vemos a linha “+1+1+1+1+1+1+1+2+t”, que mostra como a regra é interpretada

internamente pelo algoritmo *Rete*. Quando os fatos “Jogos_pesados sim” e “Custo_beneficio baixo” foram afirmados na base, é mostrado que a regra imprime_athlon1000 foi ativada: “Activation: MAIN::imprime_athlon1000 : f-1, f-2”. Isto significa que a regra imprime_athlon1000 tem todas as suas condições conhecidas pela lista de fatos “f-1, f-2”.

Depois dos fatos terem sido encontrados na base, a regra imprime_athlon1000 ainda não foi disparada, foi somente ativada, isto porque o motor de inferência não está rodando, e para fazer o motor executar é utilizado o comando *run*. Assim que o comando *run* é digitado, as regras ativadas são disparadas. Como o comando *watch all* Jess nos notifica que a regra imprime_athlon1000 foi disparada da seguinte forma: “FIRE 1 MAIN::imprime_athlon1000 f-1, f-2”. Então é impresso na tela a ação definida na regra: “Processador: Athlon 1000”. Ao final aparece o número 1, que é o retorno do comando *run* informando quantas regras foram disparadas.

Se o comando *run* for executado novamente, nenhuma regra será disparada pois a regra só é disparada uma vez para um dado conjunto de fatos. Para poder executar a regra novamente, deve ser executado o comando *reset* e os fatos devem ser novamente afirmados, para então executar o comando *run*.

As regras são identificadas pelo seu nome, e se for definida uma nova regra imprime_athlon1000, esta irá sobrepor a anterior, mesmo que a anterior tivesse sido disparada.

4 ESCOLHER UM COMPUTADOR

O computador tem alguns componentes que podem ser destacados, como o processador, que é responsável por receber dados, processá-los e devolvê-los. A placa-mãe, que é a principal placa do computador, pois contém os principais componentes do micro como o processador, memória RAM e o HD, e é onde podem ser conectadas placas periféricas como a placa de vídeo, a placa de som e o modem.

Outros componentes importantes são a placa de vídeo, responsável pela geração das imagens, a memória RAM (*Random Access Memory*), que armazena os dados para o processador, e o HD (*Hard Disk*), que é o disco onde são armazenados todos os programas e dados processados.

Antes de escolher um computador deve-se definir a finalidade para qual ele será utilizado, como por exemplo para rodar jogos que exijam alto desempenho, ferramentas de editoração gráfica que exijam alto poder de processamento matemático como o AutoCAD e 3Dstudio, se irá acessar a uma rede local, se irá acessar a internet, e também qual a relação custo/benefício procurada.

O problema mais comum encontrado na compra de um computador está relacionado a falta de conhecimento encontrado em vendedores da área, principalmente a não existência de uma orientação adequada em relação ao produto que irá oferecer a melhor relação custo/benefício ao usuário. Por exemplo, muitas pessoas querem comprar um poderoso micro baseado no processador Pentium III ou Pentium IV, só porque é o "melhor do mercado", mas não porque irá atender melhor às suas necessidades. Outro problema comum é em relação à baixa qualidade de peças utilizadas, que em busca de um preço menor utiliza-se material de baixa qualidade (Torres, 2002).

Como mencionado no início do capítulo, o componente principal do computador é o processador, tanto que geralmente o micro inteiro é chamado de "Pentium IV", "Pentium III", "Athlon", enquanto, na verdade, esses nomes referem-se somente ao processador da máquina. Entretanto, o micro é formado por outros componentes que influem na qualidade,

durabilidade e desempenho do micro, como a placa-mãe, a placa de vídeo e a memória RAM (Torres, 2002).

Segundo Torres (2002), muitas empresas optam por montar micros com vídeo *on-board*, isto é, integrado na placa-mãe. O vídeo *on-board* é mais rápido que placas de vídeo como a Trident, porém mais lentas que placas de vídeo preparadas para um alto desempenho, como as ATI Radeon ou ASUS Geforce. Outro problema na utilização do vídeo *on-board* é que os desempenhos de processamento e disco caem, pois será o próprio processador da máquina que deverá executar as funções de vídeo, e não uma placa em separado. A única vantagem do vídeo *on-board* é o preço final. Este tipo de placa-mãe normalmente utiliza *chipset* não-Intel.

De uma forma geral, pode-se dizer que para micros que pretende-se alto desempenho, não se deve utilizar placa de vídeo *on-board*. Já para o usuário comum, que não está preocupado com o desempenho e que irá utilizar o micro em tarefas simples, como processamento de textos e acesso à Internet, o uso de uma placa-mãe com vídeo integrado traz o menor custo/benefício.

De acordo com Torres (2002), deve-se escolher o processador que apresente a melhor relação custo/benefício, de acordo com a sua principal finalidade. Utilizar um micro baseado no Pentium IV para processamento de textos, e outras tarefas simples, não é vantajoso. Da mesma forma que, comprar um micro barato e com poucos recursos para a utilização de ferramentas pesadas como Corel Draw!, PageMaker ou jogos 3D, não é aconselhável.

Em geral, os processadores não Intel, como o K6-2 e o K6-III possuem baixo desempenho matemático, e não são os mais recomendados para aplicações que exijam alto poder de processamento matemático. Os processadores da Cyrix, MII e MediaGX são os que apresentam desempenho mais baixo, mas, em contrapartida, são mais baratos, sendo indicados, portanto, para micros de baixo custo para aplicações que não necessitem de tanto poder de processamento. Pode-se dizer que os processadores Intel são os que atingem desempenho mais elevado, porém são mais caros (Torres, 2002).

Segundo Torres (2002), outro componente importante na compra de um computador, é a placa mãe. Existem diversas boas marcas no mercado como: Abit, A-Trend, Asus, Biostar,

FIC, Soyo, QDI, MSI e ECS. Uma boa maneira de saber um pouco mais sobre a marca da placa-mãe é visitando o site do fabricante na Internet, que provém informações detalhadas sobre a placa, testes e prêmios que já ganhou. O principal componente da placa-mãe é o *chipset*, que é um conjunto de circuitos integrados existentes na placa-mãe. Os diversos fabricantes mais comuns são: Intel, ALi, SiS e Via. Para um processador Intel, deve-se comprar uma placa-mãe com chipset Intel, que oferecerá um melhor desempenho. Entretanto, para um processador não-Intel, deve-se optar por uma placa-mãe com chipset não-Intel, como por exemplo: ALi, SiS ou Via.

Para Torres (2002), outro ponto importante na escolha do micro é a memória RAM. O ideal é de no mínimo 128 MB, porém, quanto mais memória, melhor será o desempenho para executar aplicativos pesados ou trabalhar com vários aplicativos ao mesmo tempo. Normalmente a memória RAM é do tipo SDRAM (*Synchonous Dynamic RAM*).

Uma das peças que mais influencia no preço final do micro é o monitor de vídeo. Existem modelos dos mais diversos valores e, em geral, a qualidade do monitor é diretamente proporcional ao seu preço. Os monitores de vídeo são classificados pelo tamanho do tubo de imagens, medido em polegadas, e os modelos mais comuns são os de 14", 15", 17" e 21" polegadas (Torres, 2002).

Segundo Torres (2002), o kit multimídia é formado basicamente por dois componentes: uma unidade de CD-ROM e uma placa de som. Em relação a unidade de CD-ROM, deve-se saber que a taxa de transferência básica de uma unidade de CD-ROM é 150 KB/s. Dessa forma, uma unidade de 40x consegue transferir dados 40 vezes mais rapidamente que essa taxa, ou seja, 6.000 KB/s. As unidades acima de 12x utilizam velocidade angular constante, isto significa que a taxa de transferência informada é atingida somente na leitura dos dados localizados na borda do CD. A taxa de transferência da parte interna do CD é a metade dessa taxa.

Quanto as placas de som, as do tipo *on-board* não possuem nenhum tipo de recurso extra, especialmente o amplificador de áudio, e não apresentam uma boa relação sinal/ruído, mas para o usuário comum essas deficiências em geral não são percebidas. Entretanto, para usuários exigentes, o ideal é ter uma placa de som com mais recursos. As placas de som podem ser divididas em duas categorias: as que possuem síntese por FM e as que possuem

síntese por *wave table*. Em placas possuem síntese por FM, os sons gerados a partir de arquivos do tipo MIDI, como os de fundo de um jogo, são criados eletronicamente por um circuito chamado sintetizador de FM, gerando uma sonoridade "metalizada". Já nas placas com síntese por *wave table*, o fabricante grava os sons de instrumentos dentro de uma memória localizada na placa. Dessa forma, quando o som de um instrumento é executado, percebe-se que o som é de um instrumento real e não um som criado por computador (Torres, 2002).

As considerações feitas neste capítulo, quanto aos pontos relevantes que devem ser analisados na escolha de um computador, servirão de base para a formalização do conhecimento por parte do especialista.

5 DESENVOLVIMENTO DO TRABALHO

Conforme apresentado no capítulo 2, sobre a estrutura de um sistema especialista, pode-se dizer que uma das etapas mais importantes e mais difíceis é a da aquisição do conhecimento.

Neste capitulo será enfatizado a construção da base de conhecimentos.

5.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Este trabalho apresenta um protótipo de um sistema especialista que tem como finalidade auxiliar na escolha de um computador para uso pessoal, baseado em estudos feito na área de venda de microcomputadores.

São feitas perguntas ao usuário através de um questionário, como forma de apoio a decisão na escolha de um computador. As respostas são tratadas pelas regras declaradas na base de conhecimentos, e ao final é mostrado a configuração de um computador que melhor retrata o perfil do usuário.

As perguntas são feitas ao usuário com base no seguinte questionário:

1. Você vai utilizar o computador para rodar jogos que exijam alto desempenho?

2. Você vai utilizar ferramentas de editoração gráfica como Corel Draw, Photoshop, ou ferramentas que exijam alto poder de processamento matemático como o AutoCAD ou 3Dstudio?

3. Você irá armazenar no computador arquivos muito grandes, como vídeos ou imagens de alta definição?

4. Você vai utilizar o computador para acessar a internet?

5. Você irá acessar a uma rede local?

6. Você irá querer gravar CDs no microcomputador?

7. Você quer um drive de CD Rom?

8. Você irá assistir a filmes em DVD no computador?

9. Como você define a relação custo/benefício que você procura, entre baixo, médio e alto?

5.2 ESPECIFICAÇÃO

Para formalizar o conhecimento adquirido no estudo de venda de microcomputadores, utilizou-se de regras de produção. Abaixo são mostradas algumas delas:

Regra 1

SE Jogos pesados = SIM

E Custo benefício = BAIXO

ENTÃO Processador = "Athlon 1000"

Regra 2

SE Jogos pesados = SIM

E Custo benefício = MEDIO

ENTÃO Processador = "P4 1.5 GHz"

Regra 3

SE Jogos pesados = SIM

E Custo benefício = ALTO

ENTÃO Processador = "P4 2.0 GHz"

Regra 4

SE Jogos pesados = NAO

E Ferramentas gráficas = SIM

E Custo benefício = BAIXO

ENTÃO Processador = "Athlon 1000"

O conjunto completo com as regras definidas para a especificação do protótipo estão presentes no Apêndice 1.

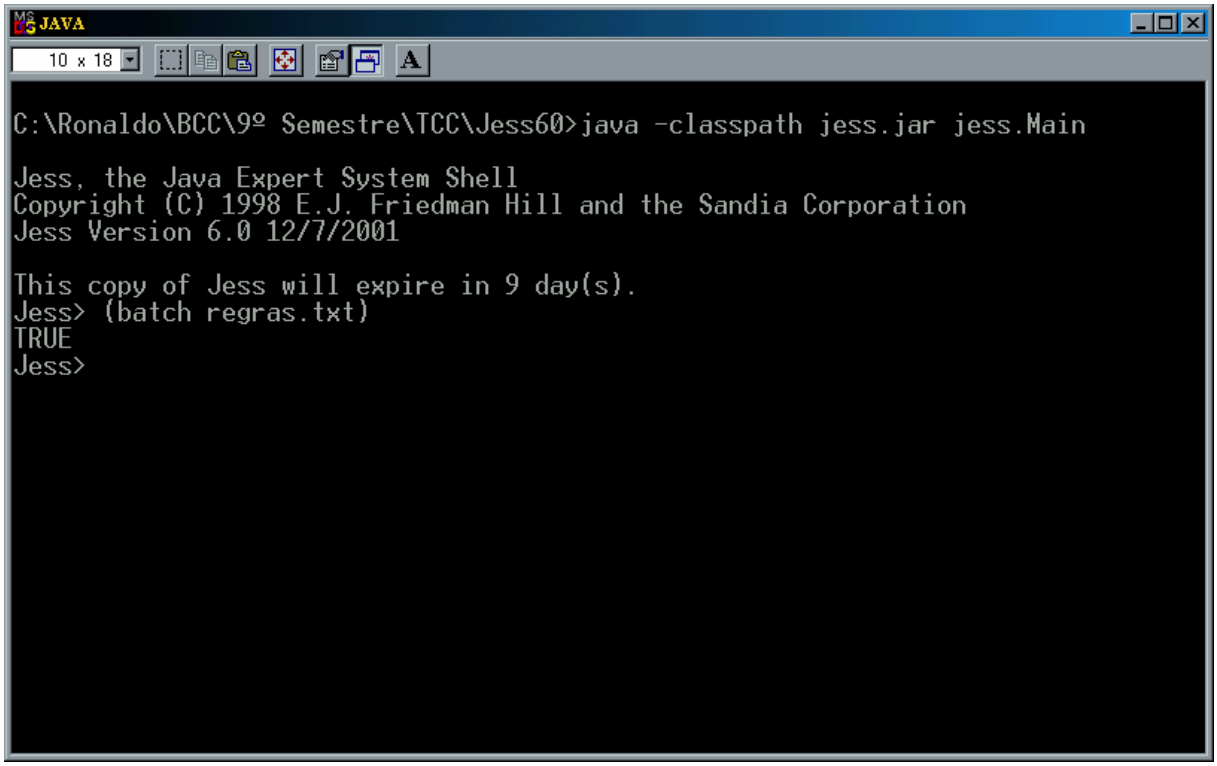
5.3 IMPLEMENTAÇÃO

A implementação do protótipo foi feita utilizando a ferramenta Jess. Para tanto, as regras foram definidas em um arquivo texto chamado “regras.txt”, que é carregado no ambiente do Jess.

5.3.1 FUNCIONALIDADE DO PROTÓTIPO

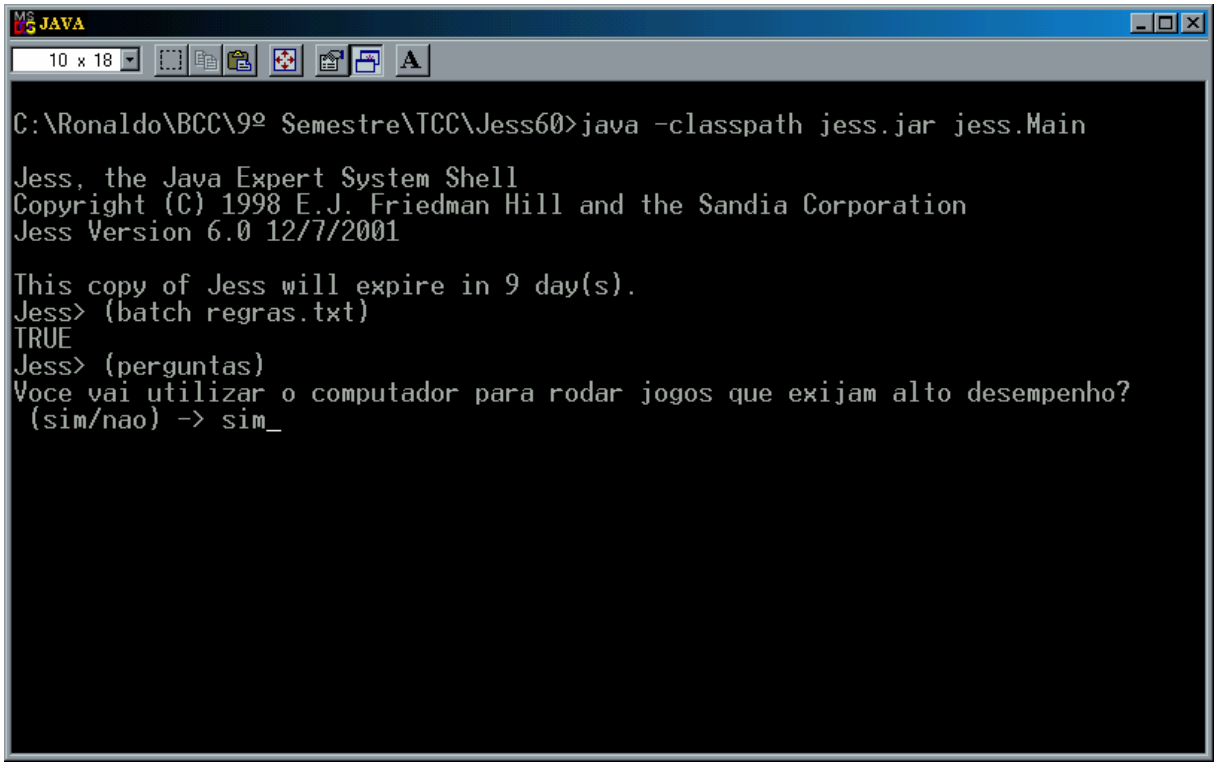
Para executar o protótipo é necessário entrar no *prompt* do Jess, e carregar o arquivo “regras.txt” com o comando *batch*. Ao carregar o arquivo, as regras são validadas pelo interpretador do Jess, que retorna “TRUE” informando que todas as regras foram adicionadas na base com sucesso, conforme pode ser verificado na figura 6:

Figura 6 - CARGA DO ARQUIVO “REGRAS.TXT”



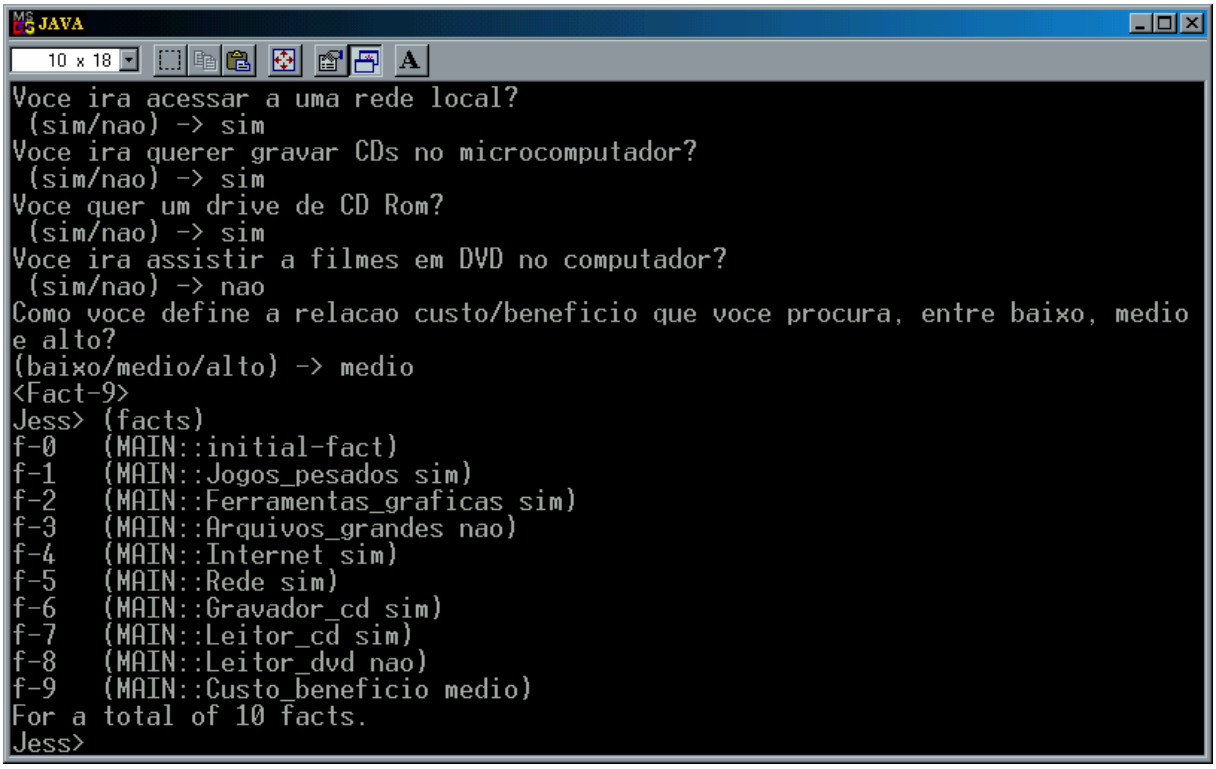
Dentre as regras declaradas, foi definida uma função chamada “perguntas”, que contém o questionário que deve ser feito ao usuário. A chamada da função é feita pelo comando “(perguntas)” no *prompt* do Jess. Ao chamar a função perguntas, o questionário irá aparecer na tela para que o usuário responda de acordo com as opções. Na figura 7 é demonstrada a primeira pergunta que é feita ao usuário:

Figura 7 – PERGUNTAS AO USUÁRIO



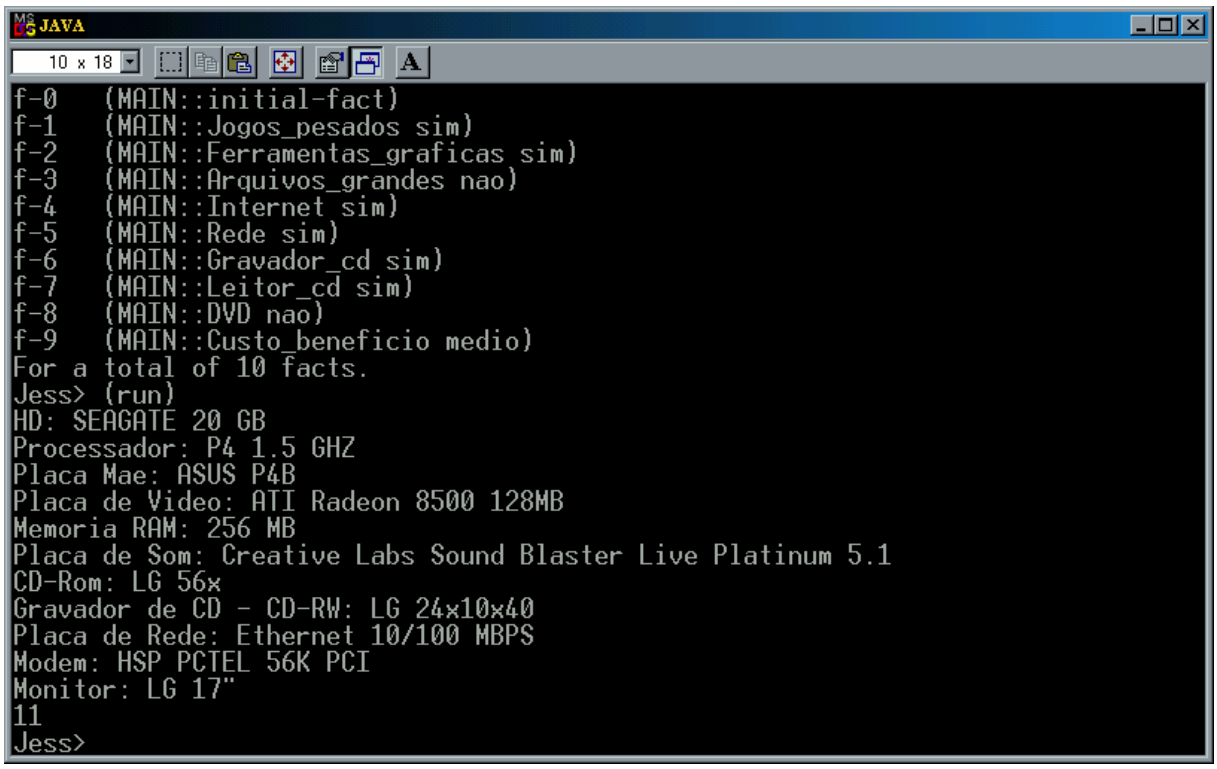
Ao responder as perguntas, as respostas são afirmadas na base, e as regras que forem satisfeitas são ativadas. Pode-se verificar os fatos que foram afirmados na base, através do comando *facts*, como pode ser observado na figura 8.

Figura 8 – VERIFICAÇÃO DOS FATOS AFIRMADOS



Até o momento, os fatos foram afirmados, mas ainda não houve retorno nenhum sobre as características do computador, isto porque o motor de inferência não está ativo, e para ativá-lo utiliza-se o comando *run*.

Figura 9 – RESULTADO NA TELA



Após a execução do comando *run*, a configuração sugerida de acordo com as respostas, é impressa na tela, conforme é demonstrado na figura 9.

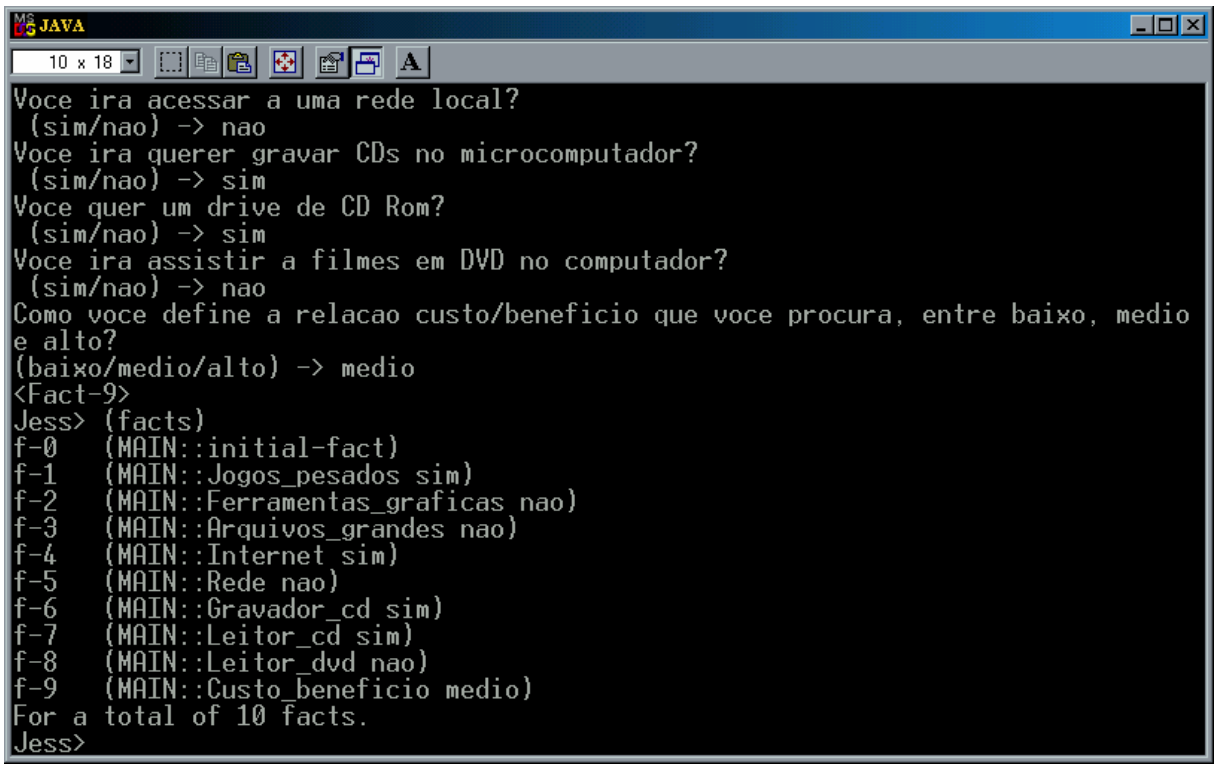
Para executar novamente o protótipo, deve-se primeiro digitar o comando *reset*, e depois chamar a função perguntas com o comando “(perguntas)” no *prompt* do Jess. Após isto, deve-se executar novamente o comando *run* para que o resultado seja impresso na tela.

Para um melhor entendimento da funcionalidade do protótipo, será criada uma situação hipotética de um cliente típico, em busca de um computador.

Defini-se então que o comprador quer utilizar o microcomputador para entretenimento, em atividades como rodar jogos pesados, acessar a internet e gravar CDs de músicas, porém, procura uma relação custo/benefício média. O usuário não tem interesse em utilizar ferramentas de editoração gráfica, gravar arquivos muito grandes e, também não pretende acessar a uma rede local ou assistir a filmes em DVD no micro.

O comprador irá então responder o questionário do sistema, conforme apresentado na seção 5.1. Ao responder o questionário, os fatos são afirmados na base. Na figura 10 demonstra-se os fatos que foram afirmados na base de acordo com as repostas.

Figura 10 – FATOS QUE FORAM AFIRMADOS NA BASE

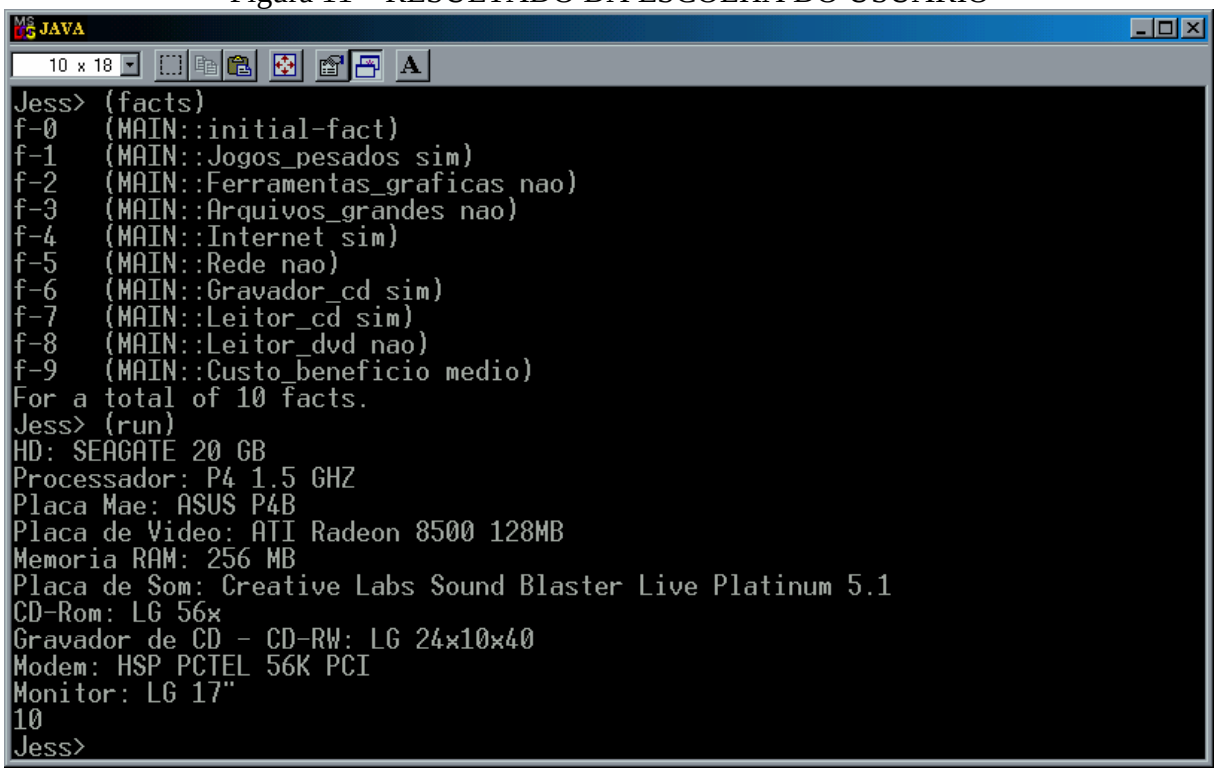


Com base nestes fatos as regras serão avaliadas pelo sistema. Neste momento, quando uma regra é satisfeita, esta é ativada no Jess.

Ao se afirmar Jogos_pesados sim e Custo_beneficio medio, serão ativadas as regras 2 e 39. Com a afirmação de Arquivos_grandes nao, a regra 46 será ativada. A afirmação de Internet sim ativa a regra 48, a afirmação de Gravador_cd sim ativa a regra 50 e a afirmação de Leitor_cd sim ativa a regra 51.

Como dito anteriormente, as regras foram somente ativadas, e para que as regras sejam executadas, utiliza-se o comando *run*. Quando o comando *run* é executado, as regras ativadas são disparadas, e o resultado é mostrado na figura 11.

Figura 11 – RESULTADO DA ESCOLHA DO USUÁRIO

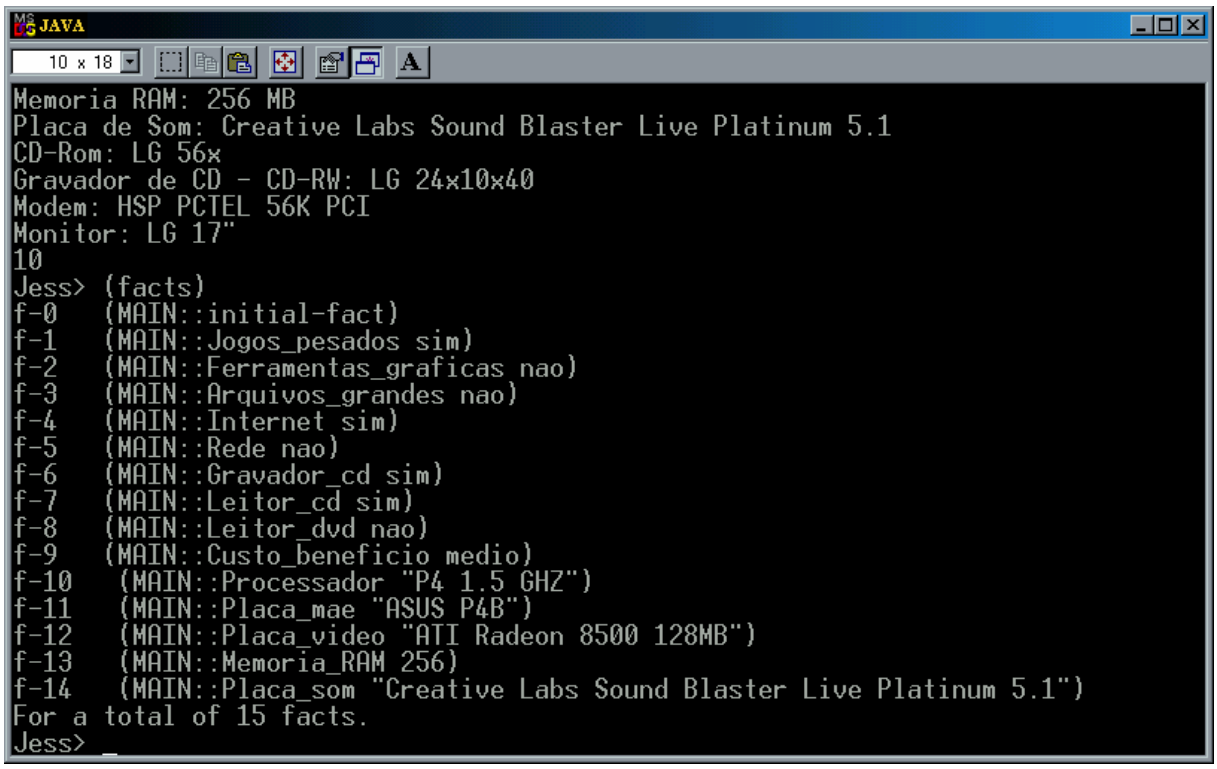


Ao ser disparada a regra 2, que estava ativada, é afirmado Processador "P4 1.5 GHz". Esta afirmação dispara a regra 11, afirmando Placa_mae "ASUS P4B". Com a afirmação de Processador e de Placa_mae, é ativada a regra 18, que afirma Placa_video "ATI Radeon 8500 128MB". Esta afirmação dispara a regra 30 definindo Memoria_RAM 256 MB. A afirmação de placa_mae ainda dispara a regra 25 definindo a placa de som como "Creative Labs Sound Blaster Live Platinum 5.1".

A execução da regra 39 define o monitor como LG 17", e a regra 46 define que o HD será "SEAGATE 20 GB". As regras 48, 50 e 51, definem respectivamente o modem como “HSP PCTEL 56K PCI", a mídia gravador de CD Rom como “LG 24x10x40” e a mídia leitor de CD Rom como “LG 56x”.

Depois de serem validadas e executadas as regras, pode-se verificar os fatos que foram afirmados utilizando o comando *facts*, como é mostrado na figura 12.

Figura 12 – FATOS QUE FORAM AFIRMADOS



6 CONCLUSÕES

O presente trabalho abordou os aspectos relacionados aos sistemas especialistas, como a sua estrutura e as formas de representação do conhecimento, e permitiu o estudo da ferramenta para desenvolvimento de sistemas especialistas Jess.

Considera-se que este trabalho atingiu o seu principal objetivo, o de desenvolver um protótipo de um SE para auxiliar na escolha de um computador para uso pessoal utilizando a ferramenta Jess.

Foram declaradas regras para formalizar o conhecimento e auxiliar na tomada de decisão. A técnica de regras de produção mostrou-se adequada, pois permitiu representar o conhecimento de forma clara e simples, e pôde ser bem aplicada a ferramenta Jess no desenvolvimento do protótipo.

O sistema especialista, no domínio do conhecimento para qual foi construído, obteve resultados satisfatórios para os objetivos pretendidos. Isto pôde ser demonstrado com a exemplificação de uma situação hipotética para a escolha de um computador, onde se obteve um resultado de acordo com a necessidade do usuário, auxiliando-o na tomada de decisão.

A ferramenta Jess mostrou-se eficiente sendo utilizada para um sistema baseado em regras de produção, pois permitiu a construção de base de conhecimentos de forma intuitiva, facilitando o trabalho do especialista, e se comportou da forma esperada na interpretação das regras.

6.1 LIMITAÇÕES

Este protótipo está limitado a base de conhecimentos criada com base em um conhecimento específico. Como na área de microcomputadores, as inovações tecnológicas são crescentes, rapidamente novos modelos de processadores, placas-mãe, placas de vídeo e outros componentes são lançados no mercado, isto exigirá que a base de conhecimentos seja atualizada constantemente.

Outra limitação deste protótipo, é quanto a interface, que não é muito amigável para o usuário, e exige uma digitação correta na entrada dos dados, uma vez que esta não é tratada. Na entrada dos dados, deve ser informado somente uma das palavras solicitadas, como “sim” ou “nao”. Ao se digitar uma opção errônea, a última opção será considerada.

6.2 EXTENSÕES

A utilização da ferramenta Jess neste trabalho abre precedentes para trabalhos e estudos futuros, que podem aprofundar-se em uma das principais características da ferramenta, que é a resolução de problemas do tipo “muitos-para-muitos”.

Ainda é válido utilizar a ferramenta Jess na construção de sistemas que agreguem métodos Java, como *applets*, permitindo utilizar o sistema na internet, e possibilitando o desenvolvimento de uma interface mais amigável.

APÊNDICE 1

Regra 1

SE Jogos pesados = SIM

E Custo benefício = BAIXO

ENTÃO Processador = "Athlon 1000"

Regra 2

SE Jogos pesados = SIM

E Custo benefício = MEDIO

ENTÃO Processador = "P4 1.5 GHz"

Regra 3

SE Jogos pesados = SIM

E Custo benefício = ALTO

ENTÃO Processador = "P4 2.0 GHz"

Regra 4

SE Jogos pesados = NAO

E Ferramentas gráficas = SIM

E Custo benefício = BAIXO

ENTÃO Processador = "Athlon 1000"

Regra 5

SE Jogos pesados = NAO

E Ferramentas gráficas = SIM

E Custo benefício = MEDIO

ENTÃO Processador = "Athlon XP 1600+"

Regra 6

SE Jogos pesados = NAO

E Ferramentas gráficas = SIM

E Custo benefício = ALTO

ENTÃO Processador = "Athlon XP 2000+"

Regra 7

SE Jogos pesados = NAO

E Ferramentas gráficas = NAO

E Custo benefício = BAIXO

ENTÃO Processador = "Duron 950 MHz"

Regra 8

SE Jogos pesados = NAO

E Ferramentas gráficas = NAO

E Custo benefício = MEDIO

ENTÃO Processador = "Celeron 1 GHz"

Regra 9

SE Jogos pesados = NAO

E Ferramentas gráficas = NAO

E Custo benefício = ALTO

ENTÃO Processador = "P III 1.0 GHz"

Regra 10

SE Processador = "Athlon 1000"

E Jogos pesados = SIM

ENTAO Placa mãe = "ASUS A7VL133-VM"

Regra 11

SE Jogos pesados = SIM

E (Processador = "P4 1.5 GHz" OU Processador = "P4 2.0 GHz")

ENTAO Placa mãe = "ASUS P4B"

Regra 12

SE Processador = "Athlon 1000"

E Jogos pesados = NAO

E Ferramentas gráficas = SIM

ENTÃO Placa Mãe = "PC CHIPS 810"

Regra 13

SE Jogos pesados = NAO

E Ferramentas gráficas = SIM

E (Processador = "Athlon XP 1600+" OU Processador = "Athlon XP 2000+")

ENTÃO Placa Mãe = "ASUS A7A 266"

Regra 14

SE Processador = "Duron 950 MHz"

ENTÃO Placa Mãe = "BIOSTAR M7VKQ"

Regra 15

SE Processador = "Celeron 1 GHz"

ENTÃO Placa Mãe = "PC CHIPS 756 LMRT"

Regra 16

SE Processador = "P III 1.0 GHz"

ENTÃO Placa Mãe = "BIOSTAR M6VLQ"

Regra 17

SE Placa Mãe = "ASUS A7VL133-VM"

ENTÃO Placa de Vídeo = "ASUS V7700 Ti Pure GeForce2 64MB"

Regra 18

SE Placa Mãe = "ASUS P4B"

E Processador = "P4 1.5 GHz"

ENTÃO Placa de Vídeo = "ATI Radeon 8500 128MB"

Regra 19

SE Placa Mãe = "ASUS P4B"

E Processador = "P4 2.0 GHz"

ENTÃO Placa de Vídeo = "ASUS V8460 Geforce4 Ti4600"

Regra 20

SE Placa Mãe = "PC CHIPS 810"

ENTÃO Placa de Vídeo = "Visiontek Xtasy 5632 GeForce2 GTS-V 32MB"

Regra 21

SE Placa Mãe = "ASUS A7A 266"

E Processador = "Athlon XP 1600+"

ENTÃO Placa de Vídeo = "Inno3d Kyro II 4500 64MB"

Regra 22

SE Placa Mãe = "ASUS A7A 266"

E Processador = "Athlon XP 2000+"

ENTÃO Placa de Vídeo = "ASUS V8170 DDR GeForce 4 MX440"

Regra 23

SE Placa Mãe = "BIOSTAR M7VKQ"

OU Placa Mãe = "PC CHIPS 756 LMRT"

OU Placa Mãe = "BIOSTAR M6VLQ"

ENTÃO Placa de Vídeo = "on-board"

Regra 24

SE Placa de Vídeo = "ASUS V7700 Ti Pure GeForce2 64MB"

ENTÃO Placa de Som = "Diamond monster Sound MX400"

Regra 25

SE Placa Mãe = "ASUS P4B"

ENTÃO Placa de Som = "Creative Labs Sound Blaster Live Platinum 5.1"

Regra 26

SE Placa Mãe = "PC CHIPS 810"

ENTÃO Placa de Som = "on-board"

Regra 27

SE Placa Mãe = "ASUS A7A 266"

ENTÃO Placa de Som = "Diamond Sonic Impact 3D"

Regra 28

SE Placa de Vídeo = "on-board"

ENTÃO Placa de Som = "on-board"

Regra 29

SE Placa de Vídeo = "ASUS V7700 Ti Pure GeForce2 64MB"

ENTÃO Memória RAM = 128

Regra 30

SE Placa de Vídeo = "ATI Radeon 8500 128MB"

ENTÃO Memória RAM = 256

Regra 31

SE Placa de Vídeo = "ASUS V8460 Geforce4 Ti4600"

ENTÃO Memória RAM = 512

Regra 32

SE Placa de Vídeo = "Visiontek Xtasy 5632 GeForce2 GTS-V 32MB"

ENTÃO Memória RAM = 128

Regra 33

SE Placa de Vídeo = "Inno3d Kyro II 4500 64MB"

ENTÃO Memória RAM = 256

Regra 34

SE Placa de Vídeo = "ASUS V8170 DDR GeForce 4 MX440"

ENTÃO Memória RAM = 512

Regra 35

SE Placa de Vídeo = "on-board"

E Placa Mãe = "BIOSTAR M7VKQ"

ENTÃO Memória RAM = 64

Regra 36

SE Placa de Vídeo = "on-board"

E Placa Mãe = "PC CHIPS 756 LMRT"

ENTÃO Memória RAM = 128

Regra 37

SE Placa de Vídeo = "on-board"

E Placa Mãe = "BIOSTAR M6VLQ"

ENTÃO Memória RAM = 256

Regra 38

SE Custo Benefício = BAIXO

ENTÃO Monitor = "LG 15”"

Regra 39

SE Custo Benefício != BAIXO

E Jogos pesados = SIM

ENTÃO Monitor = "LG 17”"

Regra 40

SE Custo Benefício != BAIXO

E Jogos pesados = NAO

E Ferramentas Gráficas = SIM

ENTÃO Monitor = "LG 17”"

Regra 41

SE Custo Benefício != BAIXO

SE Jogos pesados = NAO

E Ferramentas Gráficas = NAO

ENTÃO Monitor = "LG 15”"

Regra 42

SE Arquivos grandes = SIM

E Custo Benefício = BAIXO

ENTÃO HD = "SEAGATE 40 GB"

Regra 43

SE Arquivos grandes = SIM

E Custo Benefício = MEDIO

ENTÃO HD = "SEAGATE 60 GB"

Regra 44

SE Arquivos grandes = SIM

E Custo Benefício = ALTO

ENTÃO HD = "SEAGATE 80 GB"

Regra 45

SE Arquivos grandes = NAO

E Custo Benefício = BAIXO

ENTÃO HD = "SEAGATE 10 GB"

Regra 46

SE Arquivos grandes = NAO

E Custo Benefício = MEDIO

ENTÃO HD = "SEAGATE 20 GB"

Regra 47

SE Arquivos grandes = NAO

E Custo Benefício = ALTO

ENTÃO HD = "SEAGATE 40 GB"

Regra 48

SE Internet = SIM

ENTAO Modem = "HSP PCTEL 56K PCI"

Regra 49

SE Rede = SIM

ENTAO Placa de rede = "Ethernet 10/100 MBPS"

Regra 50

SE Gravador de CD = SIM

ENTAO CD-RW = "LG 24x10x40"

Regra 51

SE Leitor de CD = SIM

ENTAO CD-Rom = "LG 56x"

Regra 52

SE Leitor DVD Rom = SIM

ENTAO DVD_rom = "LG 12x/40x"

APÊNDICE 2

Conteúdo do arquivo regras.txt:

```
(clear)
(reset)

(defrule regra-1
  (Jogos_pesados sim)
  (Custo_beneficio baixo)
  =>
  (assert (Processador "Athlon 1000"))
  (printout t "Processador: Athlon 1000" crlf))

(defrule regra-2
  (Jogos_pesados sim)
  (Custo_beneficio medio)
  =>
  (assert (Processador "P4 1.5 GHz"))
  (printout t "Processador: P4 1.5 GHz" crlf))

(defrule regra-3
  (Jogos_pesados sim)
  (Custo_beneficio alto)
  =>
  (assert (Processador "P4 2.0 GHz"))
  (printout t "Processador: P4 2.0 GHz" crlf))

(defrule regra-4
  (Jogos_pesados nao)
  (Ferramentas_graficas sim)
  (Custo_beneficio baixo)
  =>
  (assert (Processador "Athlon 1000"))
  (printout t "Processador: Athlon 1000" crlf))

(defrule regra-5
  (Jogos_pesados nao)
  (Ferramentas_graficas sim)
  (Custo_beneficio medio)
  =>
  (assert (Processador "Athlon XP 1600+"))
  (printout t "Processador: Athlon XP 1600+" crlf))

(defrule regra-6
  (Jogos_pesados nao)
```

```
  (Ferramentas_graficas sim)
  (Custo_beneficio alto)
  =>
  (assert (Processador "Athlon XP 2000+"))
  (printout t "Processador: Athlon XP 2000+" crlf))

(defrule regra-7
  (Jogos_pesados nao)
  (Ferramentas_graficas nao)
  (Custo_beneficio baixo)
  =>
  (assert (Processador "Duron 950 MHz"))
  (printout t "Processador: Duron 950 MHz" crlf))

(defrule regra-8
  (Jogos_pesados nao)
  (Ferramentas_graficas nao)
  (Custo_beneficio medio)
  =>
  (assert (Processador "Celeron 1 GHz"))
  (printout t "Processador: Celeron 1 GHz" crlf))

(defrule regra-9
  (Jogos_pesados nao)
  (Ferramentas_graficas nao)
  (Custo_beneficio alto)
  =>
  (assert (Processador "P III 1.0 GHz"))
  (printout t "Processador: P III 1.0 GHz" crlf))

(defrule regra-10
  (Processador "Athlon 1000")
  (Jogos_pesados sim)
  =>
  (assert (Placa_mae "ASUS A7VL133-VM"))
  (printout t "Placa Mae: ASUS A7VL133-VM" crlf))

(defrule regra-11
  (Jogos_pesados sim)
  (or (Processador "P4 1.5 GHz") (Processador "P4 2.0 GHz"))
  =>
  (assert (Placa_mae "ASUS P4B"))
  (printout t "Placa Mae: ASUS P4B" crlf))

(defrule regra-12
  (Processador "Athlon 1000")
  (Jogos_pesados nao)
  (Ferramentas_graficas sim)
```

```
  =>
  (assert (Placa_mae "PC CHIPS 810"))
  (printout t "Placa Mae: PC CHIPS 810" crlf))

(defrule regra-13
  (Jogos_pesados nao)
  (Ferramentas_graficas sim)
  (or (Processador "Athlon XP 1600+") (Processador "Athlon XP 2000+"))
=>
  (assert (Placa_mae "ASUS A7A 266"))
  (printout t "Placa Mae: ASUS A7A 266" crlf))

(defrule regra-14
  (Processador "Duron 950 MHz")
  =>
  (assert (Placa_mae "BIOSTAR M7VKQ"))
  (printout t "Placa Mae: BIOSTAR M7VKQ" crlf))

(defrule regra-15
  (Processador "Celeron 1 GHz")
  =>
  (assert (Placa_mae "PC CHIPS 756 LMRT"))
  (printout t "Placa Mae: PC CHIPS 756 LMRT" crlf))

(defrule regra-16
  (Processador "P III 1.0 GHz")
  =>
  (assert (Placa_mae "BIOSTAR M6VLQ"))
  (printout t "Placa Mae: BIOSTAR M6VLQ" crlf))

(defrule regra-17
  (Placa_mae "ASUS A7VL133-VM")
  =>
  (assert (Placa_video "ASUS V7700 Ti Pure GeForce2 64MB"))
  (printout t "Placa de Video: ASUS V7700 Ti Pure GeForce2 64MB" crlf))

(defrule regra-18
  (Placa_mae "ASUS P4B")
  (Processador "P4 1.5 GHz")
  =>
  (assert (Placa_video "ATI Radeon 8500 128MB"))
  (printout t "Placa de Video: ATI Radeon 8500 128MB" crlf))

(defrule regra-19
  (Placa_mae "ASUS P4B")
  (Processador "P4 2.0 GHz")
  =>
  (assert (Placa_video "ASUS V8460 Geforce4 Ti4600"))
```

```
  (printout t "Placa de Video: ASUS V8460 Geforce4 Ti4600" crlf))

(defrule regra-20
  (Placa_mae "PC CHIPS 810")
  =>
  (assert (Placa_video "Visiontek Xtasy 5632 GeForce2 GTS-V 32MB"))
  (printout t "Placa de Video: Visiontek Xtasy 5632 GeForce2 GTS-V 32MB" crlf))

(defrule regra-21
  (Placa_mae "ASUS A7A 266")
  (Processador "Athlon XP 1600+")
  =>
  (assert (Placa_video "Inno3d Kyro II 4500 64MB"))
  (printout t "Placa de Video: Inno3d Kyro II 4500 64MB" crlf))

(defrule regra-22
  (Placa_mae "ASUS A7A 266")
  (Processador "Athlon XP 2000+")
  =>
  (assert (Placa_video "ASUS V8170 DDR GeForce 4 MX440"))
  (printout t "Placa de Video: ASUS V8170 DDR GeForce 4 MX440" crlf))

(defrule regra-23
  (or (Placa_mae "BIOSTAR M7VKQ")
      (Placa_mae "PC CHIPS 756 LMRT")
      (Placa_mae "BIOSTAR M6VLQ"))
  =>
  (assert (Placa_video "on-board"))
  (printout t "Video: on-board" crlf))

(defrule regra-24
  (Placa_video "ASUS V7700 Ti Pure GeForce2 64MB")
  =>
  (assert (Placa_som "Diamond monster Sound MX400"))
  (printout t "Placa de Som: Diamond monster Sound MX400" crlf))

(defrule regra-25
  (Placa_mae "ASUS P4B")
  =>
  (assert (Placa_som "Creative Labs Sound Blaster Live Platinum 5.1"))
  (printout t "Placa de Som: Creative Labs Sound Blaster Live Platinum 5.1" crlf))

(defrule regra-26
  (Placa_mae "PC CHIPS 810")
  =>
  (assert (Placa_som "on-board"))
  (printout t "Placa de Som: on-board" crlf))
```

```
(defrule regra-27
  (Placa_mae "ASUS A7A 266")
  =>
  (assert (Placa_som "Diamond Sonic Impact 3D"))
  (printout t "Placa de Som: Diamond Sonic Impact 3D" crlf))

(defrule regra-28
  (Placa_video "on-board")
  =>
  (assert (Placa_som "on-board"))
  (printout t "Placa de Som: on-board" crlf))

(defrule regra-29
  (Placa_video "ASUS V7700 Ti Pure GeForce2 64MB")
  =>
  (assert (Memoria_RAM 128))
  (printout t "Memoria RAM: 128 MB" crlf))

(defrule regra-30
  (Placa_video "ATI Radeon 8500 128MB")
  =>
  (assert (Memoria_RAM 256))
  (printout t "Memoria RAM: 256 MB" crlf))

(defrule regra-31
  (Placa_video "ASUS V8460 Geforce4 Ti4600")
  =>
  (assert (Memoria_RAM 512))
  (printout t "Memoria RAM: 512 MB" crlf))

(defrule regra-32
  (Placa_video "Visiontek Xtasy 5632 GeForce2 GTS-V 32MB")
  =>
  (assert (Memoria_RAM 128))
  (printout t "Memoria RAM: 128 MB" crlf))

(defrule regra-33
  (Placa_video "Inno3d Kyro II 4500 64MB")
  =>
  (assert (Memoria_RAM 256))
  (printout t "Memoria RAM: 256 MB" crlf))

(defrule regra-34
  (Placa_video "ASUS V8170 DDR GeForce 4 MX440")
  =>
  (assert (Memoria_RAM 512))
  (printout t "Memoria RAM: 512 MB" crlf))
```

```
(defrule regra-35
  (Placa_video "on-board")
  (Placa_mae "BIOSTAR M7VKQ")
  =>
  (assert (Memoria_RAM 64))
  (printout t "Memoria RAM: 64 MB" crlf))

(defrule regra-36
  (Placa_video "on-board")
  (Placa_mae "PC CHIPS 756 LMRT")
  =>
  (assert (Memoria_RAM 128))
  (printout t "Memoria RAM: 128 MB" crlf))

(defrule regra-37
  (Placa_video "on-board")
  (Placa_mae "BIOSTAR M6VLQ")
  =>
  (assert (Memoria_RAM 256))
  (printout t "Memoria RAM: 256 MB" crlf))

(defrule regra-38
  (Custo_beneficio baixo)
  =>
  (printout t "Monitor: LG 15\"" crlf))

(defrule regra-39
  (not (Custo_beneficio baixo))
  (Jogos_pesados sim)
  =>
  (printout t "Monitor: LG 17\"" crlf))

(defrule regra-40
  (not (Custo_beneficio baixo))
  (Jogos_pesados nao)
  (Ferramentas_graficas sim)
  =>
  (printout t "Monitor: LG 17\"" crlf))

(defrule regra-41
  (not (Custo_beneficio baixo))
  (Jogos_pesados nao)
  (Ferramentas_graficas nao)
  =>
  (printout t "Monitor: LG 15\"" crlf))

(defrule regra-42
  (Arquivos_grandes sim)
```

```
  (Custo_beneficio baixo)
  =>
  (printout t "HD: SEAGATE 40 GB" crlf))

(defrule regra-43
  (Arquivos_grandes sim)
  (Custo_beneficio medio)
  =>
  (printout t "HD: SEAGATE 60 GB" crlf))

(defrule regra-44
  (Arquivos_grandes sim)
  (Custo_beneficio alto)
  =>
  (printout t "HD: SEAGATE 80 GB" crlf))

(defrule regra-45
  (Arquivos_grandes nao)
  (Custo_beneficio baixo)
  =>
  (printout t "HD: SEAGATE 10 GB" crlf))

(defrule regra-46
  (Arquivos_grandes nao)
  (Custo_beneficio medio)
  =>
  (printout t "HD: SEAGATE 20 GB" crlf))

(defrule regra-47
  (Arquivos_grandes nao)
  (Custo_beneficio alto)
  =>
  (printout t "HD: SEAGATE 40 GB" crlf))

(defrule regra-48
  (Internet sim)
  =>
  (printout t "Modem: HSP PCTEL 56K PCI" crlf))

(defrule regra-49
  (Rede sim)
  =>
  (printout t "Placa de Rede: Ethernet 10/100 MBPS" crlf))

(defrule regra-50
  (Gravador_cd sim)
  =>
  (printout t "Gravador de CD - CD-RW: LG 24x10x40" crlf))
```

```
(defrule regra-51
  (Leitor_cd sim)
  =>
  (printout t "CD-Rom: LG 56x" crlf))

(defrule regra-52
  (Leitor_dvd sim)
  =>
  (printout t "DVD-Rom: LG 12x/40x" crlf))


(deffunction perguntas ()
  (printout t "Voce vai utilizar o computador para rodar jogos que exijam alto desempenho?
 (sim/nao) -> ")
  (if (eq (read) sim) then
    (assert (Jogos_pesados sim))
   else
    (assert (Jogos_pesados nao))
  )
  (printout t "Voce vai utilizar ferramentas de editoracao grafica como Corel Draw,
Photoshop, ou ferramentas que exijam alto poder de processamento matematico como o
AutoCAD ou 3Dstudio?
 (sim/nao) -> ")
  (if (eq (read) sim) then
    (assert (Ferramentas_graficas sim))
   else
    (assert (Ferramentas_graficas nao))
  )
  (printout t "Voce ira armazenar no computador arquivos muito grandes, como videos ou
imagens de alta definicao?
 (sim/nao) -> ")
  (if (eq (read) sim) then
    (assert (Arquivos_grandes sim))
   else
    (assert (Arquivos_grandes nao))
  )
  (printout t "Voce vai utilizar o computador para acessar a internet?
 (sim/nao) -> ")
  (if (eq (read) sim) then
    (assert (Internet sim))
   else
    (assert (Internet nao))
  )
  (printout t "Voce ira acessar a uma rede local?
 (sim/nao) -> ")
  (if (eq (read) sim) then
    (assert (Rede sim))
```

```
   else
    (assert (Rede nao))
  )
  (printout t "Voce ira querer gravar CDs no microcomputador?
 (sim/nao) -> ")
  (if (eq (read) sim) then
    (assert (Gravador_cd sim))
   else
    (assert (Gravador_cd nao))
  )
  (printout t "Voce quer um drive de CD Rom?
 (sim/nao) -> ")
  (if (eq (read) sim) then
    (assert (Leitor_cd sim))
   else
    (assert (Leitor_cd nao))
  )
  (printout t "Voce ira assistir a filmes em DVD no computador?
 (sim/nao) -> ")
  (if (eq (read) sim) then
    (assert (Leitor_dvd sim))
   else
    (assert (Leitor_dvd nao))
  )
  (printout t "Como voce define a relacao custo/beneficio que voce procura, entre baixo,
medio e alto?
(baixo/medio/alto) -> ")
  (bind ?custo (read))
  (if (eq ?custo baixo) then
    (assert (Custo_beneficio baixo))
   else
    (if (eq ?custo medio) then
      (assert (Custo_beneficio medio))
     else
      (assert (Custo_beneficio alto))
    )
  )
)
```

REFERÊNCIAS BIBLIOGRÁFICAS

ALEXANDRE, Adriana Bombassaro. **Protótipo de um sistema especialista utilizando a ferramenta expert sinta shell para auxílio no setor de suporte de uma *software house***. 2000. 82 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

CHAIBEN, Hamilton. **Inteligência Artificial na Educação**. Curitiba, 1999. Disponível em: <http://www.cce.ufpr.br/~hamilton/iaed/iaed.htm>. Acesso em: 15 abr. 2002.

HARMON, Paul; KING, David. **Sistemas especialistas**. Tradução Antonio Fernandes Carpinteiro. Rio de Janeiro: Campus, 1988. 304 p.

HEINZLE, Roberto. **Protótipo de uma ferramenta para criação de sistemas especialistas baseados em regras de produção**. 1995. 145 f. Tese (Mestrado em Engenharia de Produção) – Departamento de Engenharia de Produção e Sistemas, Universidade Federal de Santa Catarina, Florianópolis.

HERZBERG, Gerhard. **Jess, the Expert System Shell for the Java Platform**, Livermore, 2002. Disponível em: <http://herzberg.ca.sandia.gov/jess/>. Acesso em: 06 abr. 2002.

LIA, Laboratório de Inteligência Artificial. **Expert SINTA: uma ferramenta para criação de sistemas especialistas**, Pernambuco, 1995. Disponível em: <http://www.lia.ufc.br/>. Acesso em: 13 abr. 2002.

LOPEZ, Oscar C. **Expert Systems in production engineering**, Florianópolis, 1997. Disponível em: <http://www.eps.ufsc.br/~oscar/exp_sys/ing/>. Acesso em: 20 mar. 2002.

LUCHTENBERG, Jonas. **Protótipo de sistema especialista para área comercial utilizando a ferramenta SPIRIT**. 2000. 50f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

PEREIRA, Marcelo Dezordi. **Protótipo de um sistema especialista difuso para a seleção de imóveis em imobiliária**. 2000. 70 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

RABUSKE, Renato Antônio. **Inteligência artificial**. Florianópolis: Editora da UFSC, 1995.

RIBEIRO, Horácio da Cunha e Souza. **Introdução aos sistemas especialistas**. Rio de Janeiro: LTC - Livros Técnicos e Científicos Editora S.A., 1987.

RILEY, Gary. **CLIPS: A tool for Building Expert Systems**. Houston, 2002. Disponível em: <http://www.ghg.net/clips/CLIPS.html>. Acesso em: 02 maio 2002.

TORRES, Gabriel. **Clube do Hardware**, Rio de Janeiro, 2002. Disponível em: <http://www.clubedohardware.com.br>. Acesso em: 11 maio 2002.

SOUZA, Aline Rassweiller de. **Comparativo de ferramentas para sistemas especialistas**. 2001a. 89 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SOUZA, Marila de. **Representação do conhecimento**. Blumenau, 2001b. 6 f., color. Material disponibilizado pela professora da disciplina de Inteligência Artificial do curso de Ciências da Computação, da Universidade Regional de Blumenau, ministrada no 1º semestre do ano de 2001.

WATERMAN, D. A. **A Guide of Expert Systems**. USA: Addison – Weslwy Publishing Company Inc, 1986.

WEISS, Sholow M.; KULIKOWSKI, Casamir A.. **Guia prático para projetar sistemas especialistas**. Rio de Janeiro: LTC – Livros Técnicos e Científicos Editora S.A., 1988.