



INTELIGÊNCIA ARTIFICIAL APLICADA

AULA 4



Prof. Luciano Frontino de Medeiros



CONVERSA INICIAL

Nesta aula você estudará duas técnicas que fazem parte da linha de pesquisa simbólica, os Sistemas Especialistas e a Programação em Lógica. São abordados elementos comuns às duas técnicas, o uso de fatos e regras para representar o conhecimento, bem como exemplos práticos que auxiliarão na compreensão dos aspectos dinâmicos de cada técnica.

CONTEXTUALIZANDO

A linha de pesquisa simbólica da IA é uma das mais tradicionais, presentes desde as primeiras pesquisas, envolvendo a manipulação de símbolos para alcançar objetivos ao resolver algum problema específico. Dois dos representantes mais significativos desta linha são os sistemas especialistas e a programação em lógica. Os sistemas especialistas já eram pesquisados mesmo antes de haver computadores pessoais, evidenciando a possibilidade do conhecimento de profissionais especialistas ser representado em um sistema de computador de maneira a proporcionar informações, como se o sistema estivesse aconselhando a melhor alternativa para uma tomada de decisão, ou a descrição de um quadro sintomático. A programação em lógica permite a um agente raciocinar de acordo com a lógica de primeira ordem, permitindo a resolução de uma série de problemas da IA. Vamos agora ao estudo destas duas técnicas, representantes-chave da linha de pesquisa simbólica.

TEMA 1: SISTEMAS ESPECIALISTAS

Dentro da linha de pesquisa simbólica da IA, os Sistemas Especialistas (SE) são programas de computador que imitam o comportamento de especialistas humanos dentro de um domínio específico de conhecimento. Os sistemas especialistas podem ser classificados quanto às definições da IA no quadrante “agir como humanos”. Consiste assim numa ferramenta que possui

a capacidade de entender o conhecimento sobre um problema específico e usar esse conhecimento de maneira inteligente para sugerir alternativas de ação.

Podemos afirmar que SE é uma técnica da IA desenvolvida para **resolver problemas** em um determinado domínio, cujo conhecimento utilizado é obtido de pessoas que são especialistas naquele domínio.

No início, a ideia central dos sistemas especialistas foi originada do sistema DENDRAL, em 1965, desenvolvido a partir de uma grande soma de conhecimento heurístico manipulado por **regras**, procurando resolver o problema de se inferir estruturas moleculares a partir da informação espectrográfica de massa. A partir de 1968 até o presente, o DENDRAL foi utilizado em diversas pesquisas relacionadas à química orgânica, sendo alguns resultados derivados considerados melhores do que obtidos por especialistas humanos (BITTENCOURT, 1998, p. 276).

Desenvolvido por Edward Shortliffe, da Universidade de Stanford, na década de 1970, o MYCIN é um sistema especialista para a área médica, o qual teve um papel fundamental no desenvolvimento dos futuros sistemas especialistas. O MYCIN foi desenvolvido para resolver o problema do diagnóstico e tratamento de doenças infecciosas do sangue por meio de um conjunto de 450 regras que permitiam o diagnóstico e a prescrição de tratamentos. A utilidade deste tipo de sistema é que nem sempre um médico é também especialista na área de infecções, ainda mais considerando um ambiente hospitalar (BITTENCOURT, 1998, p. 274).

As regras do MYCIN eram codificadas na linguagem LISP. As regras eram formadas por premissas com combinações booleanas denominadas de **cláusulas**. Cada cláusula era composta de um predicado e uma tripla de parâmetros objeto-atributo-valor. Um exemplo de cláusula do MYCIN pode ser entendido da seguinte forma:



SE

- 1) *A infecção é do tipo bacteremia primária E*
- 2) *O local da cultura é um dos locais estéreis E*
- 3) *A entrada para o organismo foi pelo trato gastrointestinal*

ENTÃO

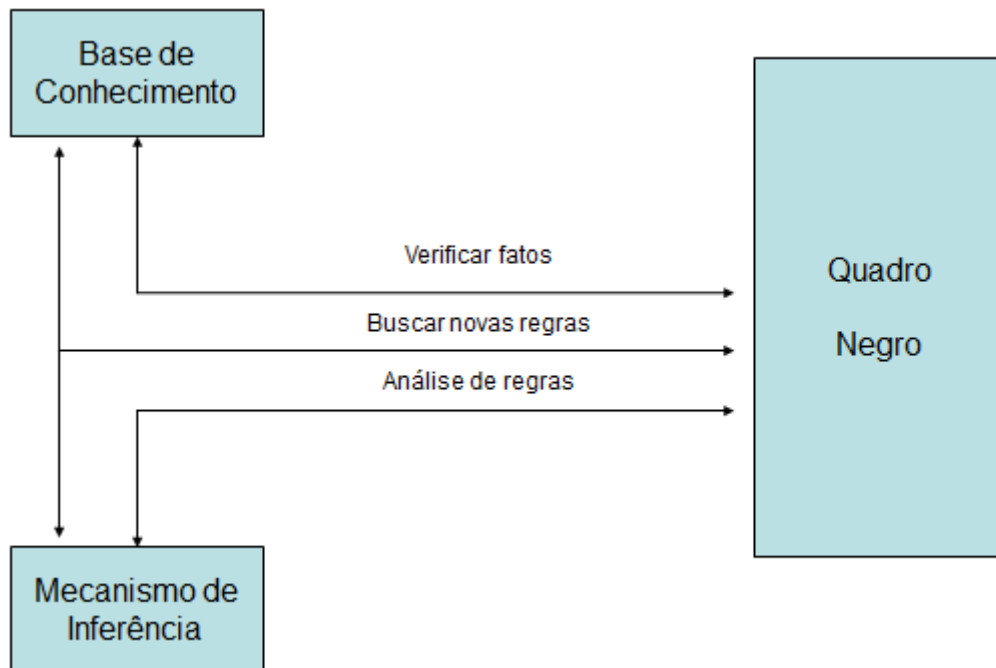
Existe evidência sugestiva (0.7) que a identidade do organismo seja um bacteroide.

As regras no MYCIN eram associadas a um coeficiente de certeza, variando na faixa de valores entre -1 e 1. Os coeficientes eram utilizados para propagar a incerteza inicial por meio do raciocínio do programa em uma cadeia de inferências.

TEMA 2: COMPONENTES DE UM SISTEMA ESPECIALISTA

De forma geral, um sistema especialista pode ser dividido em três módulos: uma base de conhecimento, um quadro negro e um mecanismo de inferência (também chamado de motor de inferência). A **base de conhecimento** é onde fica armazenado o conhecimento obtido do domínio no qual atua o SE, traduzido na forma de regras, e também uma memória de trabalho. No **quadro negro**, as informações são alimentadas ao sistema com relação às variáveis que são lidas do ambiente. O mecanismo de inferência é o responsável pelo encadeamento e teste das regras, fornecendo um resultado em função dos fatos alimentados ao sistema especialista.

Figura 1: Esquema básico de um sistema especialista



A base de conhecimento contém as condições expressas nas regras que se referem às perguntas. Estas perguntas envolvem variáveis que precisam ser instanciadas (receber valores) e passar logo após por um processo de inferência. O motor de inferência controla, portanto, a atividade do sistema especialista, ocorrendo em ciclos que se constituem de três fases:

- 1) A seleção das regras a partir dos dados correspondentes de entrada;
- 2) A resolução de conflitos indicando quais regras serão efetivamente executadas, a partir de priorização e ordenação;
- 3) A ação propriamente dita.



TEMA 3: ETAPAS PARA CONSTRUÇÃO DE UM SISTEMA ESPECIALISTA

O processo de construção de um sistema especialista costuma seguir os passos descritos abaixo:

- 1) Identificação e definição do domínio do problema;
- 2) Aquisição do Conhecimento;
- 3) Organização e Representação do Conhecimento;
- 4) Implementação do SE;
- 5) Testes e Validação.

Com relação à **identificação e definição do domínio do problema**, o desenvolvimento de um SE se inicia com a **especificação do domínio do problema**, que geralmente ocorre a partir da consulta a algumas fontes de conhecimento, tais como: livros, manuais, relatórios técnicos e principalmente da experiência e conhecimento dos especialistas. Dessa forma, são identificadas possíveis variáveis e possibilidade de divisão do problema a ser abordado em subproblemas.

Na **aquisição de conhecimento** é considerada a parte mais sensível no desenvolvimento de um SE, muitas vezes o **gargalo** do processo. Essa dificuldade está relacionada muitas vezes 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. A aquisição de conhecimento é realizada geralmente por meio de entrevistas e interações com especialistas do domínio. A aquisição de conhecimento consiste, assim, na coleta e análise de informações de um ou mais especialistas e qualquer outra fonte, possibilitando a produção de documentos.



Esses documentos formam a base de funcionamento da base de conhecimento do SE.

A organização do conhecimento de um SE é feita de fatos e regras. Um **fato** é uma forma de conhecimento declarativo. Fatos são usados para descrever relacionamentos entre estruturas de conhecimento mais complexas e controlar o uso destas estruturas durante a resolução de problemas. Em IA e SE, um fato é às vezes referenciado como uma **proposição**. Uma proposição é uma declaração que pode ser verdadeira ou falsa. Por exemplo:

“Está chovendo”

A frase é uma declaração de algo que acontece no mundo. Dependendo do que acontece no mundo, se está chovendo na realidade, a proposição acima recebe o valor “verdadeiro”. Caso esteja fazendo sol, então recebe o valor “falso”. Outro exemplo:

“A cor da bola é vermelha”

Neste caso, a proposição está organizada na forma **objeto-atributo-valor** (OAV). O objeto é “bola”, “cor” é o atributo e “vermelha” é o valor. Se olharmos uma bola azul e emitimos esta afirmação, o valor dela é “falso”. No caso de a cor ser vermelha, o valor da proposição será “verdadeiro”.

A representação do conhecimento em um SE utiliza, como visto anteriormente, regras. **Regras** são sequências lógicas compostas por **premissas** (antecedentes) e **conclusões** (consequentes). É comum o uso da expressão condicional SE-ENTÃO para representar regras dentro de um sistema especialista. Por exemplo

SE (*distância à frente < 10 cm*) **ENTÃO** (*vire 180 graus*)

SE (*temperatura > 50 graus*) **ENTÃO** (*desligue aquecedor*)

Estes exemplos de regras são ditas **determinísticas**, pois quando a premissa for verdadeira, sempre acontecerá a ação da conclusão da regra.



Regras podem ainda ter um coeficiente de confiança ou de probabilidade. Dessa forma, as regras se tornam probabilísticas, e a ação está condicionada a um fator que indica a probabilidade ou confiança de que a ação irá acontecer.

Regras probabilísticas são importantes para SEs que trabalham com informações incertas, que farão inferências sobre diagnósticos para indicar ações com mais “força” que outras.

Para a fase de **implementação**, é realizada aqui a escolha de uma linguagem de programação ou pacote (muitas vezes chamado de *shell*) pelo analista, na implementação do sistema. As linguagens utilizadas para codificar sistemas especialistas geralmente diferem das linguagens convencionais, tais como C ou Java, e tal escolha pode implicar no desenvolvimento do próprio mecanismo de inferência do SE. As *shells* apresentam vários benefícios no desenvolvimento de um SE, pois já implementam o motor de inferência. Um exemplo de *shell* é o Expert SINTA.

A última etapa na elaboração do SE são os **testes e validação** do sistema por meio de análise de casos conhecidos. Estes testes, geralmente também são realizados para a validação das regras na base de conhecimento.

TEMA 4: EXEMPLO DE SISTEMA ESPECIALISTA

Para ilustrar a operação de um sistema especialista, o exemplo a seguir refere-se a uma análise de perfil financeiro para concessão de crédito. Esse tipo de sistema pode ser entendido como uma regra de negócio de uma empresa financeira, podendo ser estendido para outros domínios, tais como companhias de seguro e planos de saúde.

O sistema especialista tem como objetivo a concessão de crédito que irá depender de uma série de variáveis que precisam ser informadas ao sistema. A modelagem do SE pode ser vista na Figura 2. A primeira regra se refere à comparação da variável *renda*. Caso seja maior que 5000, então o próximo passo é perguntar sobre o percentual da *prestação do carro*; caso seja menor

do que 10% da renda, então segue para a próxima regra, que é a comparação da *prestação da casa*; caso seja menor do que 20% da renda, então o crédito será concedido.

A partir daí o interesse agora é o de verificar qual o valor do crédito que será concedido. Para isso, uma regra compara o *tempo de trabalho*; caso seja maior ou igual a 4 anos, então o limite a ser concedido é de 10000. Se for menor ou igual a 4 anos, então é necessário comparar a variável *outras dívidas*; caso seja menor do que 5% da renda, então o limite concedido será um pouco menor, de 3000.

Figura 2: Diagrama de fluxo das regras do sistema especialista para análise de perfil de cliente visando a concessão de crédito

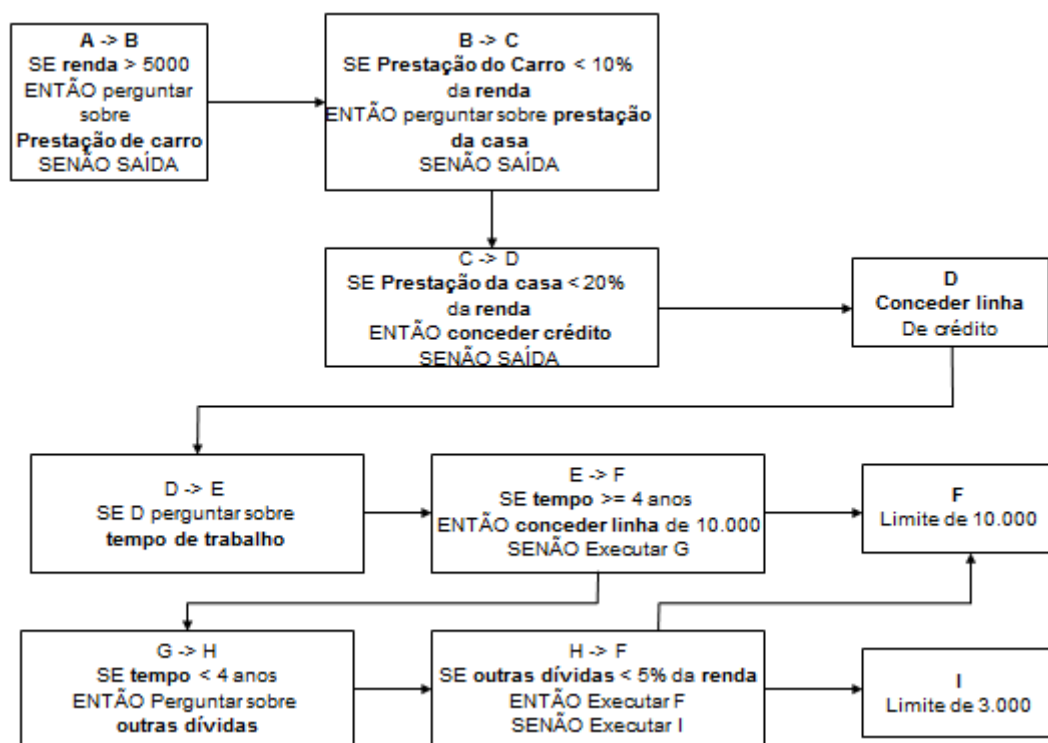
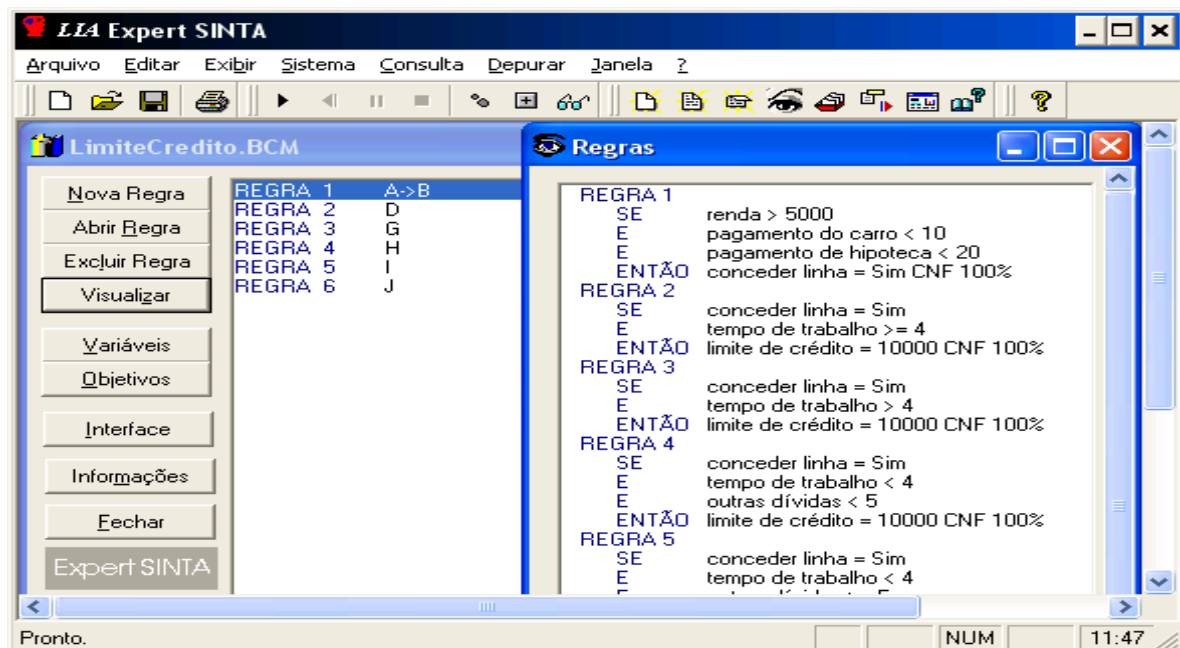


Figura 3: Tela demonstrando a implementação do sistema especialista de concessão de crédito no Expert SINTA



Para saber mais...

O expert SINTA foi desenvolvido pelo Laboratório de Informática Aplicada da UFC – Universidade Federal do Ceará. Para o download do arquivo instalador do Expert SINTA, acesse o link <<ftp://ftp.lia.ufc.br/sinta/sinta.zip>>, ou por meio do QR Code ao lado.



Para implementar esse exemplo no Shell Expert SINTA, primeiro deve-se identificar quais as variáveis a serem testadas durante a execução do SE. Dessa forma, e analisando o diagrama do fluxo das regras, verificamos que é necessário trabalhar com as seguintes variáveis:



- conceder linha;
- limite de crédito (objetivo);
- outras dívidas;
- prestação de imóvel;
- prestação do carro;
- renda;
- tempo no emprego.

No Expert SINTA, após criar uma nova base, as variáveis devem ser digitadas, devendo ser definida também a faixa de valores possíveis que a variável deverá receber. Veja a tela de variáveis na Figura 4.

No Expert SINTA, as variáveis podem ser de três tipos: **numérica**, **univalorada** e **multivalorada**. A faixa da variável numérica deve ficar na forma *min;max* (o valor mínimo, depois ponto e vírgula, e depois o valor máximo). A variável univalorada irá tratar valores “sim” ou “não”. No caso de a variável assumir uma série de valores discretos (uma lista de opções), então será multivalorada. Neste caso, as opções deverão ficar separadas por ponto e vírgula. Exemplo: para uma variável que assume valores relativos a cores primárias, a faixa deveria ser especificada como: *vermelha;azul;amarela*.

Após a digitação das variáveis, deve ser acessada a tela para a definição de objetivos. Na Figura 5 temos ilustrado o procedimento, de maneira que a variável objetivo (no caso, “limite de crédito”) deve ser colocada no box à direita da lista de variáveis.

Figura 4: Tela de entrada de variáveis e faixas de valores do Expert SINTA

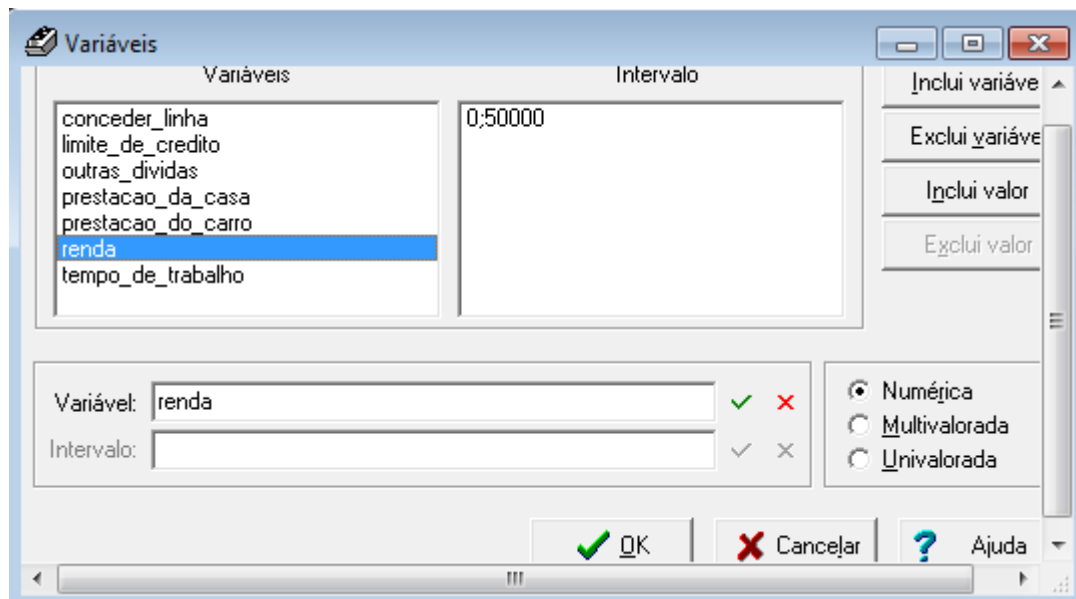
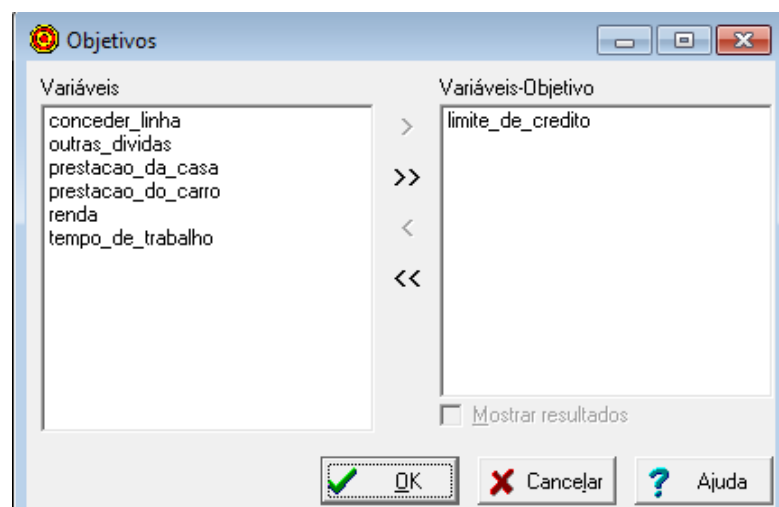
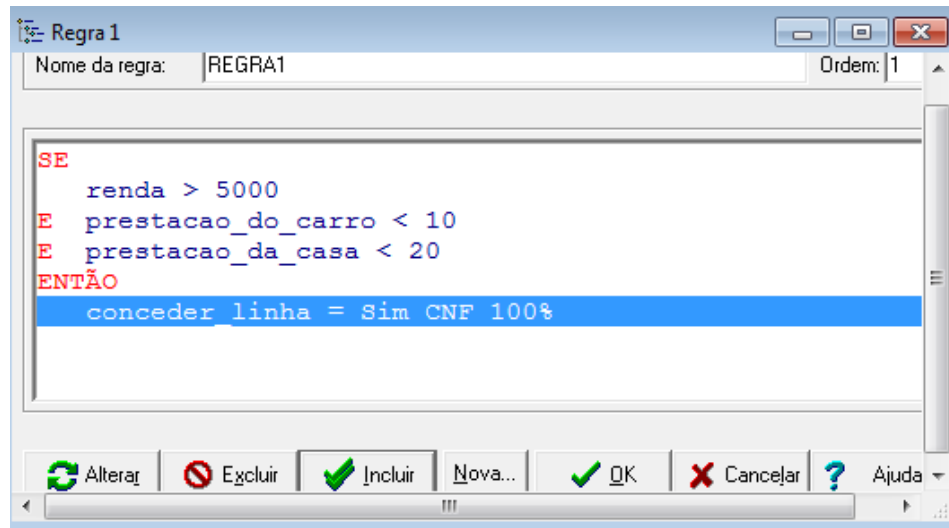


Figura 5: Definição da variável objetivo “limite de crédito”, no Expert SINTA



Após a definição do objetivo, as regras do SE já podem ser criadas. Ao clicar em *Nova Regra*, devemos adicionar a regra digitando primeiro a premissa da regra e depois a conclusão da regra. Como as variáveis já foram digitadas previamente, elas aparecem numa lista, facilitando a digitação. Veja na Figura 6 uma regra criada no Expert SINTA.

Figura 6: A regra com o teste das três variáveis *renda*, *prestação do carro* e *prestação da casa* (conforme o diagrama de fluxo na figura 2). Ao invés de digitar uma regra para cada teste, coloca-se todas numa regra apenas, adicionado a operação lógica “E” (ou seja, o teste de cada variável precisa ser verdadeiro simultaneamente para as três variáveis).



Após a digitação de todas as regras, a tela principal do Expert SINTA ficará como mostrado na Figura 7. Constatamos então que foram criadas 5 regras para a execução deste SE. Na Figura 8 temos o detalhamento de todas as regras (quando clicamos em “Visualizar”). A primeira regra se refere ao teste das variáveis *renda*, *prestação do carro* e *prestação da casa*, habilitando a variável *conceder linha*.

Figura 7: Tela principal com as regras digitadas

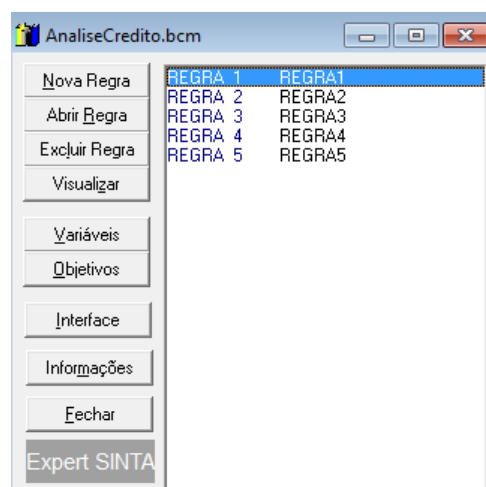


Figura 8: Listagem das regras digitadas no Expert SINTA.

```
REGRA 1
SE      renda > 5000
E      prestacao_do_carro < 10
E      prestacao_da_casa < 20
ENTÃO  conceder_linha = Sim CNF 100%
REGRA 2
SE      conceder_linha = Sim
E      tempo_de_trabalho >= 4
ENTÃO  limite_de_credito = 10000 CNF 100%
REGRA 3
SE      conceder_linha = Sim
E      tempo_de_trabalho < 4
E      outras_dividas < 5
ENTÃO  limite_de_credito = 10000 CNF 100%
REGRA 4
SE      conceder_linha = Sim
E      tempo_de_trabalho < 4
E      outras_dividas >= 5
ENTÃO  limite_de_credito = 3000 CNF 100%
REGRA 5
SE      conceder_linha = Não
OU      conceder_linha = DESCONHECIDO
ENTÃO  limite_de_credito = 0 CNF 100%
```

A segunda, terceira e quarta regras se referem à definição do limite de crédito que será concedido. Conforme o diagrama do fluxo das regras, a regra 2 testa para o caso positivo para o tempo de trabalho maior ou igual a 4 anos. No caso dessa regra falhar, a regra 3 e 4 comparam também a variável *outras dívidas*. Existe ainda a regra 5 para o caso de a variável *conceder linha* receba o valor “Não” na falha da regra 1.

Para executar o SE, ao clicar em *Iniciar consulta*, as telas com as perguntas para a entrada dos valores das variáveis são apresentadas sucessivamente. Ao final, o Expert SINTA apresenta a variável objetivo e o valor que foi obtido em função de uma execução específica.

Figura 9: Parte da execução do SE, com a tela de entrada da variável *renda*

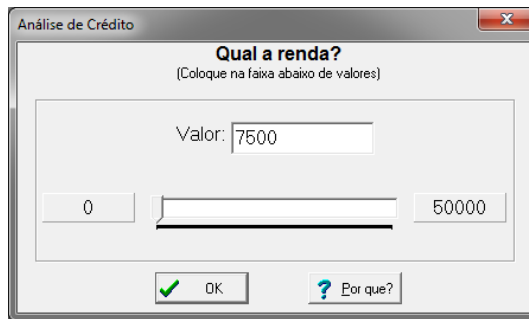
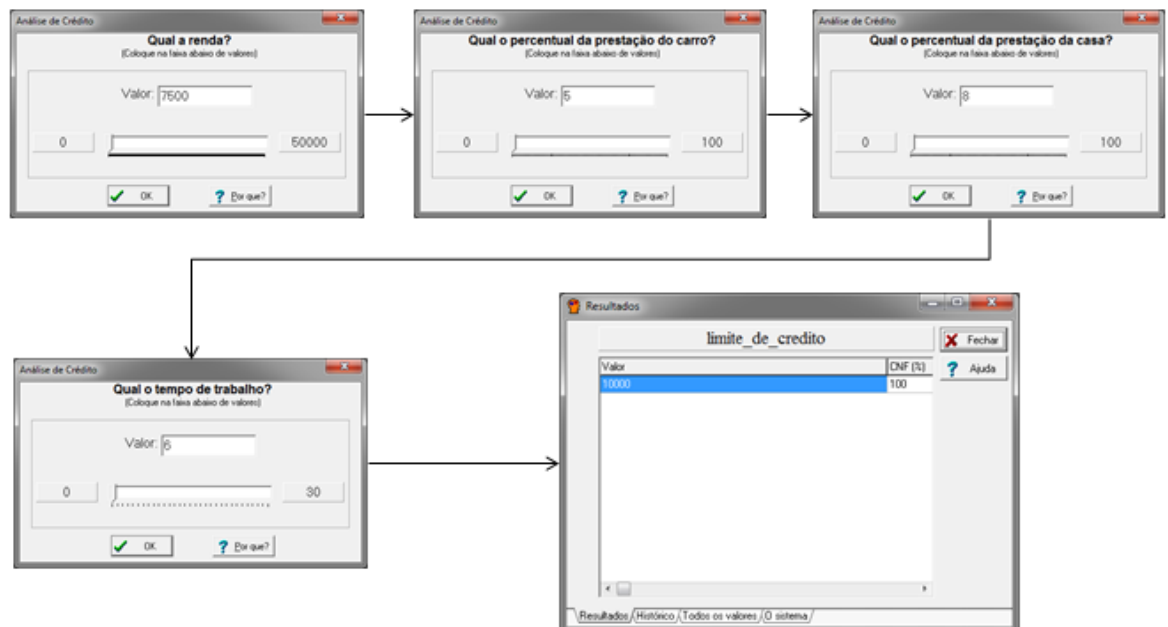


Figura 10: Uma execução específica do SE implementado, mostrando a concessão do valor de 10000 para a variável limite de crédito



Implementação em Linguagem Java

A implementação de sistemas especialistas pode ser feita também em uma linguagem de programação procedural, tal como o C++ ou Java. Dessa forma, uma regra de negócio que implementa o conhecimento de um especialista pode ser colocada em um sistema de informação para auxiliar o trabalho. O exemplo do SE para análise de crédito pode ser implementado utilizando classes ou de maneira mais simples utilizando chamadas de função. Na figura 11 temos o código em Java do programa na forma de uma função



com as variáveis de entrada como parâmetros da função e uma variável para o valor do limite de crédito como retorno. As regras podem ser implementadas na forma de condicionais com o teste das variáveis. No caso de regras que tenham grau de confiança, o código precisa ser alterado para contemplar o processamento dos graus de confiança para cada regra que é analisada.

TEMA 5: PROGRAMAÇÃO LÓGICA COM PROLOG

Dentro da linha simbólica, os estudos da lógica e da sua aplicação prática revestiram-se de fundamental importância para o aparecimento de sistemas inteligentes. Ainda que a lógica seja uma área de conhecimento com raízes na Filosofia, a partir do século XIX, os estudos de Boole (1815-1864) deram origem ao que se denomina atualmente de “álgebra booleana”. O matemático alemão Gottlob Frege (1848-1925) lançou a primeira versão de um cálculo de predicados da lógica, auxiliando a formalização exata do raciocínio dedutivo sobre conceitos matemáticos (PALAZZO, 1997, p.1).



Figura 11: Implementação em Java utilizando chamada de função para o exemplo de concessão de crédito. Abaixo, diferentes chamadas da *main* produzindo os valores de limites respectivos.

```
public static int analiseCredito(
    double renda,
    double percentualCarro,
    double percentualImovel,
    double tempoTrabalho,
    double percentualOutrasDividas) {

    int limiteCredito;

    if(renda >= 5000 &&
        percentualCarro < 0.10 &&
        percentualImovel < 0.20) {

        // Concede crédito
        if(tempoTrabalho >= 4) {
            limiteCredito = 10000;
        } else {
            if(percentualOutrasDividas < 0.05) {
                limiteCredito = 10000;
            } else {
                limiteCredito = 3000;
            }
        }
    } else {
        limiteCredito = 0;
    }
    return limiteCredito;
}

public static void main(String[] args) {
    // Limite de crédito: 10000
    System.out.println(analiseCredito(6000, 0.05, 0.1, 3, 2));
}

public static void main(String[] args) {
    // Limite de crédito: 0
    System.out.println(analiseCredito(4000, 0.05, 0.1, 3, 2));
}

public static void main(String[] args) {
    // Limite de crédito: 3000
    System.out.println(analiseCredito(5000, 0.05, 0.1, 3, 0.1));
}
```



No início da Segunda Guerra Mundial, praticamente a fundamentação teórica da lógica computacional estava pronta. Entretanto, não havia um meio prático para implementar a lógica e toda a sua necessidade de computações para a realização dos procedimentos de prova, ficando restrito a exemplos bem simplificados. Com a evolução dos computadores nos anos 1950, o potencial para implementação de programas para o raciocínio lógico começa a se tornar realidade (PALAZZO, 1997, p.1).

Na década de 1960, com a prova automática de teoremas, um procedimento que permitiu a interpretação procedimental da lógica, possibilitou que fossem estabelecidas as condições para compreendê-la como uma linguagem de programação de amplo uso. Mas o primeiro interpretador aparece somente em 1972, desenvolvido pela equipe do pesquisador Alain Colmerauer na Universidade de Aix-Marseille, com o nome de Prolog (acrônimo em francês para “Programmation em Logique”). Pesquisadores da Universidade de Edimburgo formalizaram a linguagem que se tornou referência para as implementações atuais e ficou conhecida como o “Prolog de Edimburgo” (PALAZZO, 1997, p. 2).

Uma das principais ideias subjacentes da programação em lógica é de que um algoritmo é constituído por dois elementos disjuntos: a **lógica** e o **controle**. O **componente lógico** corresponde à definição do que deve ser solucionado. O **componente de controle** estabelece como a solução pode ser obtida.

Quando programando em PROLOG, o programador precisa somente descrever o componente lógico de um algoritmo, deixando o controle da execução para ser exercido pelo sistema de programação em lógica utilizado. A atividade do programador passa a ser simplesmente a **especificação** do problema que deve ser solucionado.

Assim, um **programa em lógica** é a representação de um problema expresso por meio de um conjunto finito de um tipo especial de sentenças lógicas denominadas de cláusulas. O programador simplesmente especifica o



problema que deve ser solucionado. O controle fica a cargo do sistema de programação em lógica utilizado (PALAZZO, 1997, p. 3).

Ao contrário de um programa em linguagens como C ou Java, um programa em lógica não é a descrição de um procedimento para se obter a solução de um problema; o sistema é inteiramente responsável pelo procedimento a ser adotado na execução. Por isso, é dito que o paradigma fundamental da programação em lógica é o da **programação declarativa**, em oposição à **programação procedural**, típica das linguagens convencionais como C ou Java. O ponto focal da programação em lógica consiste em identificar a noção de computação com a noção de **dedução**.

Ao longo dos anos, a programação em PROLOG tem sido aplicada a diversas áreas, tais como:

- Sistemas de apoio à decisão;
- Simulações educacionais;
- Tutoriais inteligentes;
- Problemas matemáticos;
- Planejamento e roteirização;
- Resolução de problemas;
- Uso em regras de negócio;
- Processamento de linguagem natural.

Cláusulas em PROLOG

Um programa em lógica pode ser visto alternativamente como uma **base de dados**. Bases de dados convencionais descrevem apenas fatos (por exemplo, “Totó é um cão”). Além disso, sentenças em um programa em lógica



possuem, porém, um alcance maior, permitindo a representação de regras (por exemplo, “Todo cão é mamífero”).

Uma **cláusula** em PROLOG pode ser de três tipos:

- **Fatos:** Denotam uma verdade incondicional;
- **Regras:** Definem as condições que devem ser satisfeitas para que uma declaração seja considerada verdadeira;
- **Consulta:** Interrogação ao programa para verificar a verdade do conhecimento nele contido.

Na Tabela 1 temos um exemplo de cada cláusula, mostrando também um pouco da sintaxe de PROLOG. Num primeiro momento, os fatos e as regras precisam ser alimentados ao sistema PROLOG. Logo após, são feitas as consultas como se fossem perguntas relativas à dedução de novos conhecimentos, a partir dos conhecimentos que existem na base de conhecimento.

Tabela 1: Tipos de cláusulas em PROLOG

Cláusula	Exemplo	Descrição
Fato	pai(joão, luiz).	Este fato é traduzido como “João é o pai de Luiz”.
Regra	filho(X,Y) :- pai(Y,X).	Se X é filho de Y, então Y é pai de X. O sinal “:-” funciona como uma “seta à esquerda”, donde a conclusão à direita é obtida a partir das premissas que estão na direita.
Consulta	?- filho(luiz, joão). True	A consulta tenta obter se, a partir dos fatos e das regras existentes na base de conhecimento, Luiz é filho de João. A regra acima deduz um



novos fatos, a partir do fato e da regra anterior.

Uma cláusula é dividida nos seguintes componentes:

- **Corpo:** lista de objetivos separados por vírgulas, que devem ser interpretadas por conjunções;
- **Cabeça:** contém o resultado do corpo, inferido a partir dos objetivos.

Um fato contém apenas “cabeça”. A consulta só possui o “corpo”. Uma regra contém tanto cabeça quanto corpo.

Vimos pelo exemplo da regra na Tabela 1 que PROLOG trabalha com variáveis. Uma variável é composta de uma cadeia de letras, dígitos e do caractere “_”. O caractere “_” sozinho significa uma variável anônima, que é utilizada no caso de uma determinada variável não ter importância para o processo de dedução de uma solução. O fator chave que diferencia em PROLOG uma **variável** de uma **constante** é a primeira letra da variável sempre estar em maiúsculas (“caixa alta”). Caso esteja em letras minúsculas, o elemento é uma constante. Na tabela 1, “joão” e “luiz” estão em letras minúsculas, indicando que são constantes, e “X” e “Y” são variáveis. Mais alguns exemplos de variáveis:

X1
Resultado
Objeto2
Lista_Extensa
_var35
_344

Um dos ambientes para execução de PROLOG mais utilizados hoje em dia é o SWI-PROLOG. Desenvolvido desde 1987, é muito utilizado em

pesquisa, educação e também para aplicações comerciais. O software SWI-PROLOG é disponibilizado para uso concomitante com outras linguagens de programação, tais como C++, C#, Java, Python, etc.

Após instalado, podemos carregar os programas PROLOG codificados em arquivos de texto com o sufixo “.pl”, e executar as consultas diretamente no interpretador, conforme a Figura 12.


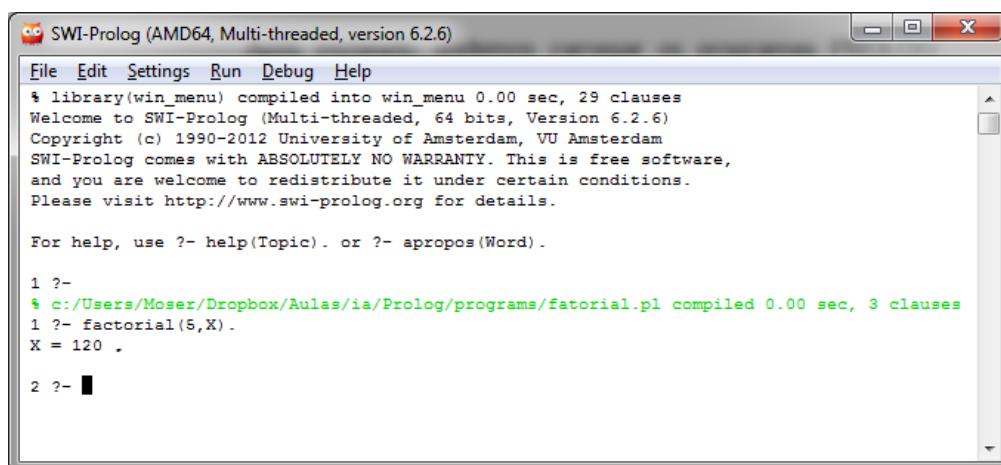
<p>Para saber mais...</p> <p>O SWI-PROLOG pode ser baixado diretamente do site: <http://www.swi-prolog.org/>, ou por meio do QR Code ao lado. Existem versões para diferentes sistemas operacionais e plataformas.</p>	
--	--

Figura 12: Tela do interpretador SWI-PROLOG com exemplo de um programa de cálculo de fatorial.



SÍNTESE

Neste capítulo foram estudadas técnicas dentro da linha simbólica de pesquisa da IA: os sistemas especialistas e a programação em lógica. Sistemas



Especialistas foram desenvolvidos para a resolução de problemas em um domínio bem específico, cujo conhecimento é coletado de pessoas especialistas em tal domínio. Portanto, os Sistemas Especialistas utilizam conhecimento heurístico manipulado por regras Se-Então. Dois SEs bastante citados na literatura são o MYCIN, utilizado para a área de diagnóstico de doenças infecciosas; e o DENDRAL, para prospecção geológica. Um Sistema Especialista é composto de uma base de conhecimento, de um mecanismo de inferência e de um quadro negro. A Programação em Lógica, tratada aqui com relação à linguagem PROLOG, deriva dos estudos da lógica de primeira ordem. O algoritmo é constituído dos componentes de controle e do componente lógico. Um programa em lógica é a representação de um problema expresso por meio de um conjunto finito de um tipo especial de sentenças lógicas denominadas de cláusulas. Programas em PROLOG têm sido utilizados em sistemas de apoio à decisão, simulações educacionais, tutoriais inteligentes, problemas matemáticos, planejamento e roteirização, resolução de problemas, uso em regras de negócio e processamento de linguagem natural. Uma cláusula em PROLOG pode ser um fato, uma regra ou uma consulta. Uma cláusula pode ser composta de corpo e cabeça.

REFERÊNCIAS

RUSSEL, S.; NORVIG, P. **Inteligência Artificial**. Tradução da 2. ed. Rio de Janeiro: Campus, 2004.

BITTENCOURT, G. **Inteligência Artificial – Ferramentas e Teorias**. Florianópolis: Ed. da UFSC, 1998.

PALAZZO, L. A. M. **Introdução a Prolog**. Pelotas: Editora UCPel, 1997.