Um Ambiente de Resolução de Problemas para Sistemas Baseados em Conhecimento

Alessandro Mueller (*)
Fernanda dos Santos Cunha (*)
Leandro José Komosinski (**)

Departamento de Ciências Estatísticas e da Computação (DCEC)
Centro Tecnológico
Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476
Florianópolis - SC
e-mail: cec1ljk@brufsc.bitnet

RESUMO

Neste artigo é apresentado um ambiente de resolução de problemas híbrido, baseado na combinação dos formalismos de representação do conhecimento de Regras de Produção e Frames. Cada formalismo é utilizado para representar uma parte do problema. Aplicações que utilizam mais de um formalismo podem ter desempenho superior aos sistemas de um único formalismo.

palavras-chaves: sistemas baseados em conhecimento, sistemas híbridos, frames e regras de produção

ABSTRACT

In this paper we show an Knowledge-based environment for problem solving. The environment combines two formalisms for knowledge representation: production rules and frames. Each formalism models a specific part of knowledge domain. Applications based on both formalisms can have better performance than others based on only one.

key-words: knowledge-based systems, hybrid systems, frames and production rules

* Alunos do Curso de Bacharelado em Ciências da Computação - UFSC
** Professor (MSc) do Curso de Bacharelado em Ciências da Computação - UFSC

1. INTRODUÇÃO

Nos sistemas de Inteligência Artificial orientados à resolução de problemas é dada muita ênfase ao papel do conhecimento, sendo, por esta razão, comumente chamados de Sistemas Baseados em Conhecimento (SBC)("Knowledge-Based Systems"). De fato, o desempenho e grau de "inteligência" de um SBC estão diretamente ligados à qualidade e usabilidade do conhecimento contido no sistema.Tais sistemas são caracterizados, de modo geral, por manterem uma Base de Conhecimentos explícita onde são armazenadas informações a respeito de um domínio de conhecimento específico [BRACHMAN85]. A ênfase sobre o conhecimento nos SBC fez com que a área de Representação do Conhecimento (RC) assumisse papel fundamental [ALTY 84],[FIKES 85],[WOODS 86],[WAH 89], levando ao desenvolvimento de vários formalismos.

A definição de um formalismo de RC em um SBC determina os tipos de inferências que podem ser utilizados. Tradicionalmente, o formalismo mais usado é o de Regras de Produção (RP). Este formalismo apresenta como vantagens a sua simplicidade sintática e seu apelo intuitivo. Entretanto, RP não provêm facilidades de representação de estruturas mais complexas. Em particular, o seu poder de expressividade é inadequado para descrição de objetos e relações entre eles (tal como objetos compostos) [KOMOSINSKI 90]. Além disso, SBCs que utilizam apenas Regras de Produção como formalismo de representação de conhecimento tendem a apresentar um baixo desempenho quando o número de regras torna-se grande.

Considerando os aspectos acima, percebeu-se que sistemas baseados em um único formalismo de RC (em particular o formalismo de RP) limitam o tipo de informação que pode ser representado e tendem a ficar ineficientes a medida que cresce a quantidade de informações que precisam ser armazenadas. Assim surge uma nova geração de SBC utilizando mais de um formalismo. A combinação típica emprega Regras de Produção e Frames (que se baseia na noção de objeto), formando uma **representação híbrida** que combina as vantagens de ambos formalismos. Os sistemas CENTAUR [AIKINS 85], KEE [FIKES 85], KEOPS [ROCHE 89] são exemplos deste enfoque híbrido.

Com o objetivo de desenvolver SBCs onde a RC e os respectivos mecanismos de inferência sejam definidos pelos formalismos de Regras de Produção e Frames está sendo desenvolvido, no Laboratório de Inteligência Artificial do DCEC, um Ambiente de Desenvolvimento de Sistemas Baseados em Conhecimento (ADSBC). O ADSBC é composto por dois módulos principais: o Módulo de Edição de Base de Conhecimento (MEBC) e o Módulo de Resolução de Problemas (MRP). Através do MEBC, o Engenheiro do Conhecimento (EC) pode editar de forma agradável uma BC (formada por Regras de Produção e Frames) a respeito de um domínio de conhecimento qualquer. Uma vez editada a BC, esta é traduzida de um formato agradável ao EC para um formato que permita a sua eficiente utilização pelo MRP.

O Módulo de Edição de Base de Conhecimento está atualmente em fase de especificação e o Módulo de Resolução de Problemas em fase de implementação (utilizando linguagem PROLOG). Neste artigo são descritos aspectos ligados à RC do ADSBC e ao Módulo de Resolução de Proble-

mas. Comenta-se também as várias possibilidades de combinação dos formalismos de RP e Frames previstas para o ADSBC.

2. REPRESENTAÇÃO DE CONHECIMENTO NO ADSBC

A Representação do Conhecimento (RC) no ADSBC é feita através da combinação de Frames e Regras de Produção. Os itens 2.1 e 2.2 descrevem, respectivamente, a estrutura de um Frame e de uma RP. O item 2.3 descreve as várias possibilidades e vantagens em se combinar os dois formalismos de RC.

2.1. FRAMES

O formalismo de RC de Frames faz com que um determinado domínio de conhecimento seja modelado através da identificação e representação dos objetos que pertencem àquele domínio. O termo objeto deve ser entendido como sendo desde objetos concretos (como por exemplo automóvel, carro, pessoa, etc) até objetos abstratos, situações ou eventos (como por exemplo uma reunião ou um jantar em um restaurante). Em ambos os casos os objetos são caracterizados por um conjunto de atributos ("slots") e, para cada atributo, está associado um valor.

Quando Minsky [MINSKY 75] definiu o formalismo de Frames o fez de maneira muito genérica e sem nenhuma preocupação com uma estrutura que pudesse servir de base para algum sistema [GOMEZ 81]. Isto permitiu que, em cada sistema que utilizasse a idéia de Frames, este fosse definido de forma a atender necessidades específicas. Seguindo esta conduta, no sistema ADSBC, um frame possui uma estrutura própria (descrita em detalhes mais adiante).

No sistema ADSBC, um frame representa uma classe de objetos e uma instância representa um objeto (indivíduo) que pertence a uma determinada classe. Assim, um problema é modelado através da definição de frames e instâncias.

Segundo a semântica definida para o formalismo de Frames no sistema ADSBC, uma instância sempre está consistente com as definições contidas no frame à qual está ligado. Isto significa que todos os slots definidos para o frame estejam preenchidos com valores válidos. Assim, qualquer alteração no valor de um slot só será aceita se este valor estiver consistente com a especificação definida para o slot.

No ADSBC os frames são definidos por fatos PROLOG e apresentam a seguinte estrutura:

```
frame( <nome_frame>,
       <superclasse>,
       <slots_de_documentação>,
       <slots_de_controle>,
       <slots_de_definição> ).
```

A seguir são apresentados em detalhes os componentes da estrutura de um frame.

2.1.1. Nome do Frame

O termo <nome_frame> é um símbolo PROLOG que identifica o nome do frame.

2.1.2. Superclasse

O termo <superclasse> é um símbolo PROLOG que identifica a superclasse do frame. Através da declaração da superclasse é possível construir taxonomias de frames resultando em um poderoso mecanismo de inferência: a Herança. Este mecanismo será explicado no item 3.3.1.

2.1.3. Slots de Documentação

Os Slots de Documentação têm, como o próprio nome indica, a função de documentar o frame onde estão declarados. No ADSBC os slots de documentação definidos são: "author", "date" e "text".
O preenchimento destes slots durante a definição do frame é opcional. Além disso, eles não são herdáveis e não fazem parte do processo de instanciação.

O slot "author" tem como valor um termo PROLOG que reprsenta o nome do autor do frame. O slots "date" tem como valor um termo PROLOG que representa a data de criação do frame. O slot "text" tem como valor um termo PROLOG que descreve a classe de objetos sendo modelada pelo frame.

2.1.4. Slots de Controle

Os Slots de Controle permitem estabelecer condições para que se possa iniciar o processo de criação de uma instância e definir os efeitos colaterais associados à criação (ou não) desta. No ADSBC estão definidos três slots de controle: "to_instantiate", "if_instantiated" e "if_not_instantiated".

O slot "to_instantiate" define condições para que se possa iniciar a instanciação de um frame. O slot "if_instantiated" define ações que devem ser realizadas caso se tenha conseguido instanciar o frame (isto é, todos os slots de definição foram preenchidos com valores válidos). Já o slot "if_not_instantiated" define ações que devem ser realizadas caso não se tenha conseguido instanciar o frame.

A definição dos Slots de Controle é opcional. Quando definido, o valor de um slot de controle é formado por uma expressão (um predicado ou um conjunção de vários predicados) que pode ser avaliada diretamente pelo interpretador PROLOG.

A motivação para a existência dos três Slots de Controle é permitir que o processo de instanciação seja sensível (e também sensibilize) ao contexto onde está ocorrendo, tornando, desta forma, a própria instanciação mais eficiente. Por exemplo, a instanciação de um frame "Analisador_de_Defeito" pode estar condicionada à existência de alguma instância que represente um defeito possível.

2.1.5. Slots de Definição

Os Slots de Definição são definidos pelo Engenheiro de Conhecimento e representam atributos comuns a todos os objetos pertencentes à classe modelada pelo frame.

Como os Slots de Definição exercem o papel principal na representação do conhecimento envolvido na classe de objetos que está sendo modelada, a sua composição difere da composição "slot-valor" dos outros tipos de slots. A idéia é que existam outras estruturas que complementem a descrição "slot-valor". Esta complementação tem os seguintes objetivos :

- definir o valor do slot em tempo de edição (e não em tempo de execução), quando possível;
- restringir o conjunto de valores que poderão ser associados ao slot (em tempo de execução) de forma que o valor seja consistente com a semântica associada ao slot;
- definir como obter o valor de um slot;
- definir quais os efeitos colaterais existentes quando do preenchimento (ou não preenchimento) de um slot.

Para atender os objetivos acima, um slot pode conter as seguintes informações:

- nome do slot (slot_name)
- valor (value)
- valor default (default_value)
- restrição de valor (value_restriction)
- "demon" se_necessário (if_needed)
- "demon" se_preenchido (if_filled)
- "demon" se_nao_preenchido (if_not_filled).

O termo "slot_name" é um símbolo PROLOG e representa o nome do slot, cujo preenchimento é obrigatório.

O termo "value" é qualquer constante PROLOG e representa o valor do slot conhecido em tempo de edição do frame e o slot não pode assumir nenhum outro valor.

O termo "default_value" é qualquer constante PROLOG e define o valor default do slot. Este termo deve ser usado quando o valor do slot é conhecido em tempo de edição do frame e o valor representa uma informação de senso comum, isto é, uma informação que se assume como verdadeira caso não exista nenhuma indicação contrária.

O termo "value_restriction" é uma expressão formada por um predicado (ou conjunto de predicados) que podem ser avaliados diretamente pelo interpretador PROLOG. Um valor é considerado consistente com o slot se a expressão avaliada for verdadeira. O termo "value_restriction" permite restringir os possíveis valores que o slot pode assumir, sendo usado para garantir a consistência do valor do slot.

O termo "if_needed" é uma expressão formada por um predicado (ou conjunto de predicados) que pode ser avaliada diretamente pelo interpretador PROLOG. Caso o resultado da avaliação for verdadeiro, o valor do slot será o valor de uma variável da expressão. Caso isto não ocorra, o slot não será preenchido.

O termo "if_filled" é uma expressão formada por um predicado (ou conjunto de predicados) que pode ser avaliada diretamente pelo interpretador PROLOG. O termo "if_filled" é avaliado caso o slot tenha sido preenchido com valor válido (consistente com as especificações definidas na faceta "value_restriction"). Caso não se tenha conseguido preencher o slot com valor válido, o termo "if_not_filled" é avaliado.

2.2. REGRAS DE PRODUÇÃO

As Regras de Produção (RP) do sistema ADSBC têm a estrutura clássica "SE condições ENTÃO ações". As condições estão normalmente associadas ao estado de uma ou mais instâncias. As ações, por sua vez, podem modificar o estado destas instâncias (alterando o valor de um ou mais slots da instância).

Na abordagem acima, a resolução de problemas é feita de forma tradicional, isto é, identifica-se os objetos pertinentes ao problema e estes são modelados através de frames e instâncias; a seguir modela-se o comportamento destes objetos através de um conjunto de RP. Resolver um problema consiste então em se determinar qual a sequência de ações que transforma as instâncias representando o estado inicial do problema em um conjunto de instâncias que representam o estado final (solução) (encadeamento para frente).

As RP são representadas por fatos PROLOG com a seguinte estrutura:

```
regra ( <nome da regra>,
        <grupo>,
        <condições>,
        <ações> ).
```

O termo <nome da regra> é um símbolo PROLOG e identifica unicamente a regra.

O termo <grupo> é um símbolo PROLOG e identifica que a regra pertence a um determinado grupo de regras relativas a um mesmo problema.

O termo <condições> é uma conjunção de predicados avaliáveis diretamente pelo interpretador PROLOG. Diz-se que as condições de uma regra estão satisfeitas quando todos os predicados que formam as condições são considerados verdadeiros no momento da avaliação.

O termo < ações> é um conjunto de predicados avaliáveis diretamente pelo interpretador PROLOG. Quando uma regra é disparada, cada predicado que faz parte das ações é avaliado.

2.3. COMBINANDO REGRAS E FRAMES

Segundo o enfoque de utilização do formalismo de RP descrito no item 2.2, percebe-se claramente que as regras assumem o papel ativo (principal) em relação ao formalismo de Frames com relação ao processo de inferência (o encadeamento de regras é mais importante que a instanciação). Para caracterizar esta situação utilizamos o termo "Regras Dominantes".

Além do enfoque de utilização de RP descrito acima existe uma outra forma de combinar-se RP e Frames. Nesta outra forma, a qual chamamos "Frames Dominantes", os frames assumem o papel ativo em relação ao formalismo de RP com relação ao processo de inferência (a instanciação é mais importante que o encadeamento de regras). As RP são reunidas em grupos afins e estes grupos são associados à slots. O processo de encadeamento de regras é ativado quando se deseja conhecer o valor de um dos slots da instância que está sendo criada. Desta forma combina-se dois mecanismos de inferência poderosos (instanciação e encadeamento) que permitem unir as vantagens específicas de cada mecanismo num mesmo sistema, tornando-se então um sistema híbrido.

3. MÓDULO DE RESOLUÇÃO DE PROBLEMAS DO ADSBC

O Módulo de Resolução de Problemas (MRP) do sistema ADSBC é composto por um Módulo Interpretador (MI), um Módulo Base de Conhecimento (MBC) e um Módulo Motor de Inferência (MMI) (ver Fig. 1). Estes módulos são descritos a seguir.

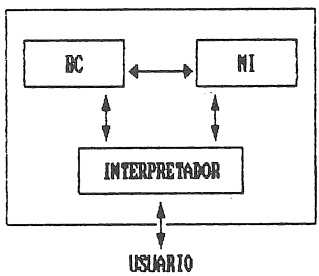

Fig. 1 - Arquitetura do ADSBC

3.1. MÓDULO INTERPRETADOR

O sistema ADSBC funciona dentro do ambiente do interpretador PROLOG em que foi desenvolvido. Assim, para se poder utilizar o sistema é preciso primeiro carregar o interpretador PROLOG. Logo em seguida é carregado o sistema ADSBC que fica armazenado na área de memória gerenciada pelo PROLOG. Este tipo de convivência entre o interpretador PROLOG e o sistema apresenta várias vantagens (descritas a seguir) em relação à facilidade de uso e poder de resolução de problemas.

Na verdade, o sistema ADSBC é formado por um conjunto de predicados cuja sintaxe é idêntica aos predicados PROLOG. Estes predicados permitem a criação de frames, instâncias, consultas à Base de Conhecimento e ativação dos vários mecanismos de inferência (ver exemplo no item 3.3). Desta forma, o Módulo Interpretador tem o papel de traduzir consultas feitas através da sintaxe PROLOG (realizadas pelo usuário do sistema) para uma sintaxe mais específica do sistema ADSBC e encaminhá-las ao Módulo Base de Conhecimento ou ao Módulo Motor de Inferência (ver Fig. 1).

A compatibilidade entre o interpretador PROLOG e o sistema ADSBC apresenta as seguintes vantagens:

- como a sintaxe do ADSBC é igual à sintaxe do PROLOG não existe a necessidade do Engenheiro do Conhecimento/usuário aprender uma nova linguagem. Isto torna a programação mais familiar do que já é conhecido.
- predicados implementados totalmente em PROLOG são naturalmente acessíveis ao sistema ADSBC (e vice-versa). Assim, pode-se explorar as características próprias de cada linguagem.
- o próprio ambiente de programação PROLOG pode ser utilizado para se desenvolver sistemas de resolução de problemas.
- adiciona-se o Método de Resolução como mais um método de inferência disponível para a resolução de problemas.
- pode-se usar a lógica dos predicados como mais um formalismo de representação do conhecimento (além de regras e frames).

3.2. MÓDULO BASE DE CONHECIMENTO (MBC)

Toda informação associada ao domínio específico de conhecimento é armazenada na Base de Conhecimento. A Base de Conhecimento no ADSBC é composta pela Memória de Conhecimento e pela Área de Trabalho, conforme ilustra a Fig. 2.

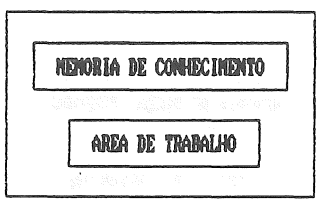

Fig. 2 - Base de Conhecimento

3.2.1. A MEMÓRIA DE CONHECIMENTO (MC)

A Memória de Conhecimento é uma memória onde as informações disponíveis sobre o domínio do problema são armazenadas. Como foi citado no item 2, o conhecimento no ADSBC pode ser representado pelos formalismos de regras e/ou frames. Por esta razão, a Memória de Conhecimento do ADSBC está subdividida em dois módulos, apresentados na Fig. 3, que são a Memória de Regras Permanente e a Memória de Frames. Estes dois módulos representam todas as informações estáticas (permanentes) do problema.

A Memória de Regras Permanente contém o conhecimento sobre o domínio do problema representado através do formalismo de regras (Regras Dominantes).

A Memória de Frames é formada pelos frames (classes de objetos) definidos pelo engenheiro de conhecimento e manipulados na aplicação, e por grupos de regras. Estes grupos de regras são formados pelos

identificadores de regras afins ligadas intimamente com os frames e que têm extrema importância quando se adota o ponto de vista de Frames Dominantes.

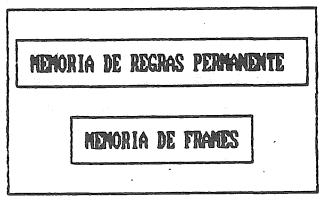

Fig. 3 - Memória de Conhecimento

3.2.2. A ÁREA DE TRABALHO (AT)

A Área de Trabalho é uma memória utilizada para armazenar as informações envolvidas no processo de inferência. No ADSBC a Área de Trabalho é formada pela Memória de Instâncias e pela Memória de Regras Temporária, como ilustra a Fig. 4. Esta divisão é feita pelo mesmo motivo que se divide a Memória de Conhecimento, ou seja, por existir mais de um formalismo de representação de conhecimento.

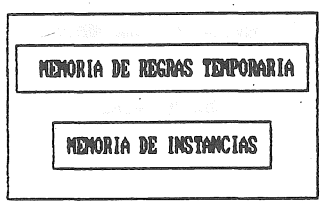

Fig. 4 - Área de Trabalho

A Memória de Instâncias contém todas as instâncias dos frames criados durante a execução da aplicação. A Memória de Instâncias do ADSBC corresponde a Memória de Fatos dos SBC que usam apenas um formalismo de representação de conhecimento.

A Memória de Regras Temporária armazena o conjunto de regras utilizadas no processo de inferência. Esta memória é considerada temporária porque é utilizada no processo de inferência somente quando as informações representadas pelos frames ocuparem um papel de maior importância no contexto (ponto de vista de Frames Dominantes). Nesta situação, as regras armazenadas nesta memória não serão todas aquelas definidas na Memória de Regras Permanente. Somente as regras que fazem parte dos grupos de regras (definidos na Memória de Frames) relacionados com as instâncias estarão presentes na Memória de Regras Temporária.

3.3. MÓDULO MOTOR DE INFERÊNCIA (MMI)

O motor de inferência de um Sistema Especialista (e Sistema Baseado em Conhecimento) constitui-se num conjunto de procedimentos que deduzem informações novas a partir de informações conhecidas [BARR 86]. Portanto, o Módulo Motor de Inferência do sistema ADSBC contém os métodos de inferência inerentes aos formalismos de representação do conhecimento de Regras de Produção e de Frames.

Os métodos de inferência existentes são:

- instanciação e classificação (para formalismo de Frames).
- encadeamento para frente (para formalismo de Regras).

3.3.1. INSTANCIAÇÃO

O processo de instanciação pode ser visto como um casamento (matching) entre um objeto observado (composto por seus slots e respectivos valores) e um modelo de objeto representado pelo frame. O casamento será válido quando todos os slots de definição do frame forem preenchidos com valores consistentes obtidos no objeto observado.

O algoritmo de criação de uma instância do frame F é composto dos seguintes passos:

1. Testar o slot de controle "to-instantiate" (se estiver definido na frame F ou for herdado por alguma superclasse sua).
2. Se o teste do passo 1 for realizado com sucesso então preencher todos os slots de definição definidos no frame F ou herdados por todas as suas superclasses.
3. Se todos os slots de definição foram preenchidos com valores válidos então avaliar o slot de controle "if-instantiated" definido no frame F ou herdado por alguma superclasse de F.
4. Se o teste do slot "to-instantiate" falhou ou algum dos slots de definição do frame F não foi preenchido então avaliar o slot de controle "if-not-instantiated" definido em F ou herdado por alguma superclasse de F.

Como exemplo da aplicabilidade da instanciação pode-se imaginar uma taxonomia de frames, onde cada frame representa um determinado objeto apresentando um tipo de defeito. Assim, instanciar um frame significa verificar se determinado objeto apresenta o defeito modelado no frame.

3.3.2. CLASSIFICAÇÃO

O mecanismo de classificação consiste em criar uma instância para cada frame da taxonomia, partindo do frame raiz até atingir as folhas. A passagem de nível na taxonomia só é feita se o frame do nível superior for instanciado.

Considerando o mesmo exemplo de taxonomia de frames mostrado no item 3.3.1, uma classificação serveria para se determinar quais os defeitos (se existissem) existentes em um determinado equipamento.

Na classificação não se sabe a priori quais frames gerarão instâncias. Este não determinismo é particularmente adequado ao processo

de diagnóstico em geral.

O algoritmo de classificação é semelhante ao algoritmo da instanciação. A diferença está no fato de que, para se realizar uma instanciação de um frame da taxonomia, não é utilizada a herança de slots.

3.3.3. ENCADEAMENTO DE REGRAS

Quanto a utilização das regras de produção, o processo é o tradicional encadeamento de regras. O motor de inferência pode ter dois tipos de estratégia de controle:

- Irrevogável (sem backtracking)
- Revogável (com backtracking).

Em uma estratégia irrevogável, após a geração do conjunto de conflitos, uma regra é selecionada de acordo com algum critério estabelecido e disparada. O processo se repete até que a solução seja encontrada ou nenhuma outra regra possa ser disparada na tentativa de encontrar a solução do problema. Nesta estratégia, o estado em que se encontra a memória de fatos anterior ao disparo da regra é perdido. Isto impossibilita a procura da solução por outro caminho, que não aquele escolhido pelo disparo da regra selecionada, caso a solução não tenha sido alcançada.

Na estratégia revogável existe a possibilidade de se alcançar o objetivo a partir de um outro caminho, pois o estado da memória de fatos anterior ao disparo de cada regra é armazenado.

3.3.4 EXEMPLO

Neste item é mostrado, através de um exemplo simples, como o Módulo de Resolução de Problemas do sistema ADSBC pode ser utilizado. O domínio de conhecimento do exemplo é a Modelagem de Computadores.

Deseja-se modelar um tipo de computador que apresente as seguintes características:

- possua um nome que identifique o computador. Este nome deve ser um símbolo. O valor deve ser perguntado ao usuário.
- possua uma memória cujos valores possíveis são 512, 640, 1024 ou 2048 Kb. Para se conhecer a quantidade de memória de um computador em particular (uma instância da classe computador) deve-se perguntar ao usuário.
- possua um processador cujos tipos possíveis são 80286, 80386 ou 80486. Caso não seja informado pelo usuário o tipo do processador então assume-se o tipo 80386 por default.
- possua um preço cujo valor depende da quantidade de memória e do tipo de processador. O preço é calculado da seguinte forma:

  Se memória < 1024 e processador < 80386
  Então preço = (memória * 2) + processador

  Se memória > 640 e processador > 80286
  Então preço = (memória * 1.5) + (processador * 1.5)

O frame "Computadores", descrito abaixo, modela a classe dos Computadores conforme as especificações anteriores. As informações descritas a seguir estão normalmente armazenadas em um arquivo (por exemplo, "COMPUT.TXT").

```
Frame: Computadores
   author: 'Leandro'
   date  : '10 - 8 -91'
   text  : 'Este frame representa a classe dos Computadores'

   slot nome:
      value_restriction: slot_value(nome,Nome) and
                         symbolp(Nome)
      if_needed: write('Qual o nome ?') and
                 read(Nome) and
                 value_is(Nome)

   slot memoria:
      value_restriction: slot_value(memoria, Mem) and
                         pertence(Mem, [512,640,1024,2048])
      if_needed: write('Quanta memoria ?') and
                 read(Mem) and
                 value_is(Mem)

   slot processador:
      default_value: 80386
      value_restriction: slot_value(processador,Proc) and
                         pertence(Proc, [80286,80386,80486])
      if_not_filled: write('Processador invalido !!')

   slot preco:
      if_needed: rule_groups([grupo_preco]),
                 run_rules,
                 preco(Pr),
                 value_is(Pr),
                 clear_groups  % Limpa MRT %

Rule_Group: grupo_preco

  Rule: pouca_memoria
   IF    slot_value(memoria,Mem) and
         menor(Mem, 1024) and
         slot_value(processador,Proc) and
         menor(Proc, 80386)
   THEN Preco is (Mem * 2) + Proc and
         assert(preco(Preco)) and
         write('Disparada regra pouca_memoria')
```

```
  Rule: muita_memoria
   IF    slot_value(memoria,Mem) and
         maior(Mem,640) and
         slot_value(processador,Proc) and
         maior(Proc,80286)
   THEN Preco is (Mem * 1.5) + (Proc * 1.5)
         assert(preco(Preco))
         write('Disparada regra muita__memoria')

%  o predicado "pertence" (usado nos slots memoria e processador)  de-
senvolvido completamente em PROLOG %

pertence(Elem,[Elem|_]) :- !.
pertence(Elem,[_|Resto]) :-
  pertence(Elem,Resto),!.
```

A seguir é mostrado um trecho de uma sessão de utilização do MRP. O símbolo "?" representa o prompt do interpretador PROLOG. As linhas iniciadas por "%" são comentários explicando o objetivo de cada comando. Assume-se que o interpretador PROLOG, o sistema ADSBC e o arquivo "COMPUT.TXT" já foram carregados.

```
%  para criar uma instância usa-se o comando make. A instância  criada
fica armazenada no fato uma_inst %

? make(computadores,Inst) and assert(uma_inst(Inst))

Qual o nome ? marca1
Quanta memoria ? 1024
Disparada regra muita_memoria

Inst = instance(frame,computadores,
                [nome,marca1,
                 memoria,1024,
                 processador,80386,
                 preco,122115)

% para recuperar o valor de um slot usa-se o comando slot_value %

? uma_inst(I) and slot_value(I,memoria,Mem)
I   = instance(frame,computadores,...)
Mem = 1024

? uma_inst(I) and slot_value(I,nome,N)
I = instance(frame,computadores,...)
N = marca1

% para alterar o valor de um slot usa-se o comando put_value. Quando o
valor proposto é inconsistente com as especificações definidas no fra-
me a consulta falha %
```

```
? uma_inst(I) and put_value(I,processador,68020,Inst)
Processador invalido !! % ativado o demon if_not_filled %
no

? uma_inst(I) and put_value(I,nome ,marca2,Inst)
I = instance(frame,computadores,
             [nome,marca1,
              memoria,1024,
              processador,80386,
              preco,122115)

Inst = instance(frame,computadores,
                [nome,marca2
                 memoria,1024,
                 processador,80386,
                 preco,122115)
```

4. CONCLUSÃO

Neste artigo foi apresentado o Módulo de Resolução de Problemas do sistema ADSBC. Este sistema combina os formalismos de representação do conhecimento de Regras de Produção e de Frames. Por estar contido dentro do ambiente interpretativo da linguagem PROLOG pode-se dispor também do formalismo da lógica dos predicados. A combinação de vários formalismos permite uma representação mais expressiva de cada tipo de informação contido num mesmo domínio de conhecimento.

Apresentou-se também algumas formas de se combinar Frames e Regras de Produção. Percebeu-se que cada frame de uma taxonomia pode representar um contexto de aplicabilidade de um conjunto restrito de Regras de Produção. Assim, combinando-se os mecanismos de inferência de classificação (que envolve o mecanismo de instanciação) com o de encadeamento de regras pode-se obter um desempenho bastante superior à um sistema composto somente por Regras de Produção onde todos competem entre si em igualdade de condições, isto é, em cada ciclo de execução todas as regras devem ser testadas para a geração do conjunto de conflitos.

A perspectiva futura do sistema ADSBC é a sua utilização em problemas de grande porte para que se possa analisar mais precisamente as reais vantagens da utilização de sistemas híbridos.

5. REFERÊNCIAS BIBLIOGRÁFICAS

[AIKINS 85] AIKINS, J.S., "A Representation Scheme Using Both Frames and Rules", Rule-Based Experts Systems, B.G, Buchanan and E.H. Shortlife (eds), Addison-Wesley, 1985.

[ALTY 84] ALTY, J.L. & COOMBS, M.J., "Expert Systems Concepts and Examples", NCC Publications, 1984.

[BARR 86] BARR, A. & FEIGENBAUM, E., "The Handbook of Artificial Intelligence", Vol. 1, Addison-Wesley, 1986.

[BRACHMAN 85] BRACHMAN, R.J. & LEVESQUE, H.J., "Readings in Knowledge Representation", Morgan Kaufmann, 1985.

[FIKES 85] FIKES, R.E. & KEHLER, T., "The Role of Frame-Based Representation in Reasoning", Communications of the ACM, 28(9):904 - 920, septeber 1985.

[GOMEZ 81] GOMEZ, F. & CHANDRASEKARAN, B., "Knowledge Organizations and Distribution for Medical Diagnosis", IEEE Trans. on Systems, Man, and Cybernetics, 11(1):34-42, january 1981.

[HARMON 85] HARMON, P. & KING, D., "Expert Systems - Artificial Inteligence in Business", John Wiley & Sons, INC., 1985.

[KOMOSINSKI 90] KOMOSINSKI, L.J., "Uma Linguagem Centrada em Frames para Desenvolvimento de Sistemas Baseados no Conhecimento", março 1990.

[NILSSON 80] NILSSON, N.J., "Principles of Artificial Inteligence", Tioga Publishing Company, 1980.

[ROCHE 89] ROCHE, C. & LAURENT, J-P., "Les approches objets et le langage LRO2 (KEOPS)", Technique et Science Informatiques, 8(1):21-39, 1989.

[WAH 89] WAH, B.W., et alii, "Computers for Symbolic Processing", Proc. of the IEEE, 77(4):509 - 539, april 1989.

[WATERMAN 86] WATERMAN, D., "A Guide to Expert Systems", Addison-Wesley, 1986.

[WOODS 86] WOODS, W., "Important Issues in Knowledge Representation", Proc. of the IEEE, 74(10):1322 - 1334, october 1986.