P R O J E T O E I M P L E M E N T A Ç Ã O

D E U M I N T E R P R E T A D O R

P R O L O G B Á S I C O

ERALDO CRUZ LACET

PROJETO E IMPLEMENTAÇÃO DE UM INTERPRETADOR PROLOG BÁSICO

Dissertação apresentada ao curso de PÓS-GRADUAÇÃO EM SISTEMAS E COMPUTAÇÃO da Universidade Federal da Paraíba, em cumprimento às exigências para obtenção do Grau de Mestre.

ÁREA DE CONCENTRAÇÃO: CIÊNCIA DA COMPUTAÇÃO

HÉLIO DE MENEZES SILVA
Orientador

CAMPINA GRANDE
JUNHO - 1985

A minha esposa Lúcia

e aos meus pais

dedico esta obra.

PROJETO E IMPLEMENTAÇÃO DE UM INTERPRETADOR PROLOG BÁSICO

E R A L D O C R U Z L A C E T

TESE SUBMETIDA AO CORPO DOCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS E COMPUTAÇÃO - OPÇÃO CIÊNCIA DA COMPUTAÇÃO, DO CENTRO DE CIÊNCIAS E TECNOLOGIA DA UNIVERSIDADE FEDERAL DA PARAÍBA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M.Sc.).

Aprovada por:

COMISSÃO EXAMINADORA

Prof. HÉLIO DE MENEZES SILVA - M.Sc.
Presidente

Prof. JOSÉ SÉRGIO ANTUNES SETTE - Doutor
Examinador

Prof. GIUSEPPE MONGIOVI - M.Sc.
Examinador

Prof. BERNARDO LULA JÚNIOR - M.Sc.
Examinador

CAMPINA GRANDE - Pb
JUNHO/1985

S U M Á R I O

RESUMO

Esta dissertação consiste no projeto e implementação de um interpretador PROLOG básico (isto é, sem uma ampla biblioteca de predicados e funções, e sem otimizações elaboradas, tais como aquela para recursão a direita). Seu objetivo é não apenas prover uma ferramenta para programação em lógica, como também servir como ponto de partida para futuras pesquisas na área de desenvolvimento de interpretadores PROLOG.

No projeto do interpretador utilizou-se a metodologia de Constantine (ver, por exemplo, Stevens - 1981) para obter-se um programa de fácil entendimento, manutenção e ampliação. Implementado no PASCALVS do IBM-4341, evitou-se utilizar as extenções da linguagem PASCAL disponíveis naquele compilador, favorecendo-se a portabilidade do interpretador. No processo resolutivo, seguindo-se van Emden (1981) e Ferguson (1981), usa-se o modelo de "árvore de prova com estruturas compartilhadas" como uma boa maneira de implementar a resolução LUSH.

UNIVERSIDADE FEDERAL DA PARAÍBA
Pró-Reitoria Para Assuntos do Interior
Coordenação Setorial de Pós-Graduação
Rua Aprigio Veloso, 882 - Tel (083) 321-7222-R 355
58.100 - Campina Grande - Paraíba

1. INTRODUÇÃO

Tem se verificado atualmente que os maiores custos associados à área de processamento de dados prendem-se ao desenvolvimento e manutenção de software, e que os custos com hardware têm caído notavelmente. Em face disto, torna-se cada vez mais necessário se dispor de linguagens como PROLOG (PROgramming in LOGic), consideradas de altíssimo nível (very high level programming languages) e que tentam reduzir em muito o trabalho do programador.

A linguagem PROLOG foi implementada originalmente na Universidade de Marseille (Rousseal - 1972) como uma ferramenta prática para Programação em Lógica (Logic Programming) (Kowalsky - 1974). Desde então, outras implementações vêm surgindo, inclusive compiladas (Warren - 1977), corrotinadas (Clark et alii - 1982), concorrentes (Hogger - 1982), etc. A linguagem vem se popularizando em todo o mundo, e tudo parece indicar que a tendência é uma popularização cada vez maior, notadamente sabendo-se que já se anuncia o "Computador de Quinta Geração", em fase de desenvolvimento no Japão, tendo PROLOG como linguagem básica.

Existe, porém ainda muito a pesquisar em PROLOG. Até quanto às maneiras de implementar PROLOG, cada implementação é algo diferente da outra, como diz Mellish (1981):

> "...no two Prolog systems are completely alike. This situation is unlikely to change very quickly, because new ideas and improvements for Prolog implementation are constantly being thought up."

No Brasil já várias universidades pesquisam a linguagem PROLOG. O interesse pela linguagem tem crescido enormemente nos últimos tempos, notadamente naqueles segmentos ligados à Inteligência Artificial e áreas afins.

Quando se trata de implementar PROLOG, a principal dificuldade é a pouca profundidade com a qual é tratado o assunto nas obras disponíveis. Em geral tais obras prendem-se quase que apenas em detalhar a linguagem, suas vantagens e aplicações.

No presente trabalho é dado ênfase à implementação, pois uma de suas principais metas é de servir de base para futuras pesquisas na área de desenvolvimento de interpretadores PROLOG. Este fato justifica ter-se despendido a maior parte do tempo na fase de projeto do interpretador, buscando torná-lo de fácil entendimento, manutenção e ampliação. Foi com este intuito que em maio de

1984, após mais de quatro meses de trabalho, decidiu-se abandonar quase que totalmente o trabalho realizado até ali e adotar a metodologia de Constantine.

Outra meta a que se propõe o presente trabalho é a de servir como ferramenta para Programação em Lógica. Foi esta meta que levou a uma nova mudança: O interpretador estava sendo implementado na linguagem C, no PDP-11/UNIX. No segundo semestre de 1984 passou a funcionar o IBM 4341 adquirido pela UFPb. Como grande parte das aplicações da linguagem PROLOG consome uma boa quantidade de memória, não disponíveis no PDP-11, decidiu-se, em novembro de 1984, abandonar a versão C, e passar a implementar em Pascal, no IBM 4341. Isto exigiu entre outas coisas desenvolver novos analisadores léxico e sintático, já que os anteriores estavam sendo desenvolvidos com o uso do LEX e do YACC disponíveis apenas no UNIX.

Devido ao fato de já existirem várias obras sobre programação PROLOG (vide, por exemplo Mellish - 1981), a linguagem será apresentada superficialmente. Assim, no capítulo 2 é dada uma rápida visão da linguagem, mostrando-se alí, através de um exemplo, a sintaxe adotada. E, no capítulo 3, são apresentados os predicados e funções embutidas.

No capítulo 4 são apresentados os conceitos teóricos fundamentais. No 5 é mostrado informalmente o funcionamento do interpretador, introduzindo-se o conceito de "Árvore de Prova com estruturas compartilhadas". No 6 apresenta-se a implementação onde buscou-se um nível de detalhamento e clareza tal, que permita a futuros pesquisadores modificar e ampliar o interpretador sem grandes dificuldades. Finalmente, no capítulo 7, são dados, como exemplos, alguns programas que serviram de teste para o interpretador.

2. UMA RÁPIDA VISÃO DA LINGUAGEM PROLOG

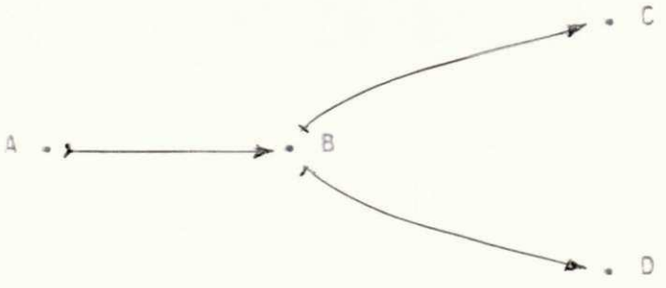
Programar em PROLOG significa introduzir uma sequência de fatos, regras e perguntas na forma de cláusulas. Tome-se o problema de achar o caminho entre dois vértices de um grafo:

Serão consideradas como regras:

a) Um caminho entre os vértices vi e vj de um grafo, será o Arco (vi,vj), se este arco existir.

b) Um caminho entre os vértices vi e vj de um grafo, será uma sequência de arcos, iniciada pelo Arco (vi, vk) e continuada pelo caminho (vk, vj) se existirem estes arco e caminho.

Será considerado, como exemplo, o grafo:



Passaremos agora ao programa PROLOG, onde a cada regra ou fato, coresponderá uma cláusula.

Uma cláusula, indica a existência de uma relação entre zero ou mais termos. Assim:

```
ExisteArco(A,B).
```

indica que entre "A" e "B" a relação "ExisteArco" é verdadeira. A relação dá-se o nome de predicado, e aos termos entre parênteses, de argumentos.

O nome de um predicado é um conjunto de caracteres iniciado por letra maiúscula, ou um conjunto de caracteres qualquer entre apóstrofos, ou ainda um símbolo especial (ver

apêndice I). Um argumento pode ser uma variável, uma função ou uma constante (isto é, um número inteiro, um número real, ou um string. Ver argumento tipo string no ítem 3.3).

Uma variável é um conjunto de caracteres iniciado por letra minúscula ou "underscore". Uma função, por sua vêz é um termo sintaticamente semelhante a um predicado, isto é, constituído pelo nome da função, com as mesmas características sintáticas que um predicado, seguido por zero ou mais argumentos entre parênteses e separados por vírgulas.

Na cláusula acima, "A" e "B" são funções de zero argumentos.

As cláusulas:

```
ExisteArco (A, B).
ExisteArco (B, C).
ExisteArco (B, D).
```

representam os fatos que descrevem o grafo dado.

Notemos que toda cláusula termina com um ponto (seguido por espaço branco, ou nova linha).

As regras podem ser representadas pelas cláusulas:

```
Caminho (vi, vj, Arco(vi, vj)) <- ExisteArco (vi, vj).
Caminho (vi, vj, Seq (Arco (vi, vk), camRest))
      <- ExisteArco (vi, vk) & Caminho (vk, vj, camRest).
```

onde o símbolo "<-" (ler "SE") indica que o predicado à esquerda deste símbolo será verdade se a conjunção de todos os predicados que se encontram à direita deste símbolo, separadas pelo simbolo "&" (ler "E"), for verdade.

O predicado situado à esquerda do símbolo "<-" é denominado "cabeça da cláusula", enquanto que a conjunção de predicados que ficam à direita deste símbolo é chamada de "corpo da cláusula". Cada predicado que constitui o corpo da cláusula é dito ser uma "submeta da cláusula".

Devemos notar que cláusulas tais como aquelas com as quais representamos os fatos de nosso problema não têm

corpo, ou seja, são compostas apenas pelas suas cabeças. E ainda, que vi, vj, vk e caminhoRest são variáveis, pois iniciam por letra minúscula.

O escopo de uma variável está limitado à cláusula onde a variável aparece. Assim, por exemplo, a variável vi aparece nas duas cláusulas com as quais representamos as regras de nosso problema, porém o vi de uma cláusula nada tem a ver com o vi da outra cláusula.

Todas as cláusulas mostradas até agora são "cláusulas definidas". Cláusulas deste tipo ao serem reconhecidas pelo interpretador PROLOG são armazenadas na estrutura de dados que forma o programa PROLOG propriamente dito. Falta-nos agora formular perguntas, tais como:

"Existe um caminho entre os vértices (A) e (D) do grafo dado? Qual"?

Esta pergunta poderá ser formulada através da cláusula:

```
? Caminho (A, D, caminhoProcurado).
```

que é denominada de "cláusula meta".

Uma cláusula meta é uma cláusula que não tem cabeça, ou seja é constituída apenas pelo corpo. Ao ser reconhecida, o interpretador ativa o resolutor de cláusulas metas, que, fazendo uso da estrutura de dados na qual estão todas as cláusulas definidas reconhecidas até o momento, busca uma solução através de uma resolução LUSH (ver capítulo 4).

3. PREDICADOS E FUNÇÕES EMBUTIDAS

3.1. Introdução

Predicados e funções embutidas são fornecidas para que se consiga obter certos resultados que seriam difíceis ou até mesmo impossíveis de serem obtidos com cláusulas e funções definidas pelo usuário. Na versão atual, por se tratar de um interpretador básico, foram colocados apenas aqueles considerados indispensáveis. Assim, os predicados embutidos resumiram-se ao predicado corte (!) (indispensável, entre outras coisas, para definição do "Nao") e dos predicados "Impr" e "ImprFalha", de fundamental importância, já que nenhum resultado será retornado para o usuário a não ser através destes dois predicados.

No que se refere às funções embutidas, foram implementadas apenas as funções aritméticas de soma (+), subtração (-), multiplicação (*), divisão inteira (:), divisão real (/), resto da divisão (Mod), a função sinal (Sinal), e a função cálculo (?). Esta última servindo para ativar as demais. Se qualquer das funções acima não figurar como argumento de uma função cálculo, será tratada como uma função definida pelo usuário, portanto não calculada.

3.2. O Predicado Corte (!)

O predicado corte é usado como uma submeta de uma cláusula que será verdade na primeira tentativa. Porém se ocorrer um retrocesso para esta submeta ela falhará e provocará a falha imediata da cláusula e da submeta que foi instanciada como esta cláusula, mesmo que existam outras cláusulas capazes de instanciar esta submeta.

Um dos principais usos do corte é na definição do predicado "Nao", através das cláusulas:

```
Nao(x) <- x & ! & Falha.
Nao(x).
```

Onde "Falha" é uma submeta que não deve ser definida como cláusula para que falhe sempre.

Deve-se notar que, se ao ser tentada a primeira cláusula, a submeta "x" falhar, a submeta "!" não terá sido tentada. Portanto, no retrocesso, será tentada a segunda cláusula, que será verdade de imediato. Se, ao contrário, x for verdade, corte será verdade, porém como a próxima submeta falha incondicionalmente o retrocesso para "!" provocará a falha imediata de "Nao(x)".

3.3. Os predicados Impr e ImprFalha

O predicado "Impr" é usado como uma submeta que escreve o valor de seus argumentos e é verdade na primeira tentativa, porém falha e nada escreve no retrocesso. Enquanto que o predicado "ImprFalha" é verdade na primeira tentativa, mas nada escreve. Porém, no retrocesso, escreve o valor de seus argumentos e falha. Estes predicados aceitam qualquer número de argumentos. Tais argumentos podem ser de qualquer tipo, incluindo o tipo string, só aceito por estes dois predicados. Um argumento tipo string é um conjunto de caracteres quaisquer, delimitado por aspas. Quando algum destes dois predicados encontram um contra-barra em um string, passa a escrever os caracteres seguintes na próxima linha.

O programa do caminho entre dois vértices de um grafo, apresentado no capítulo anterior, embora correto, está incompleto. Da forma como está escrito, nenhum resultado será retornado para o usuário. Um programa completo para aquele problema poderia ser escrito como:

```
ExisteArco(A, B).
ExistaArco(B, C).
ExisteArco(B, D).
Caminho(vi, vj, Arco(vi, vj)) <- ExisteArco(vi, vj).
Caminho(vi, vj, Seq(Arco(vi, vk), camRest))
                        <- ExisteArco(vi, vk)
                           & Caminho (vk, vj, camRest).
AchaCaminho(vi, vj) <- ImprFalha (
                        "\Nao encontrado caminho ", vi, vj)
                       & Caminho(vi, vj, camProcur)
                       & Impr("\Caminho encontrado = ",
                                                camProcur).
?AcheCaminho(A, D).
```

```
?AcheCaminho(C, D).
```

A resposta produzida por este programa será:

```
Caminno encontrado = Seq(Arco(A, B), Arco(B, D))
Nao encontrado caminho CD.
```

3.4. FUNÇÕES EMBUTIDAS

A tabela abaixo descreve as funções embutidas disponíveis:

FUNÇÃO	TIPOS ARGS x	TIPOS ARGS y	TIPO DA FUNÇÃO	VALOR DA FUNÇÃO
+(x,y)	int	int	int	Soma de x com y
	int	real	real	
	real	int	real	
	real	real	real	
-(x,y)	int	int	int	Diferença entre x e y
	int	real	real	
	real	int	real	
	real	real	real	
-(x)	int	-	int	Complemento de 10 de x
	real	-	real	
*(x,y)	int	int	int	Produto de x com y
	int	real	real	
	real	int	real	
	real	real	real	
:(x,y)	int	int	int	Divisão inteira de x por y
/(x,y)	int	int	real	Divisão real de x por y
	int	real	real	
	real	int	real	
	real	real	real	

Mod (x,y)	int	int	int	Resto da divisão de x por y
Sinal (x)	int	-	int	-1, 0 ou 1 para x negativo,
	real	-	real	zero ou positivo, respect.

Para que qualquer das funções acima assuma o valor especificado na tabela acima, é necessário que seja o, ou faça parte do, argumento de uma função cálculo (?). Assim, por exemplo, se as variáveis x, y, z assumirem os valores de -2, 3 e 8 respectivamente, então o valor de ?(*(x,+(y,z))) será -22.

Com a ajuda destas funções pode-se definir predicados tais como os mostrados abaixo:

```
Igual(x,x).
Menor(x,y) <- Igual(?(Sinal( -(x,y))),-1).
Maior(x,y) <- Igual(?(Sinal( -(x,y))),1).
Diferente(x,y) <- Nao(Igual(x,y)).
MenorIgual(x,y) <- Diferente(?(Sinal( -(x,y))),1).
MaiorIgual(x,y) <- Diferente(?(Sinal( -(x,y))),-1).
Abs(x,?( -(x))) <- Diferente(?(Sinal(x)),-1).
Abs(x,y).
```

As cláusulas que definem os predicados "Nao", "Menor", "Maior", "MaiorIgual", "Diferente", "MenorIgual", "Igual" e "Abs", apresentadas neste capítulo, podem ser acrescentadas ao início de cada programa como um dos arquivos a serem analisados (vide apêndice V).

4. CONCEITOS TEÓRICOS FUNDAMENTAIS

Um programa PROLOG é um conjunto de cláusulas definidas. Para um programa P e uma meta M, uma derivação LUSH e' uma sequência M(0), M(1), ... de metas, tais que, M(0) = M e M(i+1) é obtida de M(i) através de um processo de substituição e unificação. Dependendo das cláusulas definidas em P, é possível obter zero ou mais derivações M(i+1) a partir de M(i). Assim, se considerarmos todas as derivações possíveis de cada meta M(i), formaremos o que chamamos de "árvore de pesquisa".

Encontrar uma solução para a meta M, significa encontrar um ramo desta "árvore" em que a meta M(n) seja vazia. Por outro lado, se, percorrida toda árvore de pesquisa, não for encontrada uma meta vazia, diz-se que M não tem solução, ou seja, a meta falhou.

Uma substituição é um conjunto :

$$s = \{x1/t1, \ldots, xn/tn\}$$

de pares "variável/termo", onde as variáveis x1, ..., xn são todas distintas, e "ti" é o "termo associado" à variável "xi".

Uma "instância" de uma expressão E e' qualquer expressão, designada por [E]s ou Es, que pode ser obtida de E, pela substituição simultânea de cada ocorrência de cada variável em E, pelo respectivo termo associado de uma substituição "s". Se nenhuma variável ocorrer em Es, então Es é chamada "instância aterrada" de E.

Uma "variante" de uma expressão E, é uma instância Es de E, onde:

$$s = \{x1/y1, \ldots, xn/yn\}$$

onde x1, ..., xn são todas as variáveis de E, e y1, ..., yn são quaisquer variáveis distintas uma das outras e de x1, ..., xn.

Uma substituição u' unifica duas expressões E1 e E2, se, e somente se, [E1]u' = [E2]u' (ou [E2]u') é a instância comum de E1 e E2 determinada por u'.

Se E1 e E2 podem ser unificadas, então ha' um "unificador mais geral" (umg) de E1 e E2 se, e somente se, todos os outros unificadores u de E1 e E2 são tais que [E1]u = [[E2]umg]s para alguma substituição s.

5. IDÉIA INFORMAL DO INTERPRETADOR

O interpretador funciona de modo que, a cada cláusula reconhecida, se ela for uma cláusula definida, armazena-a em uma estrutura de dados adequada de forma a permitir facilmente o posterior acesso à mesma, como um todo, ou a qualquer dos elementos que a constituem; ou, se for uma cláusula meta, busca uma solução construindo uma árvore de prova com estruturas compartilhadas.

Uma "árvore de prova" é uma estrutura de dados que armazena, de uma maneira não redundante, o caminho, em uma árvore de pesquisa, entre a raiz e o nó atual.

O compartilhamento de estruturas baseia-se no fato de que as variáveis são as únicas partes de uma cláusula que mudam nas diferentes instâncias da mesma, o que torna possível armazenar a parte não variável da cláusula uma única vez. Desta forma, cada instância consistirá de um vetor, representando as variáveis, denominado de contexto, e um apontador para o código da cláusula.

Chama-se de "código da cláusula" a estrutura de dados que armazena uma cláusula, onde as variáveis são substituidas por índices que identificarão sua posição no contexto de cada instância.

O contexto é, por sua vez, também formado por apontadores, cada um deles apontando para o termo associado à variável correspondente, ou para uma sequência de apontadores para aquele termo.

Esta idéia será melhor esclarecida a seguir, quando utilizaremos os diagramas de Ferguson (1981) para ilustrar as várias fases por que passa a árvore de prova na resolução do programa-exemplo que encontra, se existir, um caminho entre dois vértices de um grafo. O código das cláusulas definidas pode ser escrito como:

```
(1)  ExisteArco (A, B).

(2)  ExisteArco (B, C).

(3)  ExisteArco (B, D).

(4)  Caminho (_1, _2, Arco (_1, _2)) <- ExisteArco (_1, _2).

(5)  Caminho (_1, _2, Seq (Arco (_1, _3), _4))
          <- ExisteArco (_1, _3) & Caminho (_3, _2, _4).
```

e o código da cláusula-meta pode-se escrever como:

? Caminho (A, D, _1).

onde _1, _2, ... indicam os índices das variáveis no contexto de cada instância, não importando os nomes com os quais o usuário introduziu tais variáveis.

No diagrama de Ferguson, semi-círculos superiores representam as submetas; semi-círculos inferiores, as cabeças das cláusulas; e, cada conjunto de um semi-círculo inferior ligado a zero ou mais semi-círculos superiores, uma cláusula. Uma unificação é representada pela união de um semi-círculo superior com um semi-círculo inferior, formando um círculo completo. Este círculo completo é um nó da árvore de prova. Nos semi-círculos inferiores representamos também, quando existe, cada variável do contexto por um par de colchetes, do qual parte um arco (em linha tracejada), representando o apontador para o termo associado àquela variável. Um par de colchetes sem este arco, indica que a variável ainda não foi instanciada.

A construção do diagrama é iniciada pela criação do nó raiz (círculo completo), contendo o contexto da cláusula-meta, e ao qual estarão ligadas as submetas (semi-círculos superiores) da cláusula-meta. Esta fase do diagrama, para o exemplo considerado, é mostrado na figura 1, onde podemos notar que, como a cláusula-meta só contém uma variável, apenas um par de colchetes aparece no nó raiz.

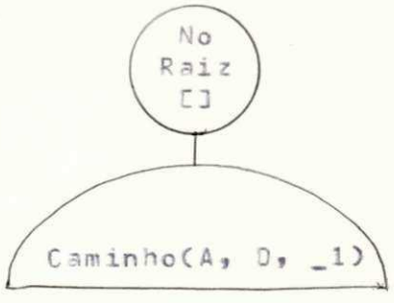


Figura 1: Fase inicial da árvore de prova:

A submeta no semi-círculo superior pode ser instanciada com a cláusula (4), obtendo-se a árvore da figura 2.

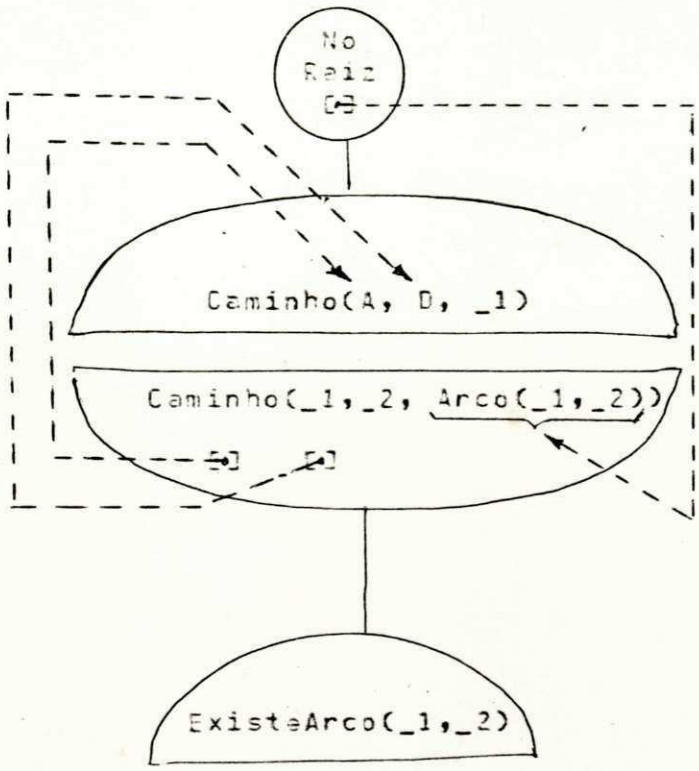


Figura 2: Árvore de prova após instanciação com a cláusula (4).

Tenta-se agora instanciar a submeta não instanciada com as cláusulas do programa. Falha-se, e retrocede-se para o estado mostrado na figura 1. Então, instancia-se a submeta com a cláusula (5), obtendo-se a árvore da figura (3).

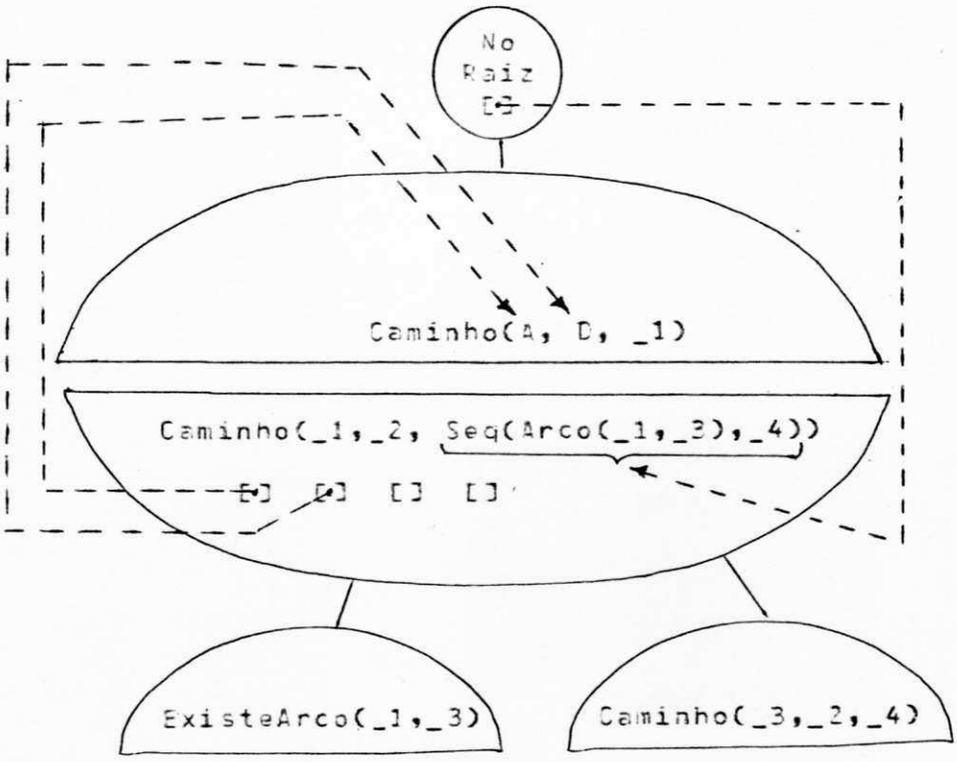


Figura 3: Árvore de prova após retrocesso para o estado da figura 1 e instanciação com a cláusula (5).

Das duas submetas não instanciadas toma-se a mais à esquerda e instancia-se com a cláusula (1), resultando na árvore de prova ilustrada na figura 4. Como a cláusula (1) não tem corpo, diz-se que a submeta foi resolvida.

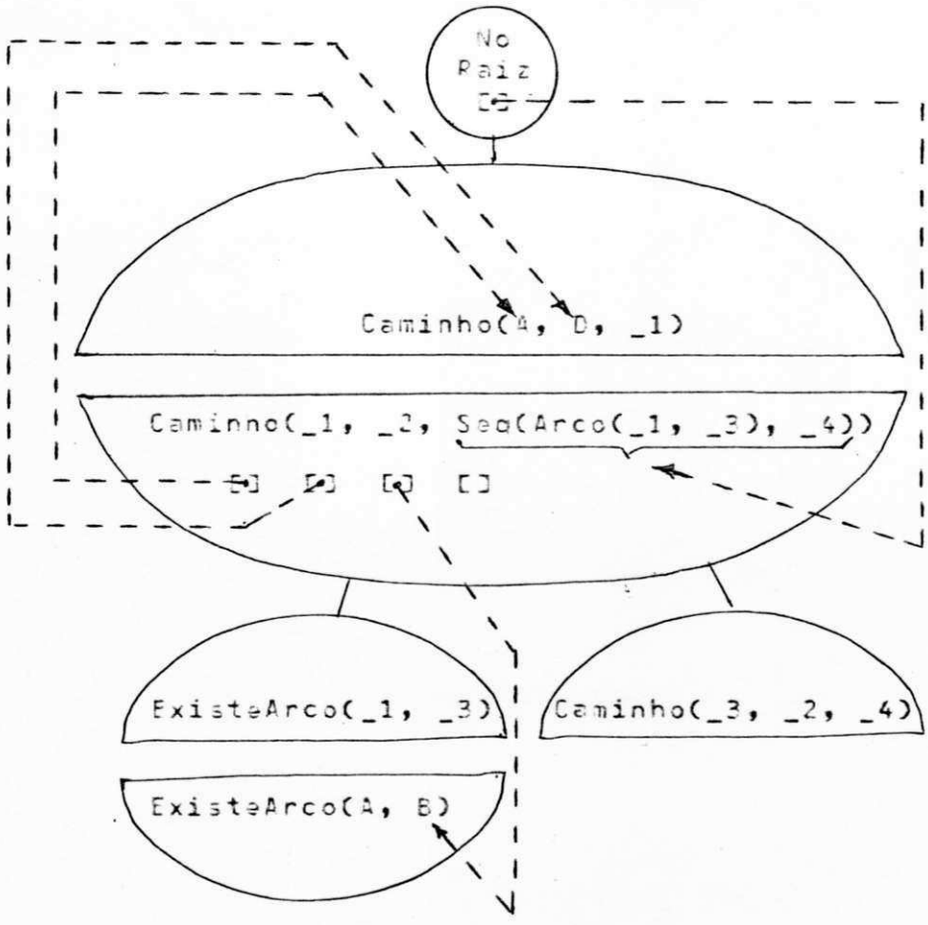


Figura 4: Árvore de prova após instanciação com a cláusula (1).

Retorna-se ao nó pai e encontra-se uma submeta não resolvida. Esta submeta é instanciada com a cláusula (4), como mostrado na figura 5.

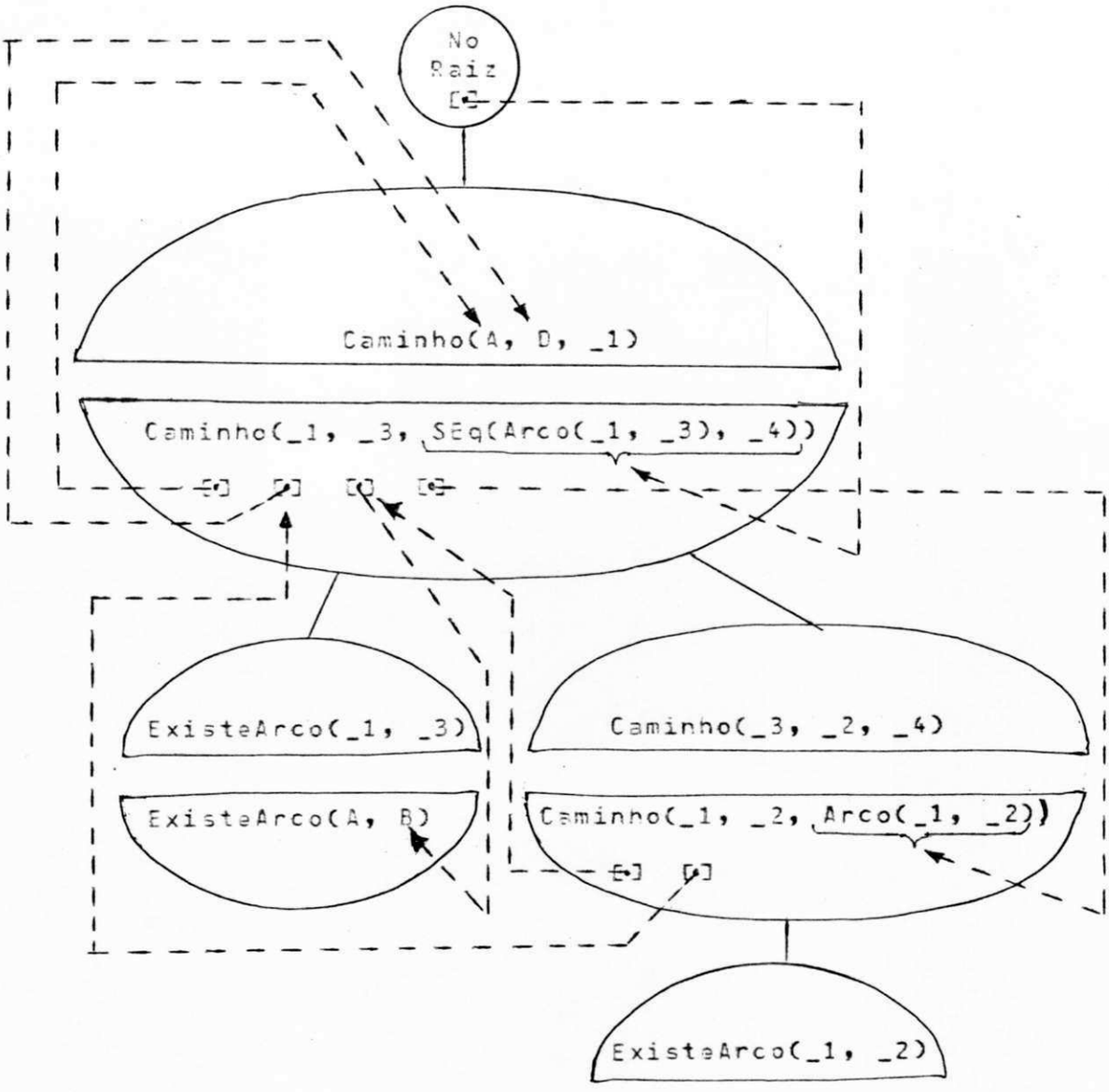


Figura 5: Árvore de prova após instanciação com a cláusula (4).

A nova submeta a ser instanciada não o pode ser com as cláusulas (1) e (2), mas o pode com a (3), como é visto na figura 6.

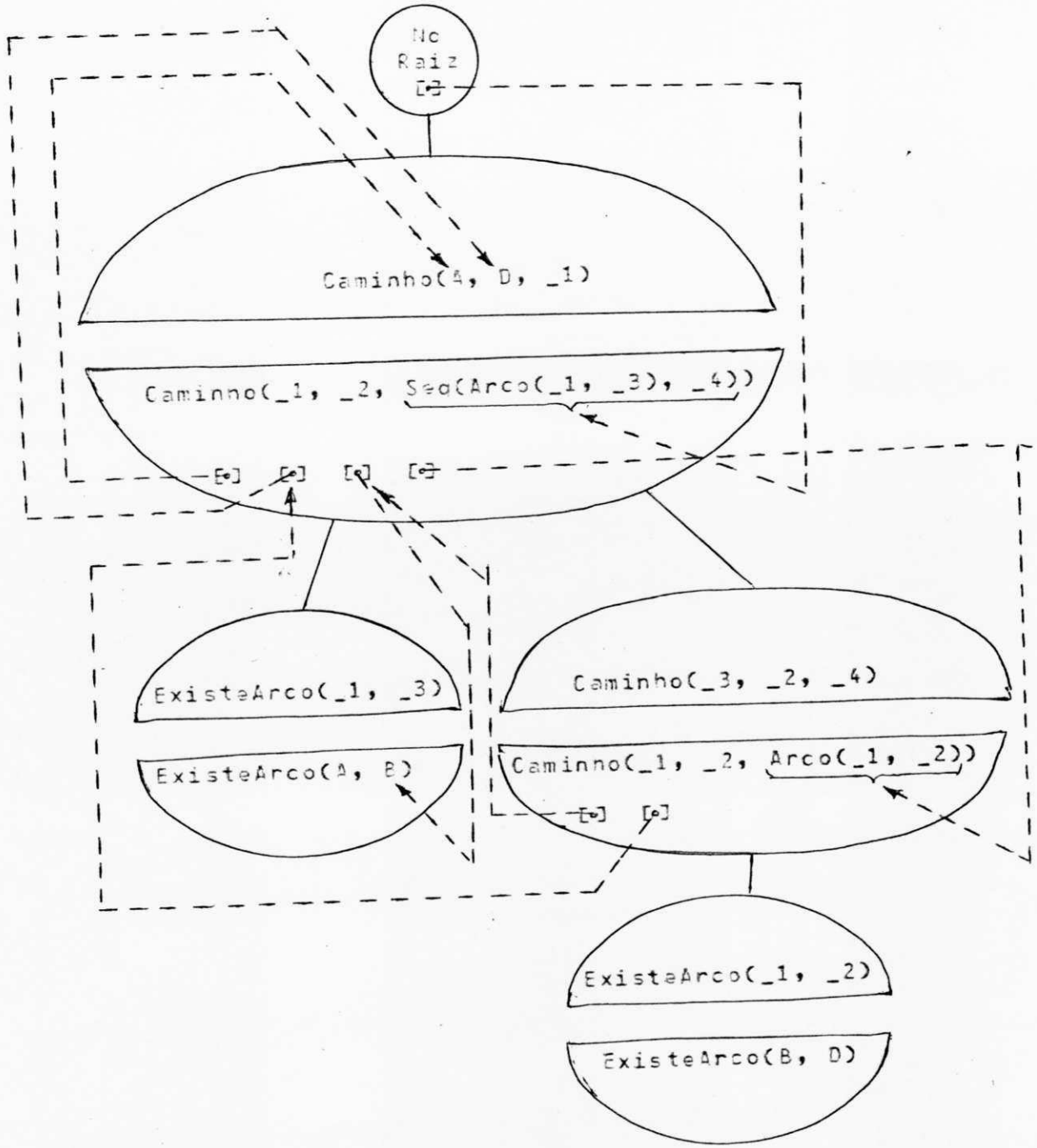


Figura 5: Arvore de prova completa, com todas as submetas resolvidas.

Retornando-se ao nó pai, não se encontram submetas a instanciar. Diz-se então que também esta submeta foi resolvida. Vai-se subindo na árvore de prova em busca de uma submeta não resolvida até chegar ao nó raiz. Chegando-se ao nó raiz e não existindo nennuma submeta a resolver, como é o caso, diz-se que a meta foi resolvida.

6. IMPLEMENTAÇÃO

6.1. INTRODUÇÃO

O Interpretador foi implementado em PASCAL, no IBM 4341, funcionando sob o sistema operacional VM/CMS. Embora compilado pelo PASCAL/VS, não foram utilizadas as extenções disponíveis naquele compilador, favorecendo-se assim a portabilidade do interpretador.

Os mecanismos descritos informalmente no capítulo 5, passam, a partir de agora, a ser descritos detalhadamente.

A estrutura de dados que permite acessar facilmente o código das cláusulas como um todo ou a qualquer das partes que a constituem, como citado no capítulo 5, será detalhada em 6.2.

Um "gráfico estruturado", constituído segundo a metodologia de Constantine (ver, por exmplo, Stevens - 1981) contendo os principais módulos do interpretador, assim como o código, em Pascal, de seu programa principal, são dados em 6.3.

Finalmente, em 6.4, é mostrado como se processa a resolução de uma cláusula meta. Ali foi usada, seguindo Bruynooghe(1980) e Ferguson(1981) a técnica de se distinguir nós determinísticos de não determinísticos. Diz-se que um nó é não determinístico quando existem cláusulas não tentadas, com mesmo nome no predicado da cabeça que a submeta instanciada no nó. Desta forma em cada retrocesso busca-se logo o nó não determinístico mais recentemente instanciado, eliminando-se de imediato, todos os nós determinísticos instanciados após o mesmo. Decorre daí não apenas uma maior eficiência, como também uma redução na área de memória utilizada pela árvore de prova, já que nos nós determinísticos não são armazenadas as informações necessárias ao retrocesso.

6.2. ESTRUTURAS DE ARMAZENAMENTO DE CLÁUSULAS

As informações referentes a cada símbolo (isto é, nome de predicado ou função) do programa são armazenados em uma tabela de símbolos (TabSimb). Os elementos dessa tabela podem ser acessados através de apontadores contidos em uma tabela Hash (TabHash). Cada elemento da tabela de símbolos contêm informações referentes a um nome de predicado ou função em estruturas:

```
;type TElemTabSimb = record ComprNome : TComprNome
                            ;Nome : TSimbolo
                            ;GpoCls : TApGpoCls
                            ;ApNomeColisor : TApElemTabSimb
                     ;end
```

onde ApNomeColisor é um apontador para uma estrutura TElemTabSimb de um símbolo que tem o mesmo índice Hash e GpoCls é um apontador para o grupo de cláusulas (estrutura TGpoCls mostrada a seguir).

Cláusulas que têm mesmo nome como predicado das cabeças são agrupadas em estruturas definidas em Pascal como:

```
;type TGpoCls = record Tipo : (Definida, Embutida)
                       ;NArgsCabec : TNArgsCabec
                       ;PrimClaus : TApClaus
                ;end
```

onde o segundo e terceiro campos referenciam o número de argumentos da cabeça e um apontador para a primeira cláusula do grupo, respectivamente.

Cada cláusula é representada numa estrutura:

```
;type TClaus = record NVars : TIndVar
                      ;Cabeca : TApPredOuFunc
                      ;PrimSMeta : TApSMOuArg
                      ;ProxClaus : TApClaus
               ;end
```

sendo "NVars" o número de variáveis da cláusula e "Cabeça", "PrimSMeta", "ProxClaus" apontadores para a cabeça, primeira submeta da cláusula e próxima cláusula do grupo, respectivamente.

Submeta ou Argumento são representados por estruturas:

```
;type TSMOuArg = record ProxMOuArg : TApSMOuArg
                       ;case Tipo:TTipoSMOuArg
                        of NInt:(ValorInteiro:
                                 integer)
                          ;NReal:(ValorReal:
                                  real)
                          ;Variavel:(IndDaVariavel:
                                     TIndVar)
                          ;PredOuFunc:(ApPredOuFunc:
                                       TApPredOuFunc)
                          ;CaracStr:(ValorCarac:
                                     char)
                 ;end
```

onde "IndDaVariavel" é o índice da variável do contexto e ApPredOuFunc um apontador para uma estrutura TPredOuFunc descrita a seguir.

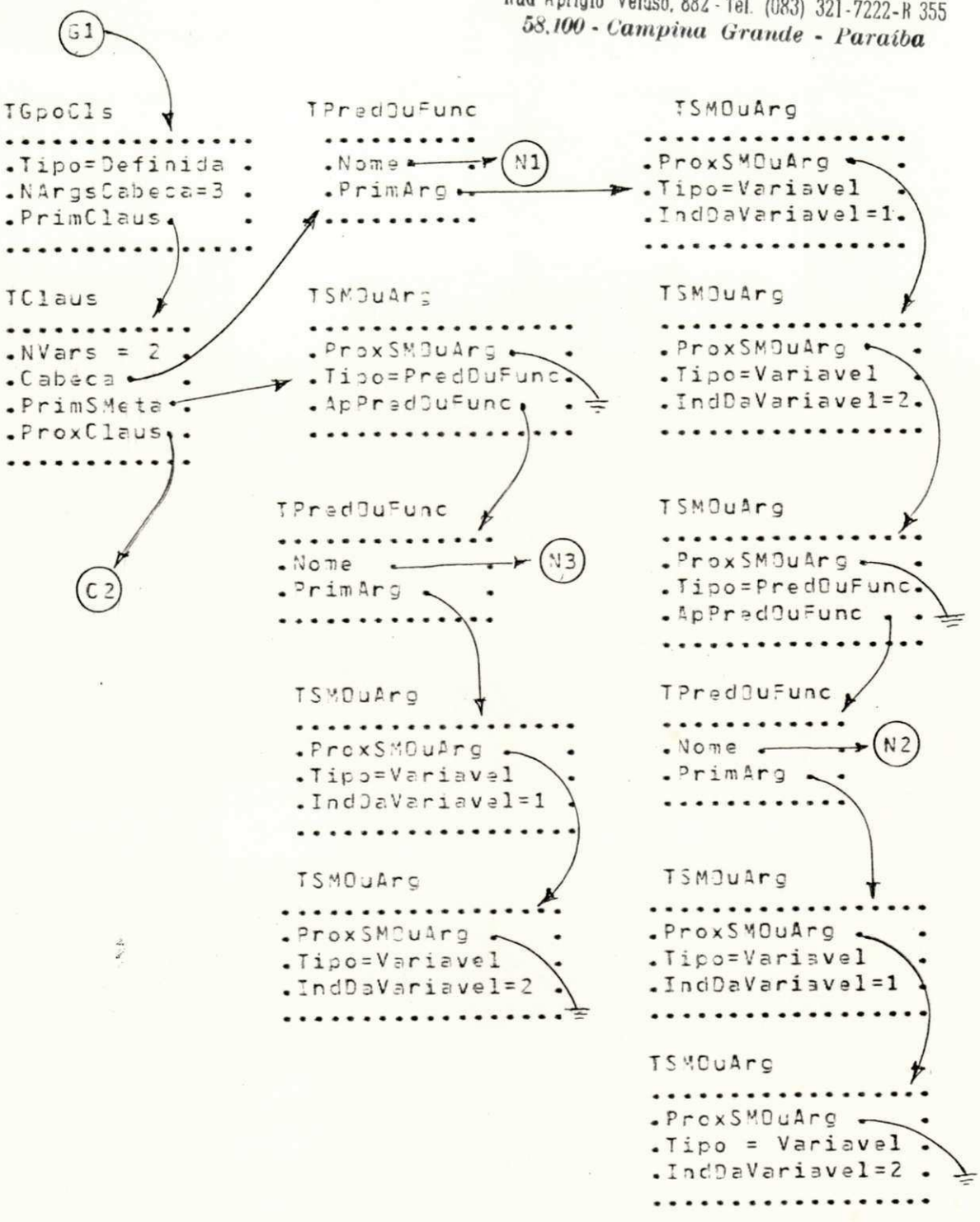
Finalmente estruturas:

```
; type TPredOuFunc = record Nome : TApElemTabSimb
                           ;PrimArg : TApSMOuArg
                   ;end
```

representam cada predicado ou função. Nome e PrimArg são, respectivamente apontadores para um elemento da tabela de símbolos e para o primeiro argumento do predicado ou função.

As estruturas descritas serão ilustradas utilizando o programa do capítulo 5. As cláusulas (1) e (2) constituirão o grupo G1 ilustrado na figura 7. As cláusulas (3), (4) e (5) um outro grupo G2, não mostrado. A estrutura da cláusula (6), por se tratar de cláusula meta, não costituirá um grupo, sendo eliminada após o processo resolutivo. Por fim, na figura 8 é ilustrada a tabela TabHash e as estruturas TElemTabSimb contendo os nomes de todos os predicados e funções do citado programa.

UNIVERSIDADE FEDERAL DA PARAÍBA
Pró-Reitoria Para Assuntos do Interior
Coordenação Setorial de Pós-Graduação
Rua Aprigio Veluso, 882 - Tel. (083) 321-7222-R 355
58.100 - Campina Grande - Paraíba



onde C2 aponta para a estrutura TClaus da cláusula 2 (não ilustrada) e N1, N2 e N3 para estruturas TElemTabSimb ilustrada na fig. 8.

Figura 7: Estruturas de um grupo de cláusulas (grupo G1 do programa-exemplo).

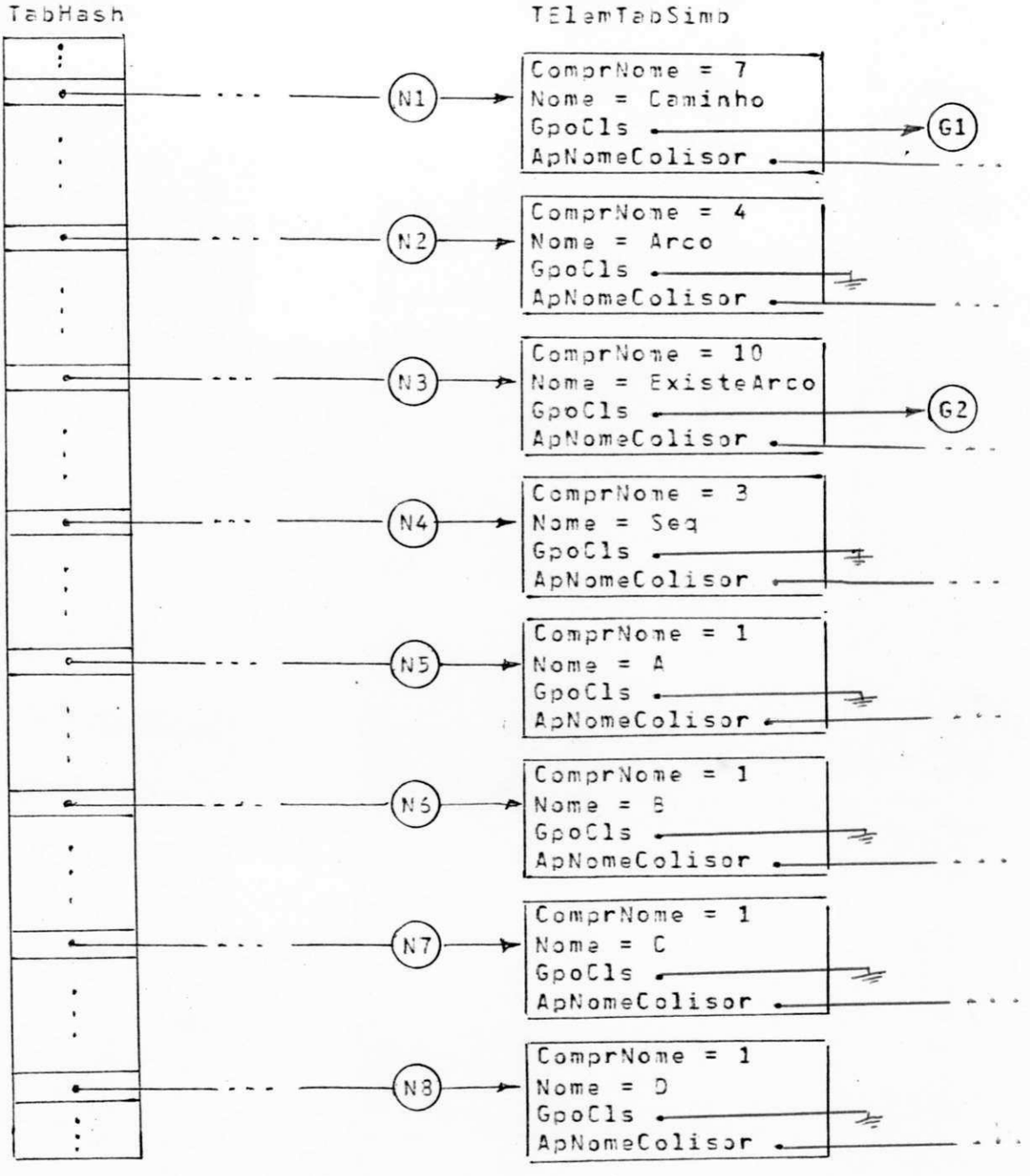


Figura 8: Tabela hash e estruturas TElemTabSimb do programa-exemplo.

6.3. O PROGRAMA E SUA ESTRUTURA

O programa foi desenvolvido seguindo, embora de uma maneira não rigorosa, a metodologia de Constantine. Foram admitidos, por exemplo, o uso de algumas variáveis globais, devido a características próprias da natureza do programa, onde tais variáveis são utilizadas em quase todos os módulos. Os principais módulos do gráfico estruturado resultante é dado na figura 9.

Segue-se o código do programa principal, em Pascal:

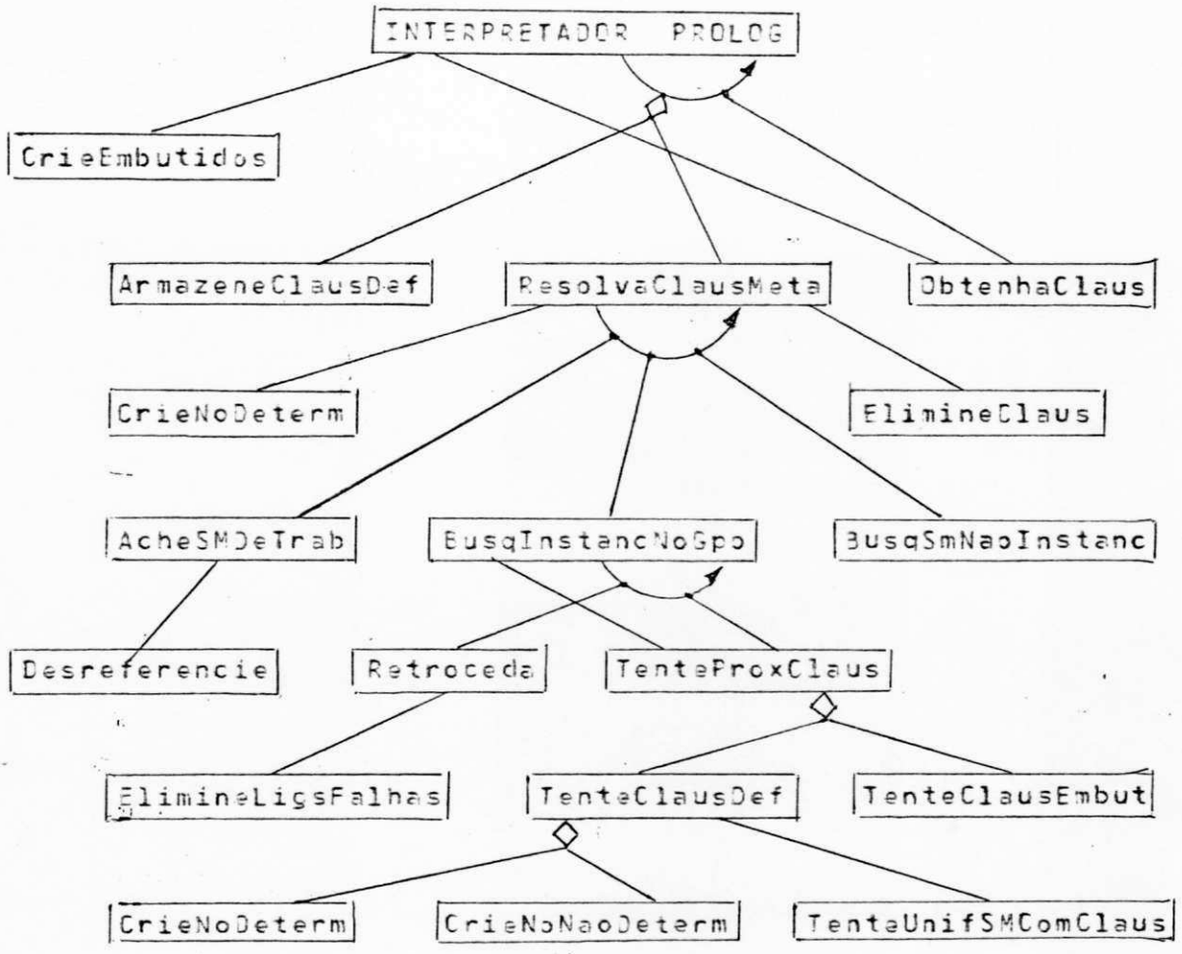
```
;begin for IndTabHash := 0 to IndMaxTabHash
       do  TabHash [IndTabHash] := nil
       ;CrieEmbutidos
       ;ObtenhaClaus (Claus)
       ;while Claus <> nil
              do begin if Claus  -> .Cabeca  <> nil
                          then ArmazeneClausDef (Claus)
                          else ResolvaClausMeta (Claus)
                       ;ObtenhaClaus (Claus)
                 ;end
 ;end
```

O módulo "CrieEmbutidos" cria na tabela de símbolos as estruturas contendo os nomes dos predicados e funções embutidas.

"ObtenhaClaus" analisa sintaticamente o arquivo fonte formando a estrutura da cláusula reconhecida já na memória do computador. Por facilidade, a análise utiliza a técnica "descendente recursiva" (ver, por exemplo, GRIES - 1971). Por se tratar de uma técnica amplamente conhecida, não serão detalhados estes módulos. Porém poderão ser vistos na listagem do programa fonte, dada no apêndice III. No apêndice I é dada, em BNF modificada, a gramática utilizada.

"ArmazeneClausDef" toma a estrutura de uma cláusula definida, obtida por "ObtenhaClaus", e a armazena numa estrutura "TGpoCls".

Finalmente, "ResolvaClausMeta" toma a estrutura de uma cláusula meta obtida por "ObtenhaClaus" e executa o processo resolutivo. Ao final deste processo, elimina essa estrutura.



CONVENÇÕES

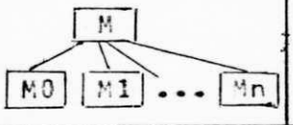
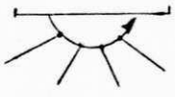
	Ativação dos módulos M1,M2,...,Mn, mesma ordem, pelo módulo M.
	Ativação de módulos em loop.
	Ativação de módulos condicionalmente.

FIGURA 9. Gráfico Estruturado com os principais módulos do programa.

6.4. A RESOLUÇÃO DE CLÁUSULAS METAS

6.4.1. INTRODUÇÃO

O processo de resolução de Cláusula Meta, como foi dito, utiliza o conceito de árvore de prova com estruturas compartilhadas. Por questão de eficiência e facilidade de implementação, todos os elementos constituintes da árvore de prova são armazenados no array "ArvProva" definido como:

```
;var ArvProva: array [0..ComprArvProva] of TElemArv
```

onde TElemArv poderá ser um inteiro representativo de uma posição de ArvProva, um apontador para uma submeta ou argumento, ou um apontador para uma cláusula, conforme definido abaixo:

```
;type TElemArv = record case TTipoElemArv
                        of TipoPosicaoArv:(PosicaoArv:
                                             TPosicaoArv)
                          ;TipoApSMOuArg:(ApSMOuArg:
                                            TApSMOuArg)
                          ;TipoApClaus:(ApClaus:
                                          TApClaus)
                 ;end
```

Fisicamente um nó da árvore de prova é uma posição de ArvProva a partir da qual, através de funções (vide "Funções de ArvProva" no Apêndice II) que são usadas como índices do array, são calculadas posições de elementos que contêm informações referentes ao nó. Na figura 10 são mostradas como e quais são estas informações referentes a um nó determinístico e não determinístico.

NÓ DETERMINISTICO

PaiDoNo		
SMetaDoNo		
CONTEXTO	InstDoContex (Var 1)	CodInst ContexInst
	InstDoContex (Var 2)	CodInst ContexInst
	.	
	.	
	.	
	.	
	InstDoContex (Var n)	CodInst ContexInst

NÓ NÃO DETERMINISTICO

PtoRetornoNo		
ProxClausNo		
BaseLgElimNo		
PaiDoNo		
SMetaDoNo		
CONTEXTO	InstDoContex (Var 1)	CodInst ContexInst
	InstDoContex (Var 2)	CodInst ContexInst
	.	
	.	
	.	
	.	
	InstDoContex (Var n)	CodInst ContexInst

FIGURA 10 : Estruturas dos nós determinísticos e não determinísticos.

Cada nó da árvore de prova corresponde a uma instanciação de uma submeta com uma cláusula. Esta instanciação será falha (e, consequentemente, o nó) se não for possível unificar cada argumento da submeta com o correspondente da cabeça da cláusula, ou quando alguma submeta da cláusula falhar.

Uma submeta falhará se não houver um grupo de cláusulas com o qual se possa instanciar a submeta (o campo "GpoCls" da estrutura "TElemTabSimb" apontada por "Nome" da estrutura de predicado da submeta for nulo) ou se a instanciação falhar e o nó for "determinístico".

Os nós são armazenados em ArvProva sequencialmente, em ordem de criação, a partir da posição de menor índice do array. Isto facilita a eliminação de nós no retrocesso, já que todos os nós a serem eliminados estarão em posições posteriores ao nó de retrocesso. Assim, para eliminar estes nós, basta atribuir à variável "PosicaoLivre" (que armazena a próxima posição livre para armazenamento de nós) o índice da posição ocupada pelo nó de retrocesso, tendo dele sido extraídas as informações úteis.

Retroceder, porém não significa apenas eliminar nós a partir do nó retrocesso. É necessário também eliminar as intanciações resultantes da criação destes nós mesmo que estas instanciações tenham sido sobre variáveis em contexto de outros nós.

As posições das instanciações passíveis de serem eliminadas, sem que o nó aonde estão o sejam, estão registradas numa pilha (pilha de ligações eliminâveis) armazenada em ArvProva, com início na posição mais elevada do array.

6.4.2. O PROCESSO RESOLUTIVO

O processo resolutivo tem início quando o programa principal ativa o módulo "ResolvaClausMeta", passando como parâmetro "ClausMeta".

"ClausMeta" é um apontador para a estrutura "TClaus" da cláusula Meta. Todas as outras estruturas necessárias ao processo resolutivo são acessadas seguindo-se apontadores a partir desta estrutura.

6.4.2.1. O Módulo ResolvaClausMeta

O módulo "ResolvaClausMeta" inícia pela criação do Nó Raiz. Este é um nó determinístico com os campos "PaiDoNo" e "SubmetaDoNo" nulos. Contém porém o contexto das variáveis que constituem a cláusula meta. A primeira submeta (a mais à esquerda) da cláusula meta é tomada como submeta atual.

A submeta atual, apesar de ser sempre a submeta que se armazena em "ArvProva " como "SubmetaDoNo", nem sempre é a submeta de trabalho (submeta sobre a qual se passará a trabalhar para encontar uma instância). Se a submeta atual for uma "meta-variável, a submeta de trabalho será a submeta resultante da desreferenciação da submeta atual. Desreferenciar uma variável significa buscar um apontador para termo associado (se houver) s seguindo o contexto da instância. Mais detalhes sobre o processo de desreferenciação serão vistos posteriormente.

A seguir tenta-se resolver a cláusula meta. Isto implica em buscar iterativamente uma submeta (atual) vazia, significando a inexistência de submetas a resolver. Se todas as possibilidades de instanciação e retrocesso de uma submeta forem esgotadas conclui-se que não é possível encontrar uma solução.

O código do programa é dado abaixo:

```
;procedure ResolvaClausMeta (var ClausMeta : TApClaus)

 ;var ArvProva : array (.0..ComprArvProva.) of TElemArv
     ;PosicaoLivre
     ,TopoPilhaLigsElim
     ,ContexSMAtual
     ,ContexSMTrab
     ,ContexClsSendoTent
     ,PaiSMAtual
     ,NoRetrocesso      : TPosicaoArv
     ;ClausSendoTentada
     ,ProxClausATentar  : TApClaus
     ;SMAtual
     ,SMDeTrab          : TApSMOuArg
     ;UnifPossivel      : boolean

 ;begin PaiSMAtual := 0
       ;SMAtual := nil
       ;PosicaoLivre := 1
       ;TopoPilhaLigsElim := ComprArvProva
       ;ClausSendoTentada := ClausMeta
       ;ProxClausATentar := nil
       ;CrieNoDeterm /* Crie No Raiz, retornando
                      * em PaiSMAtual */
       ;SMAtual := ClausMeta -> . PrimSMeta
       ;ContexSMAtual := ContexDoNo(PaiSMAtual)
       ;NoRetrocesso := 0
       ;UnifPossivel := true
       ;while (SMAtual <> nil ) and UnifPossivel
          do begin AcheSMDeTrab
                   ;BusqInstancNoGpo
                   ;BusqSMNaoInstanc
```

```
            ;end
999:  /* Ponto se Recuperacao de Erros: Sempre que
       * for detectado um erro durante a resolucao
       * de uma clausula meta, o processamento sera
       * desviado para este ponto.
      */
      ;ElimineClaus (ClausMeta)
 ;end
```

6.4.2.2. O módulo AcheSMDeTrab

Este módulo acha a submeta de trabalho. A submeta de trabalho é a própia submeta atual quando esta não é uma meta-variável. Caso contrário é a submeta do processo de desreferenciacão da submeta atual.

O código deste módulo é dado abaixo:

```
;procedure AcheSMDeTrab

  ;begin SMDeTrab := SMAtual
        ;ContexSMTrab := ContexSMAtual
        ;Desreferencie (SMDeTrab, ContexSMTrab)
        ;case SMDeTrab -> .Tipo
         of NInt, NReal: begin writeln
                              ;writeln (
 'ERRO NA RESOLUCAO DE CLAUSULA META: Submeta nao pode
                                      ser um numero' )
                              ;goto 999
/* Desvio para o ponto de recuperacao de erros
 * na rotina ResolvaClausMeta
 */
                         ;end
           ;Variavel: begin writeln
                             ;writeln (
 'ERRO NA RESOLUCAO DA CLAUSULA META: Meta-Variavel nao
      instanciada foi obtida  na  clausula  meta ou em
      alguma clausula utilizada na resulucao da  mesma.')
                             ;goto 999
 /* Desvio para o ponto de recuperacao de erros
  * na rotina ResolvaClausMeta
  */
                        ;end
           ;PredOuFunc:
```

```
      ;end
  ;end
```

6.4.2.3. O Módulo BusqInstancNoGpo

Este módulo busca uma instância para a submeta de trabalho, dentre as cláusulas do grupo de cláusulas da submeta.

O código deste módulo é mostrado abaixo:

```
;procedure BusqInstancNoGpo

 ;var GpoAtual

 ;procedure TenteProxClaus
   ;begin if SMDeTrab ->. ApPredOuFunc ->. Nome ->. GpoCls
                                  ->. TipoPredCabec <> PredDef
           then TenteClausEmbut
           else TenteClausDef
    ;end

 ;begin GpoAtual := SMDeTrab -> .ApPredOuFunc -> .Nome ->.
                                                   GpoCls
       ;if GpoAtual = nil
           then UnifPossivel := false
           else begin ProxClausATentar := GpoAtual ->.
                                                  PrimClaus
                    ;TenteProxClaus
                ;end
        ;while (not UnifPossivel) and (NoRetrocesso <> 0)
          do begin Retroceda
                   ;TenteProxClaus
            ;end
  ;end
```

Deve-se notar que a submeta atual e, consequentemente a de trabalho, pode ser modificada no retrocesso. Porém, conforme veremos no ítem 6.4.2.5, o módulo retroceda também atualiza estas variáveis. O módulo TenteClausDef é detalhado no ítem 6.4.2.6.

6.4.2.4. O Módulo BusqSMNaoInstanc

Este módulo pesquisa a árvore de prova, a partir do nó atual, em busca de uma submeta não instanciada.

O código deste módulo é dado abaixo:

```
;procedure BusqSMNaoInstanc

    ;begin while(SMAtual = nil) and (PaiSMAtual <> 1)
          do begin SMAtual := ArvProva [SMetaDoNo (
                        PaiSMAtual)]. ApSMDuArg ->.
                                        ProxSMDuArg
                   ;PaiSMAtual := ArvProva [PaiDoNo (
                           PaiSMAtual)]. PosicaoArv
                   ;ContexSMAtual := ContexDoNo (
                                     PaiSMAtual)
             ;end
    ;end
```

Neste módulo, o fato da submeta atual ser vazia (SMAtual = nil) indica que o nó pai não possui mais nenhuma submeta a instanciar. Ocorrendo isto, a submeta atual é atualizada para corresponder à próxima submeta do pai do nó pai. Se esta submeta ainda for vazia, todo o processo se repete até que seja encontrada uma submeta não vazia ou o nó pai seja o nó raiz (PaiSMAtual = 1).

6.4.2.5. O Módulo Retroceda

Neste módulo é feito o retrocesso para o nó retrocesso. Todos os nós e instanciações feitas após o nó retrocesso, inclusive, são eliminados.

O código deste módulo é dado abaixo:

```
;procedure Retrodeda

  ;procedure ElimineLigsFalhas (No : TPosicaoArv)

   ;var BaseElim
       ,InstAAnular : TPosicaoArv

   ;begin BaseElim := ArvProva [BaseLgElimNo(No)].PosicaoArv
         ;while TopoPilhaLigsElim < BaseElim
          do begin TopoPilhaLigsElim := TopoPilhaLigsElim+1
                   ;InstAAnular := ArvProva [
                                    TopoPilhaLigsElim]
                                      .PosicaoArv
                   ;ArvProva [CodInst (InstAAnular)]
                                     .ApSMOuArg := nil
                   ;ArvProva [ContexInst (InstAAnular)]
                                       .PosicaoArv := 0
            ;end
   ;end

  ;begin SMAtual := ArvProva [SMetaDoNo (NoRetrocesso)].
                                          ApSMOuArg
        ;PaiSMAtual := ArvProva [PaiDoNo (NoRetrocesso)].
                                           PosicaoArv
        ;ContexSMAtual := ArvProva [ContexDoNo (
                                    NoRetrocesso)]
                                       .PosicaoArv
        ;AcheSMDeTrab
        ;ProxClausATentar := ArvProva [ProxClausNo (
                                       NoRetrocesso)]
                                          .ApClaus
        ;EliminaLigsFalhas (NoRetrocesso)
        ;PosicaoLivre := PosicaoLibRetroc
                              (No Retrocesso)
        ;NoRetorno := ArvProva [PtoRetornoNo (
                           NoRetocesso)].PosicaoArv
  ;end
```

6.4.2.6. O Módulo TenteClausDef

Este módulo tenta instanciar a submeta de trabalho com a próxima cláusula (definida) a tentar. Se esta cláusula for a última do grupo, é criado um nó determinístico, caso contrário, não determinístico, para tentar a unificação. Se for possível unificar a submeta atual passa a ser a primeira submeta da cláusula sendo tentada.

O código deste módulo é dado abaixo:

```
;procedure TenteClausDef

    ;begin if ProxClausATentar = nil
              then UnifPossivel := False
              else begin ClausSendoTentada :=
                                    ProxClausATentar
                         ;ProxClausATentar :=
                                    ProxClausATentar ->.
                                          ProxClaus
                         ;if ProxClausATentar = nil
                             then CrieNoDeterm
                             else CrieNoNaoDeterm
                        ;if PosicaoLivre >=
                                    TopoPilhaLigsElim
                            then begin writeln
                                      ;writeln(
"ERRO NA RESOLUCAO DE CLAUS META : A area de
memoria reservada para a arvore de prova esgotou"
                                              )
                                      ;goto 999

/* Desvio para o ponto de recuperacao de erros
 * na rotina ResolvaClausMeta
*/
                                 ;end
                         ;ContexClsSendoTent :=
                                 ContexDoNo (PaiSMAtual)
                         ;TenteUnifSMComCls
                         ;if UnifPossivel
                           then begin SMAtual :=
                                        ClausSendoTentada
                                          ->. PrimSMeta
                                    ;ContexSMAtual :=
                                        ContexClsSendoTent
                                 ;end
                    ;end
    ;end
```

6.4.2.I. A criação de nós na árvore de prova

O nó raiz da árvore de prova é criado quando o módulo ResolvaClausMeta ativa CrieNoDeterm com SMAtual = nil e PaiSMAtual = 0. Todos os outros nós, seus descendentes, são criados quando TenteClausDef ou algum dos módulos de TenteClausEmbut ativa, conforme o caso, CrieNoDeterm ou CrieNoNaoDeterm.

Os códigos de CrieNoDeterm e CrieNoNaoDeterm são dados abaixo:

```
;procedure CrieNoDeterm

   ;var NoEmCriacao  : TPosicaoArv

   ;begin NoEmCriacao := NovoNoDeterm (PosicaoLivre)
         ;CompleteCriacaoNo (NoEmCriacao)
         ;PaiSMAtual := NoEmCriacao
   ;end
```

```
;procedure CrieNoNaoDeterm

   ;var NoEmCriacao : TPosicaoArv

   ;begin NoEmCriacao := NovoNoNaoDeterm (PosicaoLivre)
         ;ArvProva [PtoRetorno(NoEmCriacao)]
                        .PosicaoArv := NoRetrocesso
         ;ArvProva [ProxClausNo(NomeCriacao)]
                        .ApClaus := ProxClausATentar
         ;ArvProva [BaseLgElimNo(NoEmCriacao)]
                        .PosicaoArv := TopoPilhaLigsElim
         ;CompleteCriacaoNo (NoEmCriacao)
         ;PaiSMAtual := NoEmCriacao
         ;NoRetrocesso := NoEmCriacao
  ;end
```

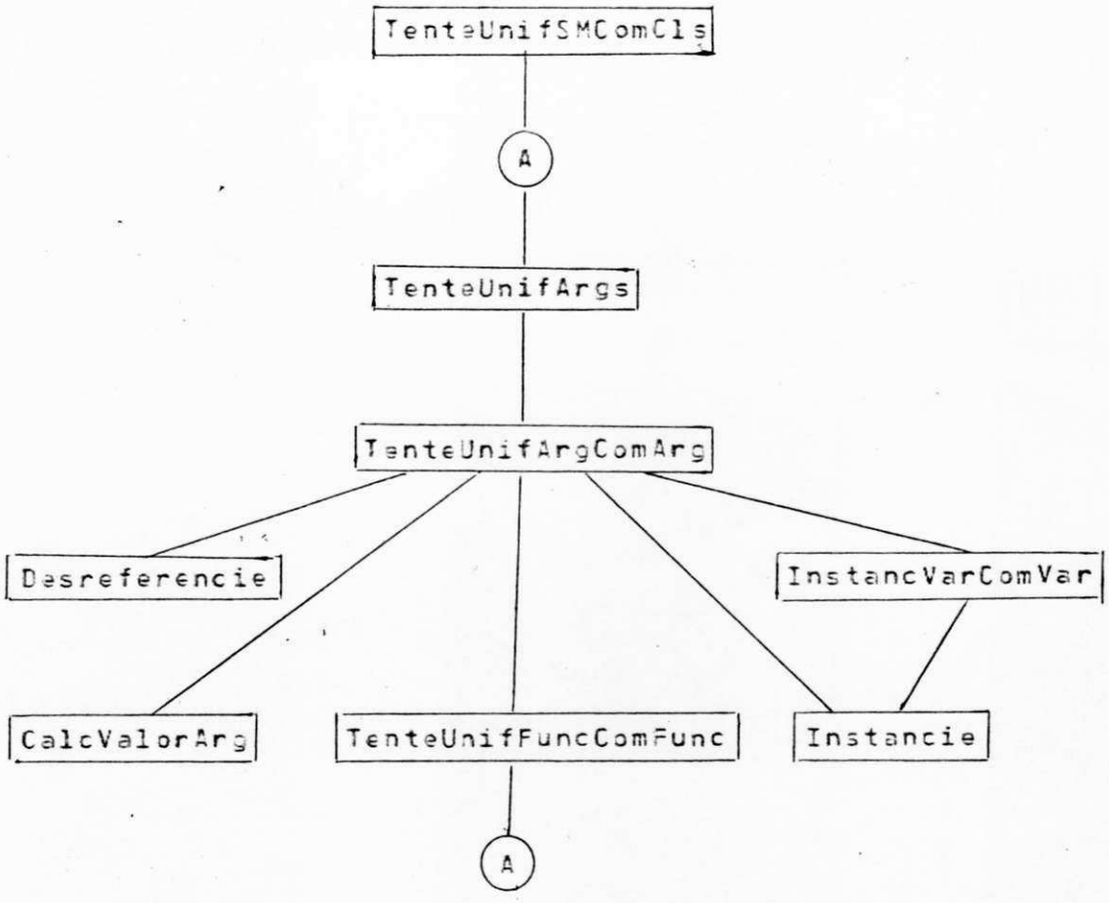
Estes dois módulos ativam CompleteCriacaoNo que lida com os campos comuns a estes dois tipos de nós, e cujo o código é dado abaixo:

```
;procedure CompleteCriacaoNo (var NoEmCriacao :
                                        TPosicaoArv )

   ;var IndVar : TIndVar
       ;Instanc : TPosicaoArv

   ;begin ArvProva [PaiDoNo(NoEmCriacao)]
                          .PosicaoArv := PaiSMAtual
         ;ArvProva [SMetaDoNo(NoEmCriacao)]
                              .ApSMOuArg := SMAtual
         ;Instanc := InstDoContex(ContexDoNo(
                               NoEmCriaco),1)
         ;for IndVar := 1 to ClausSendoTentada
                                        ->. NVars
           do begin ArvProva [CodInst(Instanc)]
                                       .ApSMOuArg := nil
                  ;ArvProva [ContexInst(Instanc)]
                                       .PosicaoArv := 0
                  ;Instanc := ProxInst (Instanc)
            ;end
         ;PosicaoLivre := PosicaoSegAoNo(NoEmCriacao,
                              ClausSendoTentada ->. NVars)
  ;end
```

Note que em todas as instâncias do contexto os campos CodInst e ContexInst são inicializados com "nil" e zero respectivamente, pois a instanciação se dará em etapas posteriores.

6.4.3. O Processo de Unificação e Instanciação

O processo de unificação e instanciação é iniciado com a ativação do módulo TenteUnifSMComCls. A parte do gráfico estruturado correspondente a este processo é dado abaixo :



Segue-se o código destes módulos:

```
/*
 * Procedure que tenta unificar submeta com clausula.
 */
 ;procedure TenteUnifSMComCls
 /*
  * Procedure em que, dados os argumentos de dois predicados,
  * tenta unificar os argumentos correspondentes.
  */
  ;procedure TenteUnifArgs ( PrimArg1 : TApSMOuArg
                            ;Contex1  : TPosicaoArv
                            ;PrimArg2 : TApSMOuArg
                            ;Contex2 : TPosicaoArv)
   ;forward

 /*
  *     Procedure que tenta unificar funcao com funcao.
  */
  ;procedure TenteUnifFuncComFunc (Func1, Func2 :
                            TApPredOuFunc
```

```
                                    ;var Contex1, Contex2 :
                                TPosicaoArv)

   ;begin if Func1 ->. Nome <> Func2 ->. Nome
             then UnifPossivel := false
             else if (Func1 ->. PrimArg = nil) and
                     (Func2 ->. PrimArg = nil)
                  then UnifPossivel := true
                  else TentaUnifArgs (Func1 ->. PrimArg
                                     ,Contex1
                                     ,Func2 ->. PrimArg
                                     ,Contex2       )

   ;end

/*
 * Definicao da procedure TenteUnifArgs, (definida
 * anteriormente com forward).
 */
 ;procedure TenteUnifArgs

 ;var Arg1, Arg2 : TApSMOuArg

 /*
  * Procedure que tenta unificar dois argumentos.
  */
  ;procedure TenteUnifArgComArg (Arg1, Arg2 : TApSMOuArg
                     ;Contex1, Contex2 :
                                          TPosicaoArv)
     ;var TipoCalc1, TipoCalc2 : TAritmet
         ;IntCalc1, IntCalc2 : integer
         ;RealCalc1, RealCalc2 : real

   /*
    * Procedure que instancia uma variavel ou seja, cria
    * uma ligacao entre a variavel e seu terno associado.
    */
    ;procedure Instancie (var Variav : TApSMOuArg
                         ;var ContexVar : TPosicaoArv
                         ;TermoAssoc : TApSMOuArg
                         ;ContexTAssoc : TPosicaoArv)

     ;var Indice : TIndVar
         ;Instanc : TPosicaoArv

     ;begin Indice := Variav ->. IndDaVariavel
           ;if Indice <> 0  /* Se nao for variavel anonima */
              then begin Instanc := InstancDoContex (
                         ContexVar, Indice)
                        ;ArvProva [CodInst (Instanc)].
                                 ApSMOuArg
                        ;ArvProva [ContexInst(Instanc)].
                                 PosicaoArv := ContexTAssoc
                        ;if Instanc < NoRetrocesso
```

```
                                then /* A posicao desta instancia
                                      * precisa ser guardada na
                                      * pilha de licagao a
                                      * eliminar
                                      */
                                begin ArvProva [
                                                TopoPilhaLigsElim].
                                                 PosicaoArv :=
                                                    Instanc
                                 ;TopoPilhaLigsElim:=
                                   TopoPilhaLigsElim - 1
                                ;if TopoPilhaLigsElim < =
                                                  PosicaoLivre
                                  then begin writeln
                                                ;writeln(
'Erro na resolucao de Claus Meta a area de memoria'
 reserved para a arvore de prova esgotou' )
                                                ;goto 999
 /* Desvio para o ponto de recuperacao de erros
  * na rotina ResolvaClausMeta
  */
                                          ;end
                              ;end
                    ;end
          ;UnifPossivel := true
  ;end

/*
 * Procedure que instancia uma variavel com outra
 * variavel nao inatanciada.
 */

  ;procedure InstancVarConVar(var Variav1, Variav2 :
                                          TApSMOuArg
                               ;var Contex1, Contex2 :
                                          TPosicaoArv)
    ;begin if(Variav1 ->. IndDaVariavel <> 0) and
              (Variav2 ->. IndDaVariavel <> 0)
              /* Se uma das variaveis for anonima,
               * nao e' necessario instanciar.
               */
              then if Contex1 < contex2
                     then Instancie(Variav2, Contex2
                                    ,Variav1, Contex1)
                     else if Contex1 > Contex2
                            then Instancie(Variav1, Contex1
                                           ,Variav2, Contex2)
                            else if Variav1 <> Variav2
                                   then Instancie(Variav1
                                                  ,Contex1
                                                  ,Variav2
                                                  ,Contex2)
                  ;UnifPossivel := true
    ;end
```

```
/*
 * TenteUnifArgComArg - Corpo da procedure
 */

;begin Disreferencie(Arg1, Contex1)
      ;Disreferencie(Arg2, Contex2)
      ;case Arg1 ->. Tipo
       of NInt : case Arg2 ->. Tipo
                  of NInt : UnifPossivel := (Arg1 ->.
                                       ValorInteiro =
                                  Arg2 ->. ValorInteiro )
                   ;Variavel : Instancie(Arg2, Contex2
                                       ,Arg1, Contex1)
                   ;NReal : UnifPossivel := false
                   ;PredOuFunc:
                      if TipoDeFunc(Arg2 ->. ApPredOuFunc)
                                         = FuncCalc
                        then begin CalcValorArg(Arg2, Contex2
                                  ,TipoCalc2, IntCalc2
                                  ,RealCalc2)
                                 ;if TipoCalc2 = NCInt
                                    then UnifPossivel :=
                                        (Arg1 ->.
                                         ValorInteiro
                                         = IntCalc2  )
                                   else UnifPossivel :=
                                              false
                             ;end
                        else UnifPossivel := false
                ;end
;NReal : case Arg2 ->. tipo
         of NReal : UnifPossivel := (Arg1 ->. ValorReal
                                    = Arg2 ->. ValorReal)
           ;variavel : Instancie(Arg2, Contex2
                                ,Arg1, Contex1)
           ;NInt : UnifPossivel := false
           ;PredOuFunc :
             if TipoDeFunc(Arg2 ->. ApPredOuFunc) =
                                          FuncCalc
               then begin CalcValorArg(Arg2, Contex2
                         ,TipoCalc2, IntCalc2
                         ,RealCalc2)
                        if TipoCalc2 =NReal
                          then UnifPossivel := (
                               Arg1 ->. ValorReal =
                                     RealCalc2)
                          else UnifPossivel := false
                    ;end
               else UnifPossivel := false
        ;end
       ;Variavel : case  Arg2 ->. Tipo
                   of NInt, NReal, PredOuFunc :
                      Instancie(Arg1, Contex1
                               ,Arg2, Contex2)
```

```
          ;Variavel : InstancVarComVar(Arg1, Arg2
                                ,Contex1, Contex2)
      ;end
;PredOuFunc :
     if TipoDeFunc(Arg1 ->. ApPredOuFunc <> FuncCalc
       then case Arg2 ->. Tipo
            of PredOuFunc : TenteUnifFuncComFunc
                                (Arg1 ->. ApPredOuFunc
                                ,Arg2 ->. ApPredOuFunc
                                ,Contex1, Contex2    )
             ;Variavel : Instancie(Arg2, Contex2
                                  ,Arg1, Contex1)
            ;NInt, NReal : UnifPossivel = false
           ;end
        else begin CalcValorArg(Arg1, Contex1, TipoCalc1
                               ,IntCalc1, RealCalc2)
                  ;case Arg2 ->. Tipo
                    of NInt : if TipoCalc1 := NCInt
                              then UnifPossivel := (Arg2
                                  ->. ValorInteiro =
                                      IntCalc1   )
                              else UnifPossivel := false
                      ;NReal : if TipoCalc1 = NCReal
                               then UnifPossivel := (Arg2
                                    ->. ValorReal =
                                        RealCalc1  )
                               else UnifPossivel := false
                      ;Variavel : Instancie(Arg2, Contex2
                                           ,Arg1, Contex1)
                      ;PredOuFunc :
                        if TipoDeFunc(Arg2 ->. ApPredOuFunc)
                                       = FuncCalc
                          then begin CalcValorArg(Arg2
                                    ,Contex2, TipoCalc2
                                    ,IntCalc2, RealCalc2)
                                ;if TipoCalc1 = TipoCalc2
                                   then if TipoCalc1 = NCInt
                                          then UnifPossivel
                                               := (IntCalc1
                                               = IntCalc2   )
                                          else UnifPossivel
                                               := (RealCalc1
                                               = RealCalc2  )
                                   else UnifPossivel := false
                               ;end
                          else UnifPossivel := false
                     ;end
                ;end
           ;end
  ;end
/*
 * TentUnifArgs - Corpo da procedure
 */
```

```
;begin Arg1 := PrimArg1
       Arg2 := PrimArg2
       UnifPossivel := true
       while (Arg1 <> nil) and (Arg2 <> nil) and
                                    UnifPossivel
         do begin TenteUnifArgComArg(Arg1, Arg2
                                     ,Contex1
                                     ,Contex2   )
                  ;Arg1 := Arg1 ->. ProxSMOuArg
                  ;Arg2 := Arg2 ->. ProxSMOuArg
            ;end
      ;if UnifPossivel
         then UnifPossivel := (Arg1 = Arg2)
;end

/*
 * TenteUnifSMComClaus - Corpo da procedure
 */

 ;begin if (SMDeTrab ->. ApPredOuFunc ->. PrimArg = nil)
           and (ClausSendoTentada ->. Cabeca ->. PrimArg
                                               = nil   )
         then UnifPossivel := true
         else TenteUnifArgs (SMDeTrab ->. ApPredOuFunc
                            ->. PrimArg, ContexSMDeTrab
                            ,ClausSendoTentada ->.
                            Cabeca ->. PrimArg
                            ,ContexClsSendoTent)
   ;end
```

7. CONCLUSÕES E FUTUROS TRABALHOS

O interpretador desenvolvido, apesar de já poder ser utilizado como uma ferramenta para programação em lógica, é ainda uma versão básica. É, portanto, despida de algumas facilidades disponíveis em outros interpretadores. Muitas destas facilidades poderiam ser adicionadas facilmente, porém devido ao fato de ter-se dedicado uma parcela significativa do tempo disponível para este trabalho nas fases de projeto e implementação de seus elementos básicos, visando-se as metas estabelecidas, decidiu-se implementar apenas aquelas facilidades consideradas indispensáveis. Deixa-se, assim a tarefa de implementar outras facilidades para futuros trabalhos.

O programa fonte, em Pascal, contém 2960 linhas (incluindo-se comentários), enquanto que o objeto contém 35468 bytes (não incluindo a biblioteca Pascal necessária à sua execução, e tendo o array ArvProva 10000 posições, ocupando assim, aproximadamente 20k bytes). No apêndice VI são feitas algumas considerações acerca da eficiência.

Dentre as facilidades que poderiam ser acrescidas, podemos citar:

a) Declaração de operadores:

Várias implementações PROLOG permitem ao usuário declarar operadores. Isto significa que o usuário pode definir os predicados ou funções que desejar como operadores infixos, prefixos ou sufixos, com a prioridade que desejar. Por exemplo:

```
Op(Eh,LR,200).
```

poderia definir o predicado "Eh" como operador infixo de associatividade LR (esquerda para direita) e de prioridade 200. Desta forma poder-se-iam definir cláusulas do tipo:

```
Joao Eh Estudioso.
```

equivalente a cláusula:

Eh(Joao,Estudioso).

Da mesma forma poder-se-ia definir as funções aritméticas como operadores, e usá-las nas formas:

Prefixa: (exemplo: Log 3.750);

Infixa: (exemplo: x + y * z);

Sufixa: (exemplo: 3 Fatorial).

Esta facilidade pode ser implementada fazendo-se algumas alterações nos módulos de ObtenhaClausula (p/ex: Criar módulos para lidar com precedência de operadores) e acrescentar nas estruturas TElemTabSimb mais um campo, apontando para uma estrutura que armazene as características do operador.

b) Outros predicados e funções embutidos:

Existe uma variedade de predicados e funções embutidas que poderiam ser acrescentadas ao interpretador.

Para acrescentar um novo predicado ou função embutida basta adicionar ao interpretador uma procedure de criação, ativada por CrieEmbutidos, e uma procedure de execução, ativada por TenteClausEmbut, no caso de predicado, ou por CalcFuncEmbut, no caso de função.

Além destas facilidades, seria interessante acrescentar otimizações, tais como aquelas conhecidas por "popping on deterministic sucess" e "tail end recursion", ver Bruynooughe (1980) e Warren (1980). Como separamos bem os tratamentos dos nós determinísticos daqueles dos nós não determinísticos, esperamos que estas otimizações sejam relativamente fáceis de fazer.

Acrescentadas estas facilidades e otimizações o passo seguinte seria partir em busca de novas fronteiras de pesquisa, tais como:

a) Incluir em PROLOG tipos abstratos de dados e mecanismos de encapsulação; ver LACET et alii (1984).

b) Investigar outros mecanismos de controle para PROLOG, tais como corrotinamento, data-flow, paralelismo, etc; ver Clark et alii (1982) e Hogger(1982).

c) Adaptar PROLOG para aplicações que exijam uma manipulação maciça de dados numéricos, dotando-o de estruturas que permitam manipular eficientemente grandes massas de números, tais como as do tipo array.

d) Investigar a possibilidade de otimizações, tais como a redução do espaço de memória da árvore de prova utilizado pelo processo resolutivo.

e) Criar um ambiente integrado e amigavel para programação em PROLOG; um exemplo disto é o "front-end SIMPLE" de micro-PROLOG; ver Clark e McCABE (1984).

Entretanto tais fronteiras são tão vastas que não caberiam em uma única dissertação de mestrado. Este trabalho propõe-se apenas a servir de acesso a estas fronteiras.

8. REFERÊNCIAS BIBLIOGRÁFICAS

BRUYNOOGHE, M. "The Memory Manegement of PROLOG Implementations". Workshop on Logic Programming, Debrecen, Hungary, 1980.

CLARK, K. L., McCABE F. e GREGORY, S. "IC - PROLOG Language Features". In: "Logic Programming". Clark e Tarnlund, Academic Press, 1982

CLARK, K. L. E McCABE. F "micro-PROLOG: PROGRAMMING IN LOGIC". Prentice - Hall, 1984

FERGUSON, R. J. "A PROLOG Interpreter for the UNIX Operating System". Tese de Mestrado, University of Waterloo, 1981.

GRIES, David. "Compiler Construction of Digital Computers". John Wiley & Sons Inc, 1971.

HOGGER, C. J. "Concurrent Logic Programming" Em "Logic Programming". Clark e Tarnlund, Academic Press, 1982.

KOWALSKY, R. A. "Predicate Logic as a Programming Language". IFIP 74, North-Holland 1974, p. 569-574.

KOWALSKY, R. A. "Algoritms = Logic + Control". Comm. of ACM, vol. 22, july 1979, p. 424-436.

LACET, Eraldo C. e SILVA, Helio de M. e CAVALCANTI, J. A. "Abstract Data Type and Logic Programing". In: Anais da Decima Conferência Latinoamericana de Informática. Centro Latinoamericano de Informaática, Viña Del Mar, Chile, 1984, p. 149-151.

LACET, Eraldo C. & SILVA, Hélio de M. "Projeto de um Interpretador PROLOG". II Simpósio Fluminense de Lógica, Filosofia e Teoria da Ciência. Instituto de Lógica, Filosofia e Teoria da Ciência NITERÓI -1984.

MELLISH, C. S. and CLOCKSIN, W. F. "Programming in PROLOG". Springer-Verlag Berlin Heidelburg, 1981.

ROBERTS, Grant. "An Implementation of PROLOG". Waterloo, 1977.

ROUSSEL, P. "Definition et Traitement de l'Egalite Formelle in Demonstration Automatique". Tese de 3o. ciclo, UER de Lumainy, Marseille, 1972.

STEVENS, Wayne P. "Using Structure Design". Willey Intercience, 1981.

VAN ENDEN, M. H. "An Algorithm for Implememting PROLOG". Research Report CS-81-28, Universyty of Waterloo, 1981.

WARREN, D. N. D. "Implementing PROLOG - Compiling Predicate Logic Programs". D. A. I. Research Report No. 39, University of Edimburg, 1977.

WARREN, D. N. D. "An Improved PROLOG Implementation Which Optimises Tail Recursion". Workshop on Logic Programming, Debrecen, Hungary, 1980.

APÊNDICE I

SINTAXE DA GRAMÁTICA EM BNF MODIFICADA:

<Sessao Prolog> ::= {<clausula>}*

<Clausula> ::= <Calsula Definida> | <Clausula Meta>

<Clausula Definida> ::= <Cabeca> [<- <Corpo>] <FimCls>

<Clausula Meta> ::= ? <Corpo> <FimCls>

<Cabeca> ::= <PredOuFunc>

<Corpo> ::= <Submeta> { & <Submeta>}*

<FimCls> ::= . <Branco>

<PredOuFunc> ::= <NomePredOuFunc> [(<Argumento>
{ , <Argumento>}*)]

<Submeta> ::= <PredOuFunc>| <Variavel>

<NomePredOuFunc> ::= <Maiuscula> {<Alfanum>}* |
<SimEspecial> | '{<Caracter>}+'

<Argumento> ::= <Inteiro> | <Real> | <String> | <Variavel>
| <Variavel Anonima> | <PredOuFunc>

<Inteiro> ::= [-] <Numero>

<Real> ::=[-] <Numero> . <Numero> [<Expoente>]

<Variavel> ::= <Minuscula> {<Alpanum>}* |
<Undescore> {<Alpanum>}+

<Variavel Anonima> ::= <Undescore>

<String> ::= "{<Caracter>}*"

<Caracter> ::= qualquer caracter EBCDIC

<Maiuscula> ::= aA| B | ... | Z

<Minuscula> ::= a | b | ... | z

<Numero> ::= {<Digito>}+

<Expoente> ::= (E | e) [-] <Digito> [<Digito>]

<Underscore ::= _

<Digito> ::= 0| 1 | 2 | ... | 9

<Alfanum > ::= <Maiuscula> | <Minuscula> | <Digito>

<Simb Especial> ::= + | - | * | : | \ | / | # | = |
; | ^ | ~ | $ | < | > | ! | ?

APÊNDICE II

FUNÇÕES USADAS NA MANIPULAÇÃO DE ARVPROVA

```
/* FUNÇÕES QUE, DADA A POSIÇÃO DE UMA ESTRUTURA
   CONCEITUAL, ARMAZENADA EM ARVPROV, CALCULAM  A
   POSIÇÃO DE ELEMENTOS QUE, CONCEITUALMENTE,
   FAZEM PARTE DA ESTRUTURA                      */

;Function PtoRetorno (No : TPosicaoArv ) : TPosicaoArv
   ;begin PtoRetorno := -3
   ;end

;Function ProxClausNo (No : TPosicaoArv ) : TPosicaoArv
   ;begin ProxClausNo := -2
   ;end
;Function BaseLgElimNo (No : TPossicaoArv ) : TPosicaoArv
   ;begin BaseLgElimNo := -1
   ;end

;Function PaiDoNo (No : TPosicaoArv ) : TPosicaoArv
   ;begin PaiDoNo := No
   ;end

;Function SMetaDoNo (No : TposicaoArv ) : TposicaoArv
   ;begin SMetaDoNo := No + 1
   ;end

;Function ContexDoNo (No : TPosicaoArv ) : TPosicaoArv
   ;begin ContexDoNo := No
   ;end

;Function InstDoContex (Contex : TPosicaoArv
                       ;IndVar : TIndVar) :TPosicaoArv
   ;begin InsDoContex := Contex + 2 * IndVar
   ;end

;Function ProxInst (Inst : TPoosicaoArv) : TPosicaoArv
   ;begin ProxInst := Inst + 2
   ;end

;Function CodInst (Inst : TPosicaoArv) : TPosicaoArv
   ;begin CodInst := Inst
   ;end

;Function ContexInst (Inst : TPosicaoArv) : TPosicaoArv
```

```
   ;begin ContexInst := Inst + 1
   ;end

;Function PosicoSegAoNo (No : TPosicaoArv
                         NVars : TIndVar) : TPosicaoArv
  ;begin PosicaoSegAoNo := NO + 2 * NVars + 2
  ;end

;Function PosicoaLibRetroc (No : TPosicaoArv) : TPosicaoArv
   ;begin PosicaoLibRetroc := -3
   ;end

;Function NovoNoDeterm (PosicaoLivre : TPosicaoArv)
                       : TPosicaoArv
   ;begin NovoNoDeterm := PosicaoLivre
   ;end

;Function NovoNoNaoDeterm (PosicaoLivre : TPosicaoArv)
                          : TPosicaoArv
   ;begin NovoNoNaoDeterm := PosicaoLivre +3
   ;end
```

APÊNDICE III

PROGRAMA FONTE COMPLETO

Segue-se a listagem do programa fonte. Devido ao fato da impressora disponível no sistema IBM da UFPb não ser capaz de imprimir letras minúsculas e vários caracteres especiais, todas as letras são impressas em maiúsculas e alguns caracteres não são impressos ou são impressos caracteres diferentes dos reais.

```
/**********************************************************************
 **********************************************************************

      I N T E R P R E T A D O R      P R O L O G

      DESENVOLVIDO POR ERALDO CRUZ LACET COMO TESE DE MESTRADO

 **********************************************************************
 **********************************************************************/



PROGRAM PROLOG(INPUT, OUTPUT)

/**********************************************************************
 *    PROGRAMA PRINCIPAL - CONSTANTES                                 *
 **********************************************************************/

  CONST NMAXARGSCABEC      = 15
        COMPRMAXSIMB       = 32
        NMAXVARSCLAUS      = 31
        COMPRARVPROVA      = 10000
        INDMAXTABHASH      = 127
        NMAXDIGITOS        = 9

/**********************************************************************
 *     PROGRAMA PRINCIPAL - DEFINI  O DE TIPOS                        *
 **********************************************************************/

  TYPE TCOMPRSIMB        = 0..COMPRMAXSIMB
       TNARGSCABEC       = 0..NMAXARGSCABEC
       TAPCLAUS          = -  TCLAUS
       TAPSMOUARG        = -  TSMOUARG
       TAPPREDOUFUNC     = -  TPREDOUFUNC
       TAPGPOCLS         = -  TGPOCLS
       TINDVAR           = 0..NMAXVARSCLAUS
       TINDTABHASH       = 0..INDMAXTABHASH
       TAPELEMTABSIMB    = -  TELEMTABSIMB
```

```
    TTIPOSMOUARG      = (NINT, NREAL, VARIAVEL, PREDOUFUNC, CARACSTR)
    TTIPOPRED         = (PREDWRITE,PREDWRITEFALHA,PREDCORTE,PREDDEF)
    TTIPOFUNC         = (FUNCSOMA, FUNCSUBT, FUNCMULT, FUNCDIVINT
                        ,FUNCSINAL
                        ,FUNCDIVREAL, FUNCMOD, FUNCCALC, FUNCDEF)
    TSIMBOLO          = ARRAY (.1..COMPRMAXSIMB.) OF CHAR
    TELEMTABSIMB      = RECORD COMPRNOME   TCOMPRSIMB
                               NOME   TSIMBOLO
                               GPOCLS   TAPGPOCLS
                               APNOMECOLISOR   TAPELEMTABSIMB
                        END
    TGPOCLS           = RECORD TIPOPREDCABEC   TTIPOPRED
                               NARGSCABEC   TNARGSCABEC
                               PRIMCLAUS   TAPCLAUS
                        END
    TCLAUS            = RECORD NVARS   TINDVAR
                               CABECA   TAPPREDOUFUNC
                               PRIMSMETA   TAPSMOUARG
                               PROXCLAUS   TAPCLAUS
                        END
    TSMOUARG          = RECORD PROXSMOUARG   TAPSMOUARG
                               CASE TIPO   TTIPOSMOUARG
                                 OF NINT   (VALORINTEIRO   INTEGER)
                                    NREAL   (VALORREAL   REAL)
                                    VARIAVEL   (INDDAVARIAVEL
                                                    TINDVAR)
                                    PREDOUFUNC   (APPREDOUFUNC
                                                    TAPPREDOUFUNC)
                                    CARACSTR   (VALORCARAC   CHAR)
                        END
  TPREDOUFUNC         = RECORD NOME   TAPELEMTABSIMB
                               PRIMARG   TAPSMOUARG
                        END

/*****************************************************************
 *     PROGRAMA PRINCIPAL - DECLARA  O DE VARIAVEIS              *
 *****************************************************************/

  VAR TABHASH      ARRAY (.0..INDMAXTABHASH.) OF TAPELEMTABSIMB
      CLAUS        TAPCLAUS
      ARQUIVO      TEXT
      PREDSEMBUT   ARRAY (.PREDWRITE..PREDCORTE.) OF TAPELEMTABSIMB
      FUNCSEMBUT   ARRAY (.FUNCSOMA..FUNCCALC.) OF TAPELEMTABSIMB
      VAR INDTABHASH   TINDTABHASH


/*****************************************************************
 * PROGRAMA PRINCIPAL - FUN  O QUE RETORNA O TIPO DO PREDICADO.
 *****************************************************************/

   FUNCTION TIPODOPRED ( PREDICADO   TAPPREDOUFUNC)   TTIPOPRED

     VAR NOMEPRED   TTIPOPRED
```

UNIVERSIDADE FEDERAL DA PARAÍBA
Pró-Reitoria Para Assuntos do Interior
Coordenação Setorial de Pós-Graduação
Rua Aprigio Veloso, 882 - Tel. (083) 321-7222-R 355
58.100 - Campina Grande - Paraíba

```
    BEGIN NOMEPRED  = PREDWRITE
          WHILE (NOMEPRED )  PREDDEF) AND
                (PREDICADO - . NOME )  PREDSEMBUT (.NOMEPRED.))
            DO NOMEPRED  = SUCC (NOMEPRED)
          TIPODOPRED  = NOMEPRED
    END


/*****************************************************************
 * PROGRAMA PRINCIPAL - FUN  O QUE RETORNA O TIPO DA FUN  O
 *****************************************************************/

   FUNCTION TIPODEFUNC (FUNC   TAPPREDOUFUNC)   TTIPOFUNC

     VAR NOMEFUNC   TTIPOFUNC

     BEGIN NOMEFUNC  = FUNCSOMA
           WHILE (NOMEFUNC )  FUNCDEF) AND
                 (FUNC - . NOME )  FUNCSEMBUT(.NOMEFUNC.))
             DO NOMEFUNC  = SUCC (NOMEFUNC)
           TIPODEFUNC  = NOMEFUNC
     END

/*****************************************************************
 * PROGRAMA PRINCIPAL - PROCEDURES QUE ELIMINAM CLAUSULAS E SUAS      *
 *                      SUBESTRUTURAS                               *
 *****************************************************************/

  PROCEDURE ELIMINEPREDOUFUNC (VAR APPREDOUFUNC   TAPPREDOUFUNC)
    FORWARD

  PROCEDURE ELIMINESMOUARG (VAR SMOUARG   TAPSMOUARG)

    BEGIN IF SMOUARG - . PROXSMOUARG  )  NIL
            THEN ELIMINESMOUARG (SMOUARG - . PROXSMOUARG)
          IF SMOUARG - . TIPO = PREDOUFUNC
            THEN ELIMINEPREDOUFUNC (SMOUARG - . APPREDOUFUNC)
          DISPOSE (SMOUARG)
    END

  PROCEDURE ELIMINEPREDOUFUNC

    BEGIN IF APPREDOUFUNC - . PRIMARG )  NIL
            THEN ELIMINESMOUARG (APPREDOUFUNC - . PRIMARG)
          DISPOSE (APPREDOUFUNC)
    END

  PROCEDURE ELIMINECLAUS (VAR CLAUS   TAPCLAUS)

    /* ELIMINA UMA CLAUSULA ASSUMINDO QUE, OU ELA   UMA CLAUSULA META, OU
     * UMA CLAUSULA DEFINIDA QUE N O ESTA LIGADA A UM GRUPO DE CLAUSULAS
     */
     BEGIN IF CLAUS - . CABECA )  NIL
             THEN ELIMINEPREDOUFUNC (CLAUS - . CABECA)
           IF CLAUS - . PRIMSMETA )  NIL
```

```
          THEN ELIMINESMOUARG (CLAUS - . PRIMSMETA)
        DISPOSE (CLAUS)
    END


/*****************************************************************
 * PROGRAMA PRINCIPAL - FUN  O QUE ACHA O NUMERO DE ARGUMENTOS DE UM  *
 *                      PREDICADO OU FUNCAO                          *
 *****************************************************************/

  FUNCTION ACHENARGS (PF   TAPPREDOUFUNC)   TNARGSCABEC

    VAR PROXARG   TAPSMOUARG
        NARGS   TNARGSCABEC

    BEGIN NARGS  = 0
          PROXARG  = PF - . PRIMARG
          WHILE PROXARG )  NIL
            DO BEGIN NARGS  = NARGS + 1
                     PROXARG  = PROXARG - . PROXSMOUARG
               END
          ACHENARGS  = NARGS
    END


'*****************************************************************
 * PROGRAMA PRINCIPAL - FUN  O QUE CALCULA O INDICE HASH DE UM NOME.
 *****************************************************************/

   FUNCTION CALCULEINDHASH (COMPRNOME   TCOMPRSIMB
                            NOME   TSIMBOLO)   TINDTABHASH

     VAR INDNOME   TCOMPRSIMB
         SOMACARAC   INTEGER

     BEGIN SOMACARAC  = 0
           FOR INDNOME  = 1 TO COMPRNOME
            DO SOMACARAC  = SOMACARAC + ORD (NOME(.INDNOME.))
           CALCULEINDHASH  = SOMACARAC MOD INDMAXTABHASH
     END


'*****************************************************************
 * PROGRAMA PRINCIPAL - PROCEDURE QUE, ANALIZANDO O ARQUIVO DE ENTRADA,
 *                      RECONHECE E FORMA AS ESTRUTURAS DE UMA CLAUSULA,
 *****************************************************************/

  PROCEDURE OBTENHACLAUS (VAR CLAUS  TAPCLAUS)

  LABEL 888  /* PONTO DE RECUPERA  O PARA ERROS SINTATICOS
             */


*****************************************************************
* OBTENHACLAUS - CONSTANTES                                      *
```

```
**********************************************************************/

 CONST MAXREPINT = 18
       RIERRO = 0
       RIFIMARQ = 1
       RIBRANCO = 2
       RICOMENTARIO = 3
       RIABREPAR = 4
       RIFECHAPAR = 5
       RIVIRGULA = 6
       RIFIMCLS = 7
       RINUMERO = 8
       RIDECREAL = 9
       RIVARANCNIMA = 10
       RIVARIAVEL = 11
       RIPREDFUNC = 12
       RIVAZIO = 13
       RISBESPECIAL = 14
       RICARACDESTRING = 15
       RISIMBSE = 16
       RIINTEIRO = 17
       RIREAL = 18


/**********************************************************************
 * CBTENHACLAUS - DEFINI  O DE TIPO                                   *
 **********************************************************************/

 TYPE REPRESINTERNA = 0..MAXREPINT


/**********************************************************************
 * CBTENHACLAUS - DECLARA  O DE VARIAVEIS
 **********************************************************************/

 VAR COMPRSIMB , CCMPRSBSALVO   TCOMPRSIMB
     SIMB, SIMBSALVO   TSIMBOLO
     REPINT, REPINTSALVA   REPRESINTERNA
     CARACT   CHAR
     TABCOMPRVAR   ARRAY (.1..NMAXVARSCLAUS.) OF TCOMPRSIMB
     TABVAR   ARRAY (.1..NMAXVARSCLAUS.) OF TSIMBOLO

/*****************************************************
********* PROCEDURE USADA NA DEPURA AO ****************/

 PROCEDURE DEPURA

        BEGIN CASE REPINT
              OF RIERRO  WRITELN ('DETECTADO ERRO')
                 RIFIMARQ  WRITELN ('ENCONTRADO O FINAL DO ARQUIVO')
                 RIBRANCO  WRITELN ('BRANCOS')
                 RICOMENTARIO  WRITELN ('COMENTARIO')
                 RIABREPAR  WRITELN ('ABRE PARENTESIS')
                 RIFECHAPAR  WRITELN ('FECHAPARENTESIS')
                 RIVIRGULA  WRITELN ('VIRGULA')
```

```
                    RIFIMCLS  WRITELN ('FIM DA CLAUSULA')
                    RINUMERO  WRITELN ('NUMERO')
                    RIDECREAL  WRITELN ('PARTE DECIMAL DE UM NUMERO REAL')
                    RIVARANONIMA  WRITELN ('VARIAVEL ANONIMA')
                    RIVARIAVEL  WRITELN ('VARIAVEL')
                    RIPREDFUNC  WRITELN ('NOME DE PREDICADO OU FUNCAO')
                    RIVAZIO  WRITELN ('FINAL DE STRING')
                    RISBESPECIAL  WRITELN ('SIMBOLO ESPECIAL')
                    RICARACDESTRING  WRITELN ('CARACTER DE STRING')
                    RISIMBSE   WRITELN ('SIMBOLO SE')
                   END
        END

/**************************************************************************
 * CBTENHACLAUS - PROCEDURE QUE IMPRIME O ULTIMO SIMBOLO RECONHECIDO
 **************************************************************************/

 PROCEDURE ESCREVASIMB

   VAR I   TCOMPRSIMB

   BEGIN FOR I  = 1 TO COMPRSIMB
          DO WRITE (SIMB(.I.))
         WRITELN
   END


/**************************************************************************
 * CBTENHACLAUS - PROCEDURE QUE OBTEM UM CARACTER DO ARQUIVO DE ENTRADA
 **************************************************************************/

 PROCEDURE GC ( VAR ARQUIVO  TEXT
                VAR CARACT   CHAR)

   BEGIN IF NOT EOF (ARQUIVO)
           THEN READ (ARQUIVO,CARACT)
           ELSE GOTO 888 /* DESVIO PARA O PONTO DE RECUPERA  O
                         * DE ERROS SINTATICOS.
                         */
   END


/**************************************************************************
 * CBTENHACLAUS - PROCEDURE QUE MANIPULA UM ERRO, OU SEJA, DESCARTA O
 *                RESTANTE DO STRING QUE FORMA UMA CLAUSULA COM ERRO.
 **************************************************************************/

 PROCEDURE ERRO ( VAR ARQUIVO  TEXT
                  VAR CARACT   CHAR)

   BEGIN WRITELN (
'PARA RECUPERAR DO ERRO FOI DESCARTADO O STRING '
                 )
         REPEAT WHILE CARACT )  '.'
                  DO BEGIN IF EOF (ARQUIVO)
```

```
                                THEN GOTO 888 /* DESVIO PARA O PONTO DE
                                              * RECUPERA  O DE ERROS
                                              * SINTATICOS
                                              */
                                WRITE ( CARACT)
                                 GC (ARQUIVO, CARACT)
                             END
                        GC (ARQUIVO, CARACT)
                UNTIL EOF (ARQUIVO) OR  EOLN (ARQUIVO) OR (CARACT )  ' ')
                WRITELN
                WRITELN (
'FINAL DA RECUPERA$'O DE ERRO'
                        )
    END


/**********************************************************************
 * OBTENHACLAUS - PROCEDURE QUE OBTEM SIMBOLOS LEXICAMENTE VALIDOS DA
 *                LINGUAGEM PROLOG
 **********************************************************************/

 PROCEDURE OBTENHASIMBVALIDO

    VAR I   TCOMPRSIMB


/**********************************************************************
 * OBTENHACLAUS/OBTENHASIMBVALIDO - PROCEDURE QUE ACRESCENTA O ULTIMO
 *      CARACTER LIDO AO SIMBOLO EM RECONHECIMENTO.
 **********************************************************************/

 PROCEDURE ADD ( CARACT   CHAR
                 VAR SIMB   TSIMBOLO
                 VAR COMPRSIMB   TCOMPRSIMB)

    BEGIN COMPRSIMB  = COMPRSIMB + 1
          SIMB (.COMPRSIMB.)  = CARACT
    END


/**********************************************************************
 * OBTENHACLAUS/OBTENHASIMBVALIDO - PROCEDURE QUE OBTEM OS SIMBOLOS QUE
 *                                  MANIPULADOS POR OBTENHASIMBVALIDO
 **********************************************************************/

 PROCEDURE OBTENHASIMBOLO (VAR ARQUIVO  TEXT
                           VAR REPINT  REPRESINTERNA
                           VAR COMPRSIMB   TCOMPRSIMB
                           VAR SIMB   TSIMBOLO
                           VAR CARACT   CHAR )


/**********************************************************************
 * OBTENHACLAUS/OBTENHASIMBVALIDO/OBTENHASIMBOLO - CONSTANTES
 **********************************************************************/
```

UNIVERSIDADE FEDERAL DA PARAÍBA
Pró-Reitoria Para Assuntos do Interior
Coordenação Setorial de Pós-Graduação
Rua Aprigio Veloso, 882 - Tel. (083) 321-7222-R 355
58.100 - Campina Grande - Paraíba

```
CONST DIGITO = (.'0'..'9'.)
      MAIUSCULA  = (.'A'..'Z','$','=',' ',''.)
      MINUSCULA = (.'A'..'Z',' ',' ',' ',' '.)
      ALFANUM = (.'A'..'Z', 'A'..'Z', '0'..'9', ' ',' ',' ',' ',' ',
                 ' ', '=', ' ', ''.)
      DELIMITADOR = (.'(', ')', ',', ' '.)
      ESPECIAL = (.' ', ' ', ' ', '(', '&', '/', '=', ' ', ' ', ' ',
                  '*', '+', '-', ' ', ' ', ')', ' '.)


/*****************************************************************
 * OBTENHACLAUS/OBTENHASIMBVALIDO/OBTENHASIMBOLO - DECLARA  O DE
 *                                                 VARIAVEL
 *****************************************************************/

VAR I   INTEGER


/*****************************************************************
 * OBTENHACLAUS/OBTENHASIMBVALIDO/OBTENHASIMBOLO - PROCEDURE QUE RECO-
 *      NHECE UM COMENTARIO
 *****************************************************************/

PROCEDURE COMENTARIO ( VAR ARQUIVO   TEXT
                       VAR REPINT   REPRESINTERNA
                       VAR CARACT   CHAR)

  BEGIN GC (ARQUIVO, CARACT)
        REPEAT WHILE CARACT )  '*'
                 DO GC (ARQUIVO, CARACT)
               GC (ARQUIVO,CARACT)
        UNTIL CARACT = '/'
        GC (ARQUIVO, CARACT)
        REPINT  = RICOMENTARIO
  END


/*****************************************************************
 * OBTENHACLAUS/OBTENHASIMBVALIDO/OBTENHASIMBOLO - PROCEDURE QUE RECO-
 *      NHECE UM STRING ENTRE APOSTROFOS COMO PREDICADO OU FUN  O
 *****************************************************************/

PROCEDURE RECENTREAPOSTROFOS ( VAR ARQUIVO   TEXT
                               VAR CARACT   CHAR
                               VAR COMPRSIMB   TCOMPRSIMB
                               VAR SIMB   TSIMBOLO)

  VAR FECHAAPOSTROFO   BOOLEAN

  BEGIN FECHAAPOSTROFO  = FALSE
        GC (ARQUIVO, CARACT)
        REPEAT WHILE CARACT )  ''''
                 DO BEGIN ADD (CARACT, SIMB, COMPRSIMB)
                          GC (ARQUIVO, CARACT)
```

```
                    END
                GC (ARQUIVO, CARACT)
                IF CARACT = '''
                  THEN BEGIN ADD (CARACT, SIMB, COMPRSIMB)
                             GC (ARQUIVO, CARACT)
                       END
                  ELSE FECHAAPOSTROFO  = TRUE
        UNTIL FECHAAPOSTROFO
  END


{*****************************************************************
 * OBTENHACLAUS/OBTENHASIMBVALIDO/OBTENHASIMBOLO - PROCEDURE QUE RECO-
 *      NHECE A PARTE DECIMAL DE UM NUMERO REAL
 *****************************************************************}

PROCEDURE RECDECREAL ( VAR ARQUIVO   TEXT
                       VAR CARACT   CHAR
                       VAR SIMB   TSIMBOLO
                       VAR COMPRSIMB   TCOMPRSIMB
                       VAR REPINT   REPRESINTERNA )

  BEGIN WHILE (CARACT IN DIGITO)
          DO BEGIN ADD (CARACT, SIMB, COMPRSIMB)
                   GC (ARQUIVO, CARACT)
             END
        IF (CARACT = 'E') OR (CARACT = 'E')
          THEN BEGIN ADD (CARACT, SIMB, COMPRSIMB)
                     GC (ARQUIVO, CARACT)
                     IF  CARACT = '-'
                       THEN BEGIN ADD (CARACT, SIMB, COMPRSIMB)
                                  GC (ARQUIVO, CARACT)
                            END
                     IF CARACT IN NOT DIGITO
                       THEN BEGIN WRITELN
                                  WRITELN (
ERRO LEXICO  ESPOENTE N C NUMERICO'
                                           )
                                  ERRO (ARQUIVO, CARACT)
                                  REPINT  = RIERRO
                            END
                       ELSE BEGIN WHILE CARACT IN DIGITO
                                    DO BEGIN ADD (CARACT, SIMB
                                                  , COMPRSIMB )
                                             GC (ARQUIVO, CARACT)
                                       END
                                  REPINT  = RIDECREAL
                            END
             END
          ELSE REPINT  = RIDECREAL
  END


{*****************************************************************
 * OBTENHACLAUS/OBTENHASIMBVALIDO/OBTENHASIMBOLO - PROCEDURE QUE RE-
```

```
*       CONHECE UMA VARIAVEL
**********************************************************************/

PROCEDURE RECVARIAVEL ( VAR ARQUIVO   TEXT
                        VAR CARACT   CHAR
                        VAR COMPRSIMB   TCOMPRSIMB
                        VAR SIMB   TSIMBOLO
                        VAR REPINT   REPRESINTERNA )

  BEGIN WHILE CARACT IN ALFANUM
        DO BEGIN ADD (CARACT, SIMB, COMPRSIMB)
                 GC (ARQUIVO, CARACT)
           END
      REPINT  = RIVARIAVEL
  END


/**********************************************************************
 * OBTENHACLAUS/OBTENHASIMBVALIDO/OBTENHASIMBOLO - PROCEDURE QUE RECO-
 *       NHECE UM STRING DE CARACTERES, ENTRE ASPAS, TOMANDO UM CARACTER
 *       DE CADA VEZ.
 **********************************************************************/

PROCEDURE RECCARACDESTRING ( VAR ARQUIVO   TEXT
                             VAR CARACT   CHAR
                             VAR COMPRSIMB   TCOMPRSIMB
                             VAR SIMB   TSIMBOLO
                             VAR REPINT   REPRESINTERNA)

  VAR I   INTEGER

  BEGIN GC (ARQUIVO, CARACT)
        IF CARACT = ' '
          THEN BEGIN GC (ARQUIVO, CARACT)
                     IF CARACT = ' '
                       THEN BEGIN ADD (CARACT, SIMB, COMPRSIMB)
                                  GC (ARQUIVO, CARACT)
                                  REPINT  = RICARACDESTRING
                            END
                       ELSE REPINT  = RIVAZIO
               END
          ELSE BEGIN ADD (CARACT, SIMB, COMPRSIMB)
                     CARACT  = ' '
                     REPINT  = RICARACDESTRING
               END
  END


/**********************************************************************
 * OBTENHACLAUSULA/OBTENHASIMBVALIDO/OBTENHASIMBOLO - CORPO DA PROCEDURE
 **********************************************************************/

  BEGIN
  IF REPINTSALVA )  RIBRANCO
    THEN BEGIN REPINT  = REPINTSALVA
```

```
            COMPRSIMB  = COMPRSBSALVO
            FOR I  = 1 TO COMPRSIMB
               DO SIMB(.I.)  = SIMBSALVO(.I.)
            REPINTSALVA  = RIBRANCO
      END
ELSE BEGIN COMPRSIMB  = 0
           IF EOF (ARQUIVO)
             THEN  REPINT  = RIFIMARQ
             ELSE CASE CARACT
             OF ' '  BEGIN ADD (CARACT, SIMB, COMPRSIMB)
                           WHILE CARACT = ' '
                             DO GC (ARQUIVO, CARACT)
                           REPINT  = RIBRANCO
                      END
                '('  BEGIN ADD (CARACT, SIMB, REPINT)
                           GC (ARQUIVO, CARACT)
                           REPINT  = RIABREPAR
                      END
                ')'  BEGIN ADD (CARACT, SIMB, COMPRSIMB)
                           GC (ARQUIVO, CARACT)
                           REPINT  = RIFECHAPAR
                      END
                ','  BEGIN ADD (CARACT, SIMB, COMPRSIMB)
                           GC (ARQUIVO, CARACT)
                           REPINT  = RIVIRGULA
                      END
                '0'..'9'  BEGIN WHILE CARACT IN DIGITO
                                DO BEGIN ADD (CARACT,SIMB,COMPRSIMB)
                                         GC (ARQUIVO, CARACT)
                                   END
                             REPINT  = RINUMERO
                          END
                '.'  BEGIN ADD (CARACT, SIMB, COMPRSIMB)
                           GC (ARQUIVO, CARACT)
                           IF CARACT = ' '
                             THEN REPINT  = RIFIMCLS
                           ELSE IF CARACT IN DIGITO
                                  THEN RECDECREAL (ARQUIVO,CARACT
                                                  ,SIMB,COMPRSIMB,REPINT)
                                  ELSE REPINT  = RISBESPECIAL
                      END
              ' '  BEGIN ADD (CARACT, SIMB, COMPRSIMB)
                         GC (ARQUIVO, CARACT)
                         IF CARACT IN NOT ALFANUM
                           THEN REPINT  = RIVARANCNIMA
                           ELSE RECVARIAVEL (ARQUIVO, CARACT
                                            ,COMPRSIMB,SIMB,REPINT)
                      END
              'A'..'Z'
                        BEGIN WHILE CARACT IN ALFANUM
                                DO BEGIN ADD (CARACT, SIMB
                                             , COMPRSIMB )
                                         GC (ARQUIVO, CARACT)
                                   END
                              REPINT  = RIPREDFUNC
```

```
                                END
                    'A'..'Z'
                                BEGIN ADD (CARACT, SIMB, COMPRSIMB)
                                      GC (ARQUIVO, CARACT)
                                      RECVARIAVEL (ARQUIVO, CARACT
                                                  ,COMPRSIMB,SIMB,REPINT)
                                END
                    ''''  BEGIN RECENTREAPOSTROFOS (ARQUIVO, CARACT
                                                  , COMPRSIMB,  SIMB)
                                REPINT  = RIPREDFUNC
                          END
                    ' '   RECCARACDESTRING (ARQUIVO, CARACT, COMPRSIMB
                                           , SIMB, REPINT)
                    '/'   BEGIN ADD (CARACT, SIMB, COMPRSIMB)
                                GC (ARQUIVO, CARACT)
                                IF CARACT = '*'
                                  THEN COMENTARIO (ARQUIVO, REPINT, CARACT)
                                  ELSE REPINT  = RISBESPECIAL
                          END
                    ' ', '(', '&', '=', ' ', ' ', ' ', '*', '+', '-', ' ',
                    ' ', ' ', ')'  BEGIN ADD (CARACT, SIMB, COMPRSIMB)
                                         GC (ARQUIVO, CARACT)
                                         REPINT  = RISBESPECIAL
                                   END
               OTHERWISE BEGIN WRITELN
                               WRITELN (
'ERRO LEXICO  SIMBOLO INVALIDO'
                                       )
                               ERRO (ARQUIVO, CARACT)
                         END
             END
          END
    END


/**************************************************************************
 * OBTENHACLAUS/OBTENHASIMBVALIDO - CORPO DA PROCEDURE
 **************************************************************************/

    BEGIN OBTENHASIMBOLO (INPUT,REPINT, COMPRSIMB, SIMB, CARACT)
          WHILE (REPINT = RIBRANCO) OR (REPINT = RICOMENTARIO)
            DO OBTENHASIMBOLO (INPUT,REPINT,COMPRSIMB,SIMB,CARACT)
          IF (REPINT = RISBESPECIAL) AND (SIMB(.1.) = ')')
            THEN BEGIN OBTENHASIMBOLO (INPUT,REPINTSALVA,COMPRSBSALVO
                                      ,SIMBSALVO, CARACT)
                       IF (REPINT = RISBESPECIAL) AND
                          (SIMBSALVO (.1.) = '-')
                         THEN BEGIN REPINT  = RISIMBSE
                                    COMPRSIMB  = 2
                                    SIMB(.2.)  = SIMBSALVO(.1.)
                                    REPINTSALVA  = RIBRANCO
                                END
                 END
            ELSE IF (REPINT = RISBESPECIAL) AND (SIMB(.1.) = '-')
                   THEN BEGIN OBTENHASIMBOLO(INPUT, REPINTSALVA
```

UNIVERSIDADE FEDERAL DA PARAÍBA
Pró-Reitoria Para Assuntos do Interior
Coordenação Setorial de Pós-Graduação
Rua Aprigio Veloso, 882 - Tel. (083) 321-7222-R 355
58.100 - Campina Grande - Paraíba

```
                                        ,COMPRSBSALVO,SIMBSALVO
                                        ,CARACT)
                                IF REPINTSALVA = RINUMERO
                                  THEN BEGIN COMPRSIMB  = COMPRSBSALVO + 1

                                             FOR I  = 1 TO COMPRSBSALVO
                                              DO SIMB(.I+1.)  = SIMBSALVO
                                                                (.I.)
                                             REPINT  = RINUMERO
                                             REPINTSALVA  = RIBRANCO
                                        END
                            END
                    IF REPINT = RINUMERO
                      THEN BEGIN OBTENHASIMBOLO (INPUT, REPINTSALVA
                                                ,COMPRSBSALVO, SIMBSALVO
                                                ,CARACT )
                                 IF REPINTSALVA = RIDECREAL
                                   THEN BEGIN REPINT  = RIREAL
                                              FOR I  = 1 TO COMPRSBSALVO
                                               DO ADD (SIMBSALVO(.I.),SIMB
                                                      , COMPRSIMB )
                                              REPINTSALVA  = RIBRANCO
                                         END
                                   ELSE REPINT  = RIINTEIRO
                           END
    END

/**************************************************************************
 * OBTENHACLAUS - PROCEDURE QUE VERIFICA SE O SIMBOLO RECONHECIDO
 *      DIFERENTE DO NOME INTRODUZIDO NOS PARAMETROS.
 **************************************************************************/

  FUNCTION NOMESDIFERENTES (COMPRNOME   TCOMPRSIMB
                            NOME   TSIMBOLO        )   BOOLEAN

      VAR I   TCOMPRSIMB
          MESMONOME   BOOLEAN

      BEGIN I  = 1
            MESMONOME  = TRUE
            WHILE (I )= COMPRSIMB) AND MESMONOME
                  DO BEGIN IF NOME(.I.) )  SIMB(.I.)
                             THEN MESMONOME  = FALSE
                           I  = I + 1
                     END
            NOMESDIFERENTES  = NOT MESMONOME
      END


/**************************************************************************
 * OBTENHACLAUS - FUN  O QUE BUSCA, NA TABELA DE VARIAVEIS, A VARIAVEL
 *                REPRESENTADA PELO ULTIMO SIMBOLO RECONHECIDO. SE N O
 *                ENCONTRAR, RETORNA O INDICE DO PROXIMO ELEMENTO A AR-
 *                MAZENAR NA TABELA.
 **************************************************************************/
```

```
    FUNCTION BUSQVARNATABELA (NVARS   TINDVAR)  TINDVAR

      VAR IND   TINDVAR

      BEGIN IND  = 1
            WHILE (IND )= NVARS) AND ((CCMPRSIMB )  TABCCMPRVAR (.IND.))
                                  OR NOMESDIFERENTES (TABCOMPRVAR (.IND.)
                                                     ,TABVAR (.IND.)))
              DO IND  = IND + 1
            BUSQVARNATABELA  = IND
      END


/*****************************************************************
 * OBTENHACLAUS - PROCEDURE QUE OBTEM E ARMAZENA NA MEMORIA A ESTRUTURA
 *                DE UM PREDICADO OU FUN  O.
 *****************************************************************/

    PROCEDURE OBTENHAPREDOUFUNC (VAR PREDOUFUNC   TAPPREDOUFUNC
                                 VAR NVARS   TINDVAR)

      FORWARD


/*****************************************************************
 * OBTENHACLAUS - PROCEDURE QUE OBTEM E ARMAZENA NA MEMORIA A ESTRUTURA
 *                DE UMA LISTA DE ARGUMENTOS.
 *****************************************************************/

    PROCEDURE OBTENHAARGUMENTOS (VAR ARGUMENTO   TAPSMCUARG
                                 VAR NVARS   TINDVAR)


/*****************************************************************
 * OBTENHACLAUS/OBTENHAARGUMENTOS - PROCEDURE QUE OBTEM E ARMAZENA NA
 *      MEMORIA UM ARGUMENTO INTEIRO.
 *****************************************************************/

    PROCEDURE OBTENHAINTEIRO (VAR ARGINT   TAPSMOUARG)


/*****************************************************************
 * OBTENHACLAUS/OBTENHAARGUMENTOS/OBTENHAINTEIRO - FUN  O  QUE  CALCULA
 *      O VALOR INTEIRO CORRESPONDENTE AOS CARACTERES NUMERICOS CONTIDOS
 *      EM SIMB.
 *****************************************************************/

    FUNCTION CALCVALINTEIRO (CCMPRSIMB   TCOMPRSIMB
                             SIMB   TSIMBOLO )   INTEGER

      VAR I   TCOMPRSIMB
          NUMERO   INTEGER
          SINAL    -1..1
```

```
      BEGIN IF COMPRSIMB )= NMAXDIGITOS
              THEN BEGIN NUMERO  = 0
                         IF SIMB(.1.) = '-'
                           THEN BEGIN SINAL  = -1
                                      I  = 2
                                END
                           ELSE BEGIN SINAL  = 1
                                      I  = 1
                                END
                         WHILE I )= COMPRSIMB
                          DO BEGIN NUMERO  = NUMERO * 10 + (ORD(SIMB(.I.)
                                                            ) -ORD('0'))
                                   I  = I + 1
                             END
                         CALCVALINTEIRO  = SINAL * NUMERO
                   END
              ELSE BEGIN WRITELN
                         WRITELN (
  'ERRO SINTATICO  NUMERO INTEIRO COM MAIS QUE ',NMAXDIGITOS,' DIGITOS'
                                  )
                         ERRO (ARQUIVO,CARACT)
                         ELIMINECLAUS (CLAUS)
                         GOTO 888 /* DESVIO PARA O PONTO DE RECUPERA  O
                                   * DE ERROS SINTATICOS
                                   */
                   END
      END


/**********************************************************************
 * OBTENHACLAUS/OBTENHAARGUMENTOS/OBTENHAINTEIRO - CORPO DA PROCEDURE
 **********************************************************************/

        BEGIN NEW (ARGINT)
              ARGINT - . TIPO  = NINT
              ARGINT - . VALORINTEIRO  = CALCVALINTEIRO (COMPRSIMB,SIMB)
              OBTENHASIMBVALIDO
        END


/**********************************************************************
 * OBTENHACLAUS/OBTENHAARGUMENTOS - PROCEDURE QUE OBTEM E ARMAZENA NA
 *       MEMORIA UM ARGUMENTO REAL
 **********************************************************************/

    PROCEDURE OBTENHAREAL (VAR ARGREAL  TAPSMOUARG)


/**********************************************************************
 * OBTENHACLAUS/OBTENHAARGUMENTOS/OBTENHAREAL - FUN  O QUE CALCULA O
 *       VALOR REAL CORRESPONDENTE AOS CARACTERES NUMERICOS CONTIDOS EM
 *       SIMB.
 **********************************************************************/

    FUNCTION CALCVALREAL (COMPRSIMB   TCOMPRSIMB
```

```
                        SIMB   TSIMBOLO)  REAL

    VAR I   TCCMPRSIMB
        PREAL, PDEC   REAL
        SINAL, SINALESP   -1..1
        PESP   -99..99

    BEGIN PESP  = 0
          PREAL  = 0.0
          PDEC  = 0.1
          IF SIMB(.1.) = '-'
            THEN BEGIN SINAL   = -1
                       I  = 2
                 END
            ELSE BEGIN SINAL   = 1
                       I  = 1
                 END
         WHILE SIMB(.I.) )  '.'
           DO BEGIN PREAL  = PREAL * 10 + (ORD(SIMB(.I.)) - ORD('0'))
                    I  = I + 1
              END
          I  = I + 1
          WHILE (I )= COMPRSIMB) AND (SIMB(.I.) )  'E') AND (SIMB(.I.)
                                                          )  'E')
           DO BEGIN PREAL  = PREAL + PDEC * (ORD(SIMB(.I.)) - ORD('0'))
                    PDEC  = PDEC * 0.1
                    I  = I + 1
              END
          IF I ) COMPRSIMB
            THEN BEGIN I  = I + 1
                       IF SIMB(.I.) = '-'
                         THEN BEGIN SINALESP  = -1
                                    I  = I + 1
                              END
                         ELSE SINALESP  = 1
                       PESP  = ORD(SIMB(.I.)) - ORD('0')
                       IF I ) COMPRSIMB
                         THEN PESP  = SINALESP * (10 * PESP + ORD(SIMB
                                                   (.I+1.)) - ORD('0'))
                         ELSE PESP  = SINALESP * PESP
                       IF I + 2 )= COMPRSIMB
                         THEN BEGIN WRITELN
                                    WRITELN (
'ERRO SINTATICO  ESPOENTE COM MAIS QUE DOIS DIGITOS'
                                            )
                                    ERRO (ARQUIVO,CARACT)
                                    ELIMINECLAUS(CLAUS)
                                    GOTO 888 /* DESVIO PARA O PONTO DE
                                             * RECUPERA  O DE ERROS
                                             * SINTATICOS.
                                             */
                              END
                 END
          IF PESP = 0
            THEN CALCVALREAL  = PREAL
```

UNIVERSIDADE FEDERAL DA PARAÍBA
Pró-Reitoria Para Assuntos do Interior
Coordenação Setorial de Pós-Graduação
Rua Aprigio Veloso, 882 - Tel (083) 321-7222-R 355
58.100 - Campina Grande - Paraíba

```
            ELSE CALCVALREAL  = PREAL * EXP (PESP*LN(10.0))
    END


/*****************************************************************
 * OBTENHACLAUS/OBTENHAARGUMENTOS/OBTENHAREAL - CORPO DA PROCEDURE
 *****************************************************************/

    BEGIN NEW (ARGREAL)
          ARGREAL - . TIPO   = NREAL
          ARGREAL - . VALORREAL   = CALCVALREAL (COMPRSIMB,SIMB)
          OBTENHASIMBVALIDO
    END


/*****************************************************************
 * OBTENHACLAUS/OBTENHAARGUMENTO - PROCEDURE QUE OBTEM E ARMAZENA NA
 *      MEMORIA UM ARGUMENTO VARIAVEL
 *****************************************************************/

   PROCEDURE OBTENHAVARIAVEL (VAR ARGVAR    TAPSMOUARG
                              VAR NVARS   TINDVAR)

    VAR I    TCOMPRSIMB

    BEGIN NEW (ARGVAR)
           WITH ARGVAR -
             DO BEGIN TIPO   = VARIAVEL
                      INDDAVARIAVEL   = BUSQVARNATABELA (NVARS)
                      IF INDDAVARIAVEL    NVARS
                        THEN BEGIN NVARS   = INDDAVARIAVEL
                                   TABCOMPRVAR (.INDDAVARIAVEL.)   =
                                                          COMPRSIMB
                                   FOR I   = 1 TO COMPRSIMB
                                    DO TABVAR (.INDDAVARIAVEL.)(.I.)   =
                                                        SIMB (.I.)
                                END
                END
         OBTENHASIMBVALIDO
    END


/*****************************************************************
 * OBTENHACLAUS/OBTENHAARGUMENTOS - PROCEDURE QUE OBTEM E ARMAZENA NA
 *      MEMORIA A ESTRUTURA DE UM ARGUMENTO TIPO CARACTER DE STRING
 *****************************************************************/

   PROCEDURE OBTENHACARACSTR (VAR ARGCH    TAPSMOUARG)

    BEGIN NEW (ARGCH)
          ARGCH - . TIPO   = CARACSTR
          ARGCH - . VALORCARAC   = SIMB(.1.)
          OBTENHASIMBVALIDO
    END
```

```
/**********************************************************************
 * OBTENHACLAUS/OBTENHAARGUMENTOS - CORPO DA PROCEDURE
 **********************************************************************/
    BEGIN IF (REPINT )  RICARACDESTRING) AND (REPINT )  RIVAZIO)
            THEN OBTENHASIMBVALIDO
          IF (REPINT = RIPREDFUNC) OR (REPINT = RISBESPECIAL)
            THEN BEGIN NEW (ARGUMENTO)
                       ARGUMENTO - . TIPO  = PREDOUFUNC
                       OBTENHAPREDOUFUNC (ARGUMENTO - . APPREDOUFUNC
                                          , NVARS )
                 END
            ELSE IF REPINT = RIVARANONIMA
                   THEN BEGIN NEW (ARGUMENTO)
                              ARGUMENTO - . TIPO  = VARIAVEL
                              ARGUMENTO - . INDDAVARIAVEL  = 0
                              OBTENHASIMBVALIDO
                        END
                   ELSE IF REPINT = RIVARIAVEL
                          THEN OBTENHAVARIAVEL (ARGUMENTO,NVARS)
                          ELSE IF REPINT = RIINTEIRO
                                 THEN OBTENHAINTEIRO (ARGUMENTO)
                                 ELSE IF REPINT = RIREAL
                                        THEN OBTENHAREAL (ARGUMENTO)
                                        ELSE IF (REPINT =
                                                   RICARACDESTRING)
                                               OR (REPINT = RIVAZIO)
                                             THEN OBTENHACARACSTR (
                                                    ARGUMENTO)
                                             ELSE BEGIN WRITELN
                                                        WRITELN (
'ERRO SINTATICO  ESPERAVA-SE UM ARGUMENTO NO ENTANTO FOI OBTIDO O'
                                                                )
                                                        WRITE (
'                SIMBOLO  '
                                                              )
                                                        ESCREVASIMB
                                                  ERRO (INPUT,CARACT)
                                                  ELIMINECLAUS(CLAUS)
                                                  GOTO 888
                                                  /* DESVIO P/PONTO DE
                                                   * RECUPERA AO ERROS
                                                   */
                                                  END
          IF REPINT = RIFECHAPAR
            THEN BEGIN ARGUMENTO - . PROXSMOUARG  = NIL
                       OBTENHASIMBVALIDO
                 END
            ELSE IF (REPINT = RIVIRGULA) OR (REPINT = RICARACDESTRING)
                    OR (REPINT = RIVAZIO)
                   THEN OBTENHAARGUMENTOS (ARGUMENTO - . PROXSMOUARG
                                           , NVARS )
                   ELSE BEGIN WRITELN
                              WRITELN (
'ERRO SINTATICO  ESPERAVA-SE UMA VIRGULA OU UM FECHA-PARENTESIS, NO'
```

```
                                    )
                                WRITE (
'                   ENTATO OBTEVE-SE O SIMBOLO  '
                                    )
                                ESCREVASIMB
                                ERRO (INPUT, CARACT)
                                ELIMINECLAUS (CLAUS)
                                GOTO 888 /* DESVIO P/PONTO DE RECUPERA  O
                                         * DE ERROS SINTATICOS
                                         */
                            END
    END

/**********************************************************************
 * OBTENHACLAUS - PROCEDURE QUE BUSCA UM ELEMENTO DA TABELA DE SIMBOLOS,
 *                CRIANDO-O CASO AINDA N O EXISTA.
 **********************************************************************/

  PROCEDURE BUSQELEMTABSIMB (VAR ELEM   TAPELEMTABSIMB)

    VAR INDHASH   TINDTABHASH

/**********************************************************************
 * OBTENHACLAUS/BUSQELEMTABSIMB - PROCEDURE QUE CRIA E ARMAZENA NA MEMO-
 *                                RIA A ESTRUTURA DE UM NOVO ELEMENTO DA
 *                                TABELA DE SIMBOLOS.
 **********************************************************************/

  PROCEDURE CRIEELEMTABSIMB (VAR ELEM   TAPELEMTABSIMB)

    VAR I   TCOMPRSIMB

    BEGIN NEW (ELEM)
          ELEM - . COMPRNOME  = COMPRSIMB
          FOR I  = 1 TO COMPRSIMB
           DO ELEM - . NOME(.I.)  = SIMB(.I.)
          ELEM - . GPOCLS  = NIL
          ELEM - . APNOMECOLISOR  = NIL
    END


/**********************************************************************
 * OBTENHACLAUS/BUSQELEMTABSIMB - PROCEDURE QUE BUSCA UM NOME NA LISTA
 *                                DE ELEMENTOS DA TABELA QUE TEM O MES-
 *                                MO INDICE HASH
 **********************************************************************/

  PROCEDURE BUSQELEMLISTATAB (VAR ELEM   TAPELEMTABSIMB)



/**********************************************************************
 * OBTENHACLAUS/BUSQELEMTABSIMB/BUSQELEMLISTATAB - CORPO DA PROCEDURE
 **********************************************************************/
```

```
    BEGIN IF (ELEM - . COMPRNOME )  COMPRSIMB) OR NOMESDIFERENTES (
                                                    ELEM - . COMPRNOME
                                                   ,ELEM - . NOME   )
            THEN IF ELEM - . APNOMECOLISOR = NIL
                   THEN BEGIN CRIEELEMTABSIMB (ELEM - . APNOMECOLISOR)
                              ELEM  = ELEM - . APNOMECOLISOR
                        END
                   ELSE BEGIN ELEM  = ELEM - . APNOMECOLISOR
                              BUSQELEMLISTATAB (ELEM)
                        END
    END


/*****************************************************************
 * OBTENHACLAUS/BUSQELEMTABSIMB - CORPO DA PROCEDURE
 *****************************************************************/

    BEGIN INDHASH  = CALCULEINDHASH (COMPRSIMB,SIMB)
          IF TABHASH (.INDHASH.) = NIL
            THEN BEGIN CRIEELEMTABSIMB (ELEM)
                       TABHASH (.INDHASH.)  = ELEM
                 END
            ELSE BEGIN ELEM  = TABHASH(.INDHASH.)
                       BUSQELEMLISTATAB (ELEM)
                 END
    END


/*****************************************************************
 * OBTENHACLAUS/OBTENHAPREDOUFUNC - DEFINI  O DA PROCEDURE (DEFINIDA
 *                                  ANTERIORMENTE COM  FORWARD ).
 *****************************************************************/

  PROCEDURE OBTENHAPREDOUFUNC

    BEGIN NEW (PREDOUFUNC)
          BUSQELEMTABSIMB (PREDOUFUNC - . NOME)
          OBTENHASIMBVALIDO
          IF REPINT )  RIABREPAR
            THEN PREDOUFUNC - . PRIMARG  = NIL
            ELSE OBTENHAARGUMENTOS (PREDOUFUNC - . PRIMARG, NVARS)
    END


/*****************************************************************
 * OBTENHACLAUS - PROCEDURE QUE OBTEM E ARMAZENA NA MEMORIA A ESTRUTURA
 *                DA CABECA DE UMA CLAUSULA.
 *****************************************************************/

  PROCEDURE OBTENHACABECA (VAR CABECA   TAPPREDOUFUNC
                           VAR NVARS   TINDVAR)

    BEGIN IF SIMB (.1.) = ' '
            THEN CABECA  = NIL
```

UNIVERSIDADE FEDERAL DA PARAÍBA
Pró-Reitoria Para Assuntos do Interior
Coordenação Setorial de Pós-Graduação
Rua Aprigio Veloso, 882 - Tel (083) 321-7222-R 355
58.100 - Campina Grande - Paraíba

```
            ELSE IF (REPINT )  RIPREDFUNC) AND (REPINT )  RISBESPECIAL)
                   THEN BEGIN WRITELN
                              WRITELN (
'ERRO SINTATICO  ESPERAVA-SE O NOME DO PREDICADO DA CABE A, POREM FOI'
                                       )
                              WRITE (
'                OBTIDO O SIMBOLO  '
                                    )
                              ESCREVASIMB
                              ERRO (INPUT, CARACT)
                              ELIMINECLAUS (CLAUS)
                              GOTO 888 /* DESVIO PARA O PONTO DE RECU-
                                       * PERACAO DE ERROS SINTATICOS
                                       */
                         END
                   ELSE OBTENHAPREDOUFUNC (CABECA,NVARS)
     END


/**********************************************************************
 * OBTENHACLAUS - PROCEDURE QUE OBTEM E ARMAZENA NA MEMORIA A ESTRUTURA
 *                DE UMA META-VARIAVEL.
 **********************************************************************/

   PROCEDURE OBTENHAMETAVAR (VAR METAVAR   TAPSMOUARG
                             VAR NVARS   TINDVAR)

     VAR INDICE   TINDVAR

     BEGIN INDICE  = BUSQVARNATABELA (NVARS)
           IF INDICE )= NVARS
             THEN BEGIN NEW (METAVAR)
                        METAVAR - . TIPO  = VARIAVEL
                        METAVAR - . INDDAVARIAVEL  = INDICE
                  END
             ELSE BEGIN WRITELN
                        WRITELN (
'ERRO SINTATICO  PARA QUE UM NOME POSSA APARECER COMO META-VARIAVEL,  '
                                 )
                        WRITELN (
'                NECESSARIO QUE TENHA APARECIDO ANTERIORMENTE, NA CLAU-'
                                 )
                        WRITELN (
'                SULA, COMO VARIAVEL'
                                 )
                        ERRO (INPUT, CARACT)
                        ELIMINECLAUS (CLAUS)
                        GOTO 888 /* DESVIO PARA O PONTO DE RECUPERA  O
                                 * DE ERROS SINTATICOS
                                 */
                  END
          OBTENHASIMBVALIDO
     END
```

```
/**********************************************************************
 * OBTENHACLAUS - PROCEDURE QUE OBTEM E ARMAZENA NA MEMORIA UMA LISTA
 *                DE SUBMETAS DE UMA CLAUSULA.
 **********************************************************************/

   PROCEDURE OBTENHASUBMETAS (VAR SMETA    TAPSMOUARG
                              VAR NVARS    TINDVAR)

     BEGIN OBTENHASIMBVALIDO
           IF REPINT = RIVARIAVEL
             THEN OBTENHAMETAVAR (SMETA, NVARS)
             ELSE IF (REPINT = RIPREDFUNC) OR (REPINT = RISBESPECIAL)
                    THEN BEGIN NEW (SMETA)
                               SMETA - . TIPO  = PREDOUFUNC
                               OBTENHAPREDOUFUNC (SMETA - . APPREDOUFUNC
                                                  , NVARS)
                         END
                    ELSE BEGIN WRITELN
                               WRITELN (
'ERRO SINTATICO  ESPERAVA-SE UMA SUBMETA E NO ENTANTO FOI OBTIDO O'
                                        )
                               WRITE (
'                SIMBOLO  '
                                      )
                               ESCREVASIMB
                               ERRO (INPUT, CARACT)
                               ELIMINECLAUS (CLAUS)
                               GOTO 888 /* DESVIO PARA O PONTO DE RECU-
                                        * PERA  O DE ERROS SINTATICOS
                                        */
                         END
          IF REPINT = RIFIMCLS
            THEN SMETA - . PROXSMOUARG  = NIL
            ELSE IF SIMB(.1.) = '&'
                   THEN OBTENHASUBMETAS (SMETA - . PROXSMOUARG
                                                        ,NVARS)
                   ELSE BEGIN WRITELN
                              WRITELN (
'ERRO SINTATICO  ESPERAVA-SE O SIMBOLO & OU . (FINAL DE CLAUSULA), NO'
                                       )
                              WRITE (
'                ENTANTO OBTEVE-SE O SIMBOLO  '
                                      )
                              ESCREVASIMB
                              ERRO (INPUT, CARACT)
                              ELIMINECLAUS (CLAUS)
                              GOTO 888 /* DESVIO PARA O PONTO DE RECU-
                                       * PERA  O DE ERROS SINTATICOS
                                       */
                        END
     END

/**********************************************************************
 * OBTENHACLAUS - PROCEDURE QUE OBTEM E ARMAZENA NA MEMORIA A ESTRUTURA
```

```
*                 DO CORPO DA CLAUSULAS (SUBMETAS).
*****************************************************************/

   PROCEDURE OBTENHACORPO (VAR PRIMSMETA   TAPSMOUARG
                           VAR NVARS   TINDVAR)

     BEGIN IF REPINT = RIFIMCLS
             THEN PRIMSMETA  = NIL
             ELSE BEGIN IF (SIMB(.1.) = ' ') OR (REPINT = RISIMBSE)
                        THEN OBTENHASUBMETAS (PRIMSMETA, NVARS)
                        ELSE BEGIN WRITELN
                                   WRITELN (
'ERRO SINTATICO  ESPERAVA-SE O CORPO OU O PONTO DE FINAL DE CLAUSULA,'
                                           )
                                   WRITE (
'               POREM FOI ENCOTRADO O SIMBOLO  '
                                           )
                                   ESCREVASIMB
                                   ERRO (INPUT, CARACT)
                                   ELIMINECLAUS (CLAUS)
                                   GOTO 888 /* DESVIC PARA O PONTO DE
                                            * RECUPERA  O DE ERROS
                                            * SINTATICOS
                                            */
                             END
                  END
     END


/*****************************************************************
 * OBTENHACLAUS - CORPO DA PROCEDURE
 *****************************************************************/

   BEGIN
   888  /* PONTO DE RECUPERACAO DE ERROS SINTATICOS */
        IF NOT EOF
          THEN BEGIN GC (INPUT, CARACT)
                     REPINTSALVA  = RIBRANCO
                     OBTENHASIMBVALIDO
               END
          ELSE REPINT  = RIFIMARQ
        IF REPINT = RIFIMARQ
          THEN CLAUS  = NIL
          ELSE BEGIN NEW (CLAUS)
                     WITH CLAUS -
                      DO BEGIN NVARS  = 0
                               OBTENHACABECA (CABECA, NVARS)
                               OBTENHACORPO (PRIMSMETA, NVARS)
                               PROXCLAUS  = NIL
                               IF (CABECA )  NIL) AND
                                  (TIPODOPRED (CABECA) )  PREDDEF)
                                 THEN BEGIN WRITELN
                                            WRITELN (
'ERRO SINTATICO  N O   PERMITIDO REDEFINIR PREDICADOS EMBUTIDOS'
                                                    )
```

UNIVERSIDADE FEDERAL DA PARAÍBA
Pró-Reitoria Para Assuntos do Interior
Coordenação Setorial de Pós-Graduação
Rua Aprigio Veloso, 882 - Tel (083) 321-7222-R 355
58.100 - Campina Grande - Paraíba

```
                                                ELIMINECLAUS (CLAUS)
                                                GOTO 888 /* DESVIO PARA O
                                                         * PONTO DE RECU-
                                                         * PERA  O DE ER-
                                                         * ROS SINTATICOS
                                                         */
                                          END
                                 END
                          END
    END


/*****************************************************************
 * PROGRAMA PRINCIPAL - PROCEDURE QUE ARMAZENA CLAUSULAS DEFINIDAS
 *****************************************************************/

  PROCEDURE ARMAZENECLAUSDEF (CLAUSDEF   TAPCLAUS)

    VAR CLAUSAUX   TAPCLAUS

    BEGIN WITH CLAUSDEF - . CABECA - . NOME -
          DO IF GPOCLS = NIL
               THEN BEGIN NEW (GPOCLS)
                          WITH GPOCLS -
                            DO BEGIN TIPOPREDCABEC  = PREDDEF
                                     NARGSCABEC  = ACHENARGS (CLAUS -
                                                              .CABECA)
                                     PRIMCLAUS  = CLAUSDEF
/*                                   WRITELN (
'PROLOGLACET - FOI ARMAZENADA UMA CLAUSULA DEFINIDA EM UM NOVO GRUPO'
                                             )
 */                                END
                    END
               ELSE WITH GPOCLS -
                      DO BEGIN IF NARGSCABEC )  ACHENARGS (CLAUS - .
                                                           CABECA)
                                  THEN BEGIN WRITELN
                                             WRITELN (
'ERRO NO ARMAZENAMENTO DE CLAUSULA DEFINIDA  N O   PERMITIDO CLAUSULAS'
                                                      )
                                             WRITELN (
'QUE TENHAM CABE AS COM MESMO PREDICADO E NUMERO DE ARGUMENTOS DIFE-'
                                                      )
                                             WRITELN (
'RENTES'
                                                      )
                                             ELIMINECLAUS (CLAUSDEF)
                                       END
                                  ELSE BEGIN CLAUSAUX  = PRIMCLAUS
                                             IF CLAUSAUX = NIL
                                               THEN PRIMCLAUS  =
                                                            CLAUSDEF
                                               ELSE WHILE CLAUSAUX - .
                                                          PROXCLAUS )  NIL
                                                    DO CLAUSAUX  =
```

```
                                                        CLAUSAUX - .
                                                          PROXCLAUS
                                                CLAUSAUX - . PROXCLAUS  =
                                                          CLAUSDEF
/*                                              WRITELN (
'PROLOGLACET - FOI ARMAZENADA UMA CLAUSULA DEFINIDA EM GRUPO EXISTENTE'
                                                        )
 */                                     END
                        END
    END


/*****************************************************************
 * PROGRAMA PRINCIPAL - PROCEDURE QUE RESOLVE UMA CLAUSULA META
 *****************************************************************/

  PROCEDURE RESOLVACLAUSMETA (VAR CLAUSMETA   TAPCLAUS)

    LABEL 999  /* RETORNO DE ERROS DETECTADOS DURANTE A RESOLUCAO */


/*****************************************************************
 * RESOLVACLAUSMETA - DEFINI  O DE TIPOS
 *****************************************************************/

   TYPE TPOSICAOARV    = 0..COMPRARVPROVA
        TTIPOELEMARV   = (TIPOPOSICAOARV, TIPOAPSMOUARG, TIPOAPCLAUS)
        TELEMARV       = RECORD CASE TTIPOELEMARV
                                 OF TIPOPOSICAOARV   (POSICAOARV
                                                       TPOSICAOARV)
                                    TIPOAPSMOUARG    (APSMOUARG
                                                       TAPSMOUARG)
                                    TIPOAPCLAUS    (APCLAUS   TAPCLAUS)
                                 END
        TARITMET       = (NCINT, NCREAL)


/*****************************************************************
 * RESOLVACLAUSMETA - DECLARA  O DE VARIAVEIS
 *****************************************************************/

   VAR ARVPROVA                 ARRAY (.0..COMPRARVPROVA.) OF TELEMARV
                                 /* EMBORA FISICAMENTE SEJA UM ARRAY, ESTA
                                  * ESTRUTURA CONT M AS ESTRUTURAS E SUB-
                                  * ESTRUTURAS QUE CONSTITUEM OS NOS DA
                                  * ARVORE DE PROVA NO MODELO CONCEITUAL
                                  * APRESENTADO (VER ...              ),
                                  * ASSIM COMO UMA PILHA QUE CONT M AS
                                  * LIGA  ES A SEREM ELIMINADAS NO CASO DE
                                  * RETROCESSO
                                  */
       POSICAOLIVRE
      ,TOPOPILHALIGSELIM
      ,CONTEXSMATUAL
      ,CONTEXSMTRAB
```

```
      ,CONTEXCLSSENDOTENT
      ,PAISMATUAL
      ,NORETROCESSO              TPOSICAOARV
       CLAUSSENDOTENTADA
      ,PROXCLAUSATENTAR          TAPCLAUS
       SMATUAL
      ,SMDETRAB                  TAPSMOUARG
       UNIFPOSSIVEL              BOOLEAN
```



```
/*****************************************************************
 * RESOLVACLAUSMETA - FUN  ES QUE, DADA A POSI  O DE UMA  ESTRUTURA
 *                    CONCEITUAL , ARMAZENADA EM  ARVPROVA , CALCULAM A
 *                    POSICAO DE ELEMENTOS QUE, CONCEITUALMENTE, FAZEM
 *                    PARTE DA ESTRUTURA
 *****************************************************************/

  FUNCTION PTORETORNONO (NO   TPOSICAOARV)   TPOSICAOARV
    BEGIN PTORETORNONO  = NO - 3
    END

  FUNCTION PROXCLAUSNO (NO   TPOSICAOARV)   TPOSICAOARV
    BEGIN PROXCLAUSNO  = NO - 2
    END

  FUNCTION BASELGELIMNO (NO   TPOSICAOARV)   TPOSICAOARV
    BEGIN BASELGELIMNO  = NO - 1
    END

  FUNCTION PAIDONO (NO   TPOSICAOARV)   TPOSICAOARV
    BEGIN PAIDONO  = NO
    END

  FUNCTION SMETADONO (NO   TPOSICAOARV)   TPOSICAOARV
    BEGIN SMETADONO  = NO + 1
    END

  FUNCTION CONTEXDONO (NO   TPOSICAOARV)   TPOSICAOARV
    BEGIN CONTEXDONO  = NO
    END

  FUNCTION INSTDOCONTEX (CONTEX   TPOSICAOARV
                         INDVAR   TINDVAR)   TPOSICAOARV
    BEGIN INSTDOCONTEX  = CONTEX + 2 * INDVAR
    END

  FUNCTION PROXINST (INST   TPOSICAOARV)   TPOSICAOARV
    BEGIN PROXINST  = INST + 2
    END

  FUNCTION CODINST (INST   TPOSICAOARV)   TPOSICAOARV
    BEGIN CODINST  = INST
    END

  FUNCTION CONTEXINST (INST   TPOSICAOARV)   TPOSICAOARV
```

```
    BEGIN CONTEXINST  = INST + 1
    END

  FUNCTION POSICAOSEGAONO (NO    TPOSICAOARV
                           NVARS   TINDVAR)   TPOSICAOARV
    BEGIN POSICAOSEGAONO  = NO + 2 * NVARS + 2
    END

  FUNCTION POSICAOLIBRETROC (NO   TPOSICAOARV)   TPOSICAOARV
    BEGIN POSICAOLIBRETROC  = NO - 3
    END

  FUNCTION NOVONODETERM (POSICAOLIVRE   TPOSICAOARV)   TPOSICAOARV
    BEGIN NOVONODETERM  = POSICAOLIVRE
    END

  FUNCTION NOVONONAODETERM (POSICAOLIVRE   TPOSICAOARV)   TPOSICAOARV
    BEGIN NOVONONAODETERM  = POSICAOLIVRE + 3
    END


/*****************************************************************
 * RESOLVACLAUSMETA -  PROCEDURE QUE IMPRIME O CONTEUDO DE UMA ESTRUTURA
 *                     TIPO TPREDOUFUNC
 *****************************************************************/

   PROCEDURE ESCREVAPREDOUFUNC (PREDFUNC   TAPPREDOUFUNC
                                CONTEX   TPOSICAOARV )

     FORWARD


/*****************************************************************
 * RESOLVACLAUSMETA/BUSCINSTANCNOGPO/TENTECLAUSEMBUT - PROCEDURE QUE
 *      COM OS QUAIS AS VARIAVEIS EST O INSTANCIADAS.
 *****************************************************************/

   PROCEDURE ESCREVAVALORVAR (INDVAR   TINDVAR
                              CONTEX   TPOSICAOARV)

     VAR CODIGO   TAPSMOUARG
         INSTANC   TPOSICAOARV

     BEGIN IF INDVAR = 0
             THEN WRITE (' ')
             ELSE BEGIN INSTANC  = INSTDOCONTEX (CONTEX,INDVAR)
                        CODIGO  = ARVPROVA(.CODINST(INSTANC).).APSMOUARG
                        IF CODIGO = NIL
                          THEN WRITE ('   ',INDVAR,'   ')
                          ELSE WITH CODIGO -
                                 DO CASE TIPO
                                     OF NINT  WRITE (VALORINTEIRO)
                                        NREAL  WRITE (VALORREAL)
                                        VARIAVEL  ESCREVAVALORVAR
                                                  (INDDAVARIAVEL
```

```
                                                    ,ARVPROVA(.CONTEXINST
                                                              (INSTANC).)
                                                               .POSICAOARV
                                                    )
                                          PREDOUFUNC  ESCREVAPREDOUFUNC
                                                    (APPREDOUFUNC
                                                    ,ARVPROVA(.CONTEXINST
                                                              (INSTANC).)
                                                               .POSICAOARV
                                                    )
                                          CARACSTR  BEGIN WRITELN
                                                          WRITELN (
'ADVERTENCIA  STRING NAO PODE SER INSTANCIADO'                    )
                                                    END
                                    END
                  END
    END


/*****************************************************************
 * RESOLVACLAUSMETA - PROCEDURE QUE ESCREVE UM STRING
 *****************************************************************/

  PROCEDURE ESCREVASTRING (VAR ARGCARAC   TAPSMOUARG)

    BEGIN IF (ARGCARAC - . PROXSMOUARG )  NIL) AND
             (ARGCARAC - . PROXSMOUARG - . TIPO = CARACSTR)
          THEN BEGIN IF ARGCARAC - . VALORCARAC = ' '
                       THEN WRITELN
                       ELSE WRITE (ARGCARAC - . VALORCARAC)
                     ARGCARAC  = ARGCARAC - . PROXSMOUARG
                     ESCREVASTRING (ARGCARAC)
                END
    END


/*****************************************************************
 * RESOLVACLAUSMETA  -  PROCEDURE QUE IMPRIME O CONTEUDO DAS ESTRUTURAS
 *                      DE UMA LISTA DE ARGUMENTOS
 *****************************************************************/

  PROCEDURE ESCREVAARGS (ARG   TAPSMOUARG
                         CONTEX   TPOSICAOARV
                         SEPVIRGULA   BOOLEAN)

    BEGIN UNIFPOSSIVEL  = TRUE
          WITH ARG -
           DO BEGIN CASE TIPO
                      OF NINT  WRITE (VALORINTEIRO)
                         NREAL  WRITE (VALORREAL)
                         VARIAVEL  ESCREVAVALORVAR (INDDAVARIAVEL
                                                          ,CONTEX)
                         PREDOUFUNC  ESCREVAPREDOUFUNC (ARG - .
                                          APPREDOUFUNC, CONTEX)
                         CARACSTR  ESCREVASTRING (ARG)
```

```
                    END
                END
            IF ARG - . PROXSMOUARG ) NIL
              THEN BEGIN IF SEPVIRGULA
                           THEN WRITE (', ')
                         ESCREVAARGS (ARG - . PROXSMOUARG
                                     ,CONTEX, SEPVIRGULA)
                   END
    END

/**********************************************************************
 * RESOLVACLAUSMETA - PROCEDURE QUE CALCULA O VALOR DE UM ARGUMENTO DE
 *                    FUN  O EMBUTIDA.
 **********************************************************************/

  PROCEDURE CALCVALORARG (ARG    TAPSMOUARG
                          CONTEX   TPOSICAOARV
                          VAR TIPOCALC   TARITMET
                          VAR INTCALC   INTEGER
                          VAR REALCALC   REAL)

    FORWARD

/**********************************************************************
 * RESOLVACLAUSMETA - PROCEDURE QUE CALCULA A FUN  O EMBUTIDA  +  (SOMA)
 **********************************************************************/

  PROCEDURE CALCSOMA (ARG1   TAPSMOUARG
                      CONTEX   TPOSICAOARV
                      VAR TIPOCALC   TARITMET
                      VAR INTCALC   INTEGER
                      VAR REALCALC   REAL)

    VAR TIPOCALC1, TIPOCALC2   TARITMET
        INTCALC1, INTCALC2   INTEGER
        REALCALC1, REALCALC2   REAL

    BEGIN CALCVALORARG (ARG1, CONTEX, TIPOCALC1, INTCALC1, REALCALC1)
          CALCVALORARG (ARG1 - . PROXSMOUARG, CONTEX
                                , TIPOCALC2, INTCALC2, REALCALC2)
          IF TIPOCALC1 = NCINT
            THEN IF TIPOCALC2 = NCINT
                   THEN BEGIN TIPOCALC  = NCINT
                              INTCALC  = INTCALC1+INTCALC2
                        END
                   ELSE BEGIN TIPOCALC  = NCREAL
                              REALCALC  = INTCALC1 + REALCALC2
                        END
            ELSE BEGIN TIPOCALC  = NCREAL
                       IF TIPOCALC2 = NCINT
                         THEN REALCALC  = REALCALC1 + INTCALC1
                         ELSE REALCALC  = REALCALC1 + REALCALC2
                 END
```

```
    END

/**************************************************************************
 * RESOLVACLAUSMETA - PROCEDURE QUE CALCULA A FUN  O  -  (SUBTRA  O).
 **************************************************************************/

   PROCEDURE CALCSUBT (ARG1   TAPSMOUARG
                       CONTEX   TPOSICAOARV
                       VAR TIPOCALC   TARITMET
                       VAR INTCALC   INTEGER
                       VAR REALCALC   REAL)

     VAR TIPOCALC1, TIPOCALC2   TARITMET
         INTCALC1, INTCALC2   INTEGER
         REALCALC1, REALCALC2   REAL

     BEGIN IF ARG1 -  .PROXSMOUARG = NIL
           THEN BEGIN CALCVALORARG (ARG1, CONTEX, TIPOCALC
                                    , INTCALC1, REALCALC1)
                      IF TIPOCALC = NCINT
                        THEN INTCALC  = -INTCALC1
                        ELSE REALCALC  = -REALCALC1
                END
           ELSE BEGIN CALCVALORARG (ARG1, CONTEX, TIPOCALC1, INTCALC1
                                                          ,REALCALC1)
                      CALCVALORARG (ARG1 - . PROXSMOUARG, CONTEX
                                    ,TIPOCALC2 ,INTCALC2, REALCALC2)
                      IF TIPOCALC1 = NCINT
                        THEN IF TIPOCALC2 = NCINT
                                THEN BEGIN TIPOCALC  = NCINT
                                           INTCALC  = INTCALC1-INTCALC2
                                     END
                                ELSE BEGIN TIPOCALC  = NCREAL
                                           REALCALC  = INTCALC1 -
                                                       REALCALC2
                                     END
                        ELSE BEGIN TIPOCALC  = NCREAL
                                   IF TIPOCALC2 = NCINT
                                     THEN REALCALC  = REALCALC1 -
                                                      INTCALC2
                                     ELSE REALCALC  = REALCALC1 -
                                                      REALCALC2
                                END
                END
     END

/**************************************************************************
 * RESOLVACLAUSMETA - PROCEDURE QUE CALCULA A FUN  O  *  (MULTIPLICA  O)
 **************************************************************************/

   PROCEDURE CALCMULT (ARG1   TAPSMOUARG
                       CONTEX   TPOSICAOARV
                       VAR TIPOCALC   TARITMET
```

UNIVERSIDADE FEDERAL DA PARAÍBA
Pró-Reitoria Para Assuntos do Interior
Coordenação Setorial de Pós-Graduação
Rua Aprigio Veloso, 882 - Tel (083) 321-7222-R 355
58.100 - Campina Grande - Paraíba

```
                        VAR INTCALC   INTEGER
                        VAR REALCALC   REAL)

     VAR TIPOCALC1, TIPOCALC2   TARITMET
         INTCALC1, INTCALC2   INTEGER
         REALCALC1, REALCALC2   REAL

     BEGIN CALCVALORARG (ARG1, CONTEX, TIPOCALC1, INTCALC1, REALCALC1)
           CALCVALORARG (ARG1 - . PROXSMOUARG, CONTEX
                                        ,TIPOCALC2 ,INTCALC2, REALCALC2)
           IF TIPOCALC1 = NCINT
             THEN IF TIPOCALC2 = NCINT
                    THEN BEGIN TIPOCALC  = NCINT
                               INTCALC  = INTCALC1 * INTCALC2
                         END
                    ELSE BEGIN TIPOCALC  = NCREAL
                               REALCALC  = INTCALC1 * REALCALC2
                         END
             ELSE BEGIN TIPOCALC  = NCREAL
                        IF TIPOCALC2 = NCINT
                          THEN REALCALC  = REALCALC1 * INTCALC2
                          ELSE REALCALC  = REALCALC1 * REALCALC2
                  END
     END


/**********************************************************************
 * RESOLVACLAUSMETA - PROCEDURE QUE CALCULA A FUN  O     (DIVIS O
 *                                                          INTEIRA)
 **********************************************************************/

   PROCEDURE CALCDIVINT (ARG1   TAPSMOUARG
                         CONTEX   TPOSICAOARV
                         VAR INTCALC   INTEGER)

     VAR TIPOCALC1, TIPOCALC2   TARITMET
         INTCALC1, INTCALC2   INTEGER
         REALCALC1, REALCALC2   REAL

     BEGIN CALCVALORARG (ARG1, CONTEX, TIPOCALC1, INTCALC1 ,REALCALC1)
           CALCVALORARG (ARG1 - . PROXSMOUARG, CONTEX
                                        ,TIPOCALC2, INTCALC2, REALCALC2)
           IF (TIPOCALC1 )  NCINT) OR (TIPOCALC2 )  NCINT)
             THEN BEGIN WRITELN
                        WRITELN (
'ERRO NA RESOLU$'O DE CLAUSULA META  TENTADO CALCULAR DIVIS O INTEIRA')
                        WRITELN (
'                              COM ARGUMENTOS N O INTEIROS')
                        GOTO 999 /* DESVIO PARA O PONTO DE
                                  * RECUPERA  O DE ERROS NA
                                  * ROTINA RESOLVACLAUSMETA
                                  */
                  END
             ELSE INTCALC  = INTCALC1 DIV INTCALC2
     END
```

```
                    VAR INTCALC   INTEGER
                    VAR REALCALC   REAL)

    VAR TIPOCALC1, TIPOCALC2   TARITMET
        INTCALC1, INTCALC2   INTEGER
        REALCALC1, REALCALC2   REAL

    BEGIN CALCVALORARG (ARG1, CONTEX, TIPOCALC1, INTCALC1, REALCALC1)
          CALCVALORARG (ARG1 - . PROXSMOUARG, CONTEX
                                    ,TIPOCALC2 ,INTCALC2, REALCALC2)
          IF TIPOCALC1 = NCINT
            THEN IF TIPOCALC2 = NCINT
                   THEN BEGIN TIPOCALC  = NCINT
                              INTCALC  = INTCALC1 * INTCALC2
                         END
                   ELSE BEGIN TIPOCALC  = NCREAL
                              REALCALC  = INTCALC1 * REALCALC2
                         END
            ELSE BEGIN TIPOCALC  = NCREAL
                       IF TIPOCALC2 = NCINT
                         THEN REALCALC  = REALCALC1 * INTCALC2
                         ELSE REALCALC  = REALCALC1 * REALCALC2
                 END
    END


/**********************************************************************
 * RESOLVACLAUSMETA - PROCEDURE QUE CALCULA A FUN  O     (DIVIS O
 *                                                          INTEIRA)
 **********************************************************************/

   PROCEDURE CALCDIVINT (ARG1   TAPSMOUARG
                         CONTEX   TPOSICAOARV
                         VAR INTCALC   INTEGER)

    VAR TIPOCALC1, TIPOCALC2   TARITMET
        INTCALC1, INTCALC2   INTEGER
        REALCALC1, REALCALC2   REAL

    BEGIN CALCVALORARG (ARG1, CONTEX, TIPOCALC1, INTCALC1 ,REALCALC1)
          CALCVALORARG (ARG1 - . PROXSMOUARG, CONTEX
                                    ,TIPOCALC2, INTCALC2, REALCALC2)
          IF (TIPOCALC1 )  NCINT) OR (TIPOCALC2 )  NCINT)
            THEN BEGIN WRITELN
                       WRITELN (
'ERRO NA RESOLU$'O DE CLAUSULA META  TENTADO CALCULAR DIVIS O INTEIRA')
                       WRITELN (
'                            COM ARGUMENTOS N O INTEIROS')
                       GOTO 999 /* DESVIO PARA O PONTO DE
                                 * RECUPERA  O DE ERROS NA
                                 * ROTINA RESOLVACLAUSMETA
                                 */
                 END
            ELSE INTCALC  = INTCALC1 DIV INTCALC2
    END
```

```
/**********************************************************************
 * RESOLVACLAUSMETA - PROCEDURE QUE CALCULA A FUNCAO  /  (DIVIS O REAL).
 **********************************************************************/

   PROCEDURE CALCDIVREAL (ARG1   TAPSMOUARG
                          CONTEX   TPOSICAOARV
                          VAR REALCALC   REAL)

     VAR TIPOCALC1, TIPOCALC2   TARITMET
         INTCALC1, INTCALC2   INTEGER
         REALCALC1, REALCALC2   REAL

     BEGIN CALCVALORARG (ARG1, CONTEX, TIPOCALC1, INTCALC1, REALCALC1)
           CALCVALORARG (ARG1 - . PROXSMOUARG, CONTEX
                                    ,TIPOCALC2, INTCALC2, REALCALC2)
           IF TIPOCALC1 = NCINT
             THEN IF TIPOCALC2 = NCINT
                    THEN REALCALC  = INTCALC1 / INTCALC2
                    ELSE REALCALC  = INTCALC1 / REALCALC2
             ELSE IF TIPOCALC2 = NCINT
                    THEN REALCALC  = REALCALC1 / INTCALC2
                    ELSE REALCALC  = REALCALC1 / REALCALC2
     END


/**********************************************************************
 * RESOLVACLAUSMETA - PROCEDURE QUE CALCULA A FUNC O  MOD  (RESTO DA
 *                                                          DIVISAO)
 **********************************************************************/

   PROCEDURE CALCMOD (ARG1   TAPSMOUARG
                      CONTEX   TPOSICAOARV
                      VAR INTCALC   INTEGER)

     VAR TIPOCALC1, TIPOCALC2   TARITMET
         INTCALC1, INTCALC2   INTEGER
         REALCALC1, REALCALC2   REAL

     BEGIN CALCVALORARG (ARG1, CONTEX, TIPOCALC1, INTCALC1, REALCALC1)
           CALCVALORARG (ARG1 - . PROXSMOUARG, CONTEX
                                  , TIPOCALC2, INTCALC2, REALCALC2)
           IF (TIPOCALC1 )  NCINT) OR (TIPOCALC2 )  NCINT)
             THEN BEGIN WRITELN
                        WRITELN (
'ERRO NA RESOLU$'O DE CLAUSULA META  TENTADO CALCULAR A FUN  O  MOD ')
                        WRITELN (
'                              COM ARGUMENTOS N O INTEIROS')
                        GOTO 999 /* DESVIO PARA O PONTO DE
                                  * RECUPERA  O DE ERROS NA
                                  * ROTINA RESOLVACLAUSMETA
                                  */
                  END
             ELSE INTCALC  = INTCALC1 MOD INTCALC2
     END
```

```
/**********************************************************************
 * RESOLVACLAUSMETA - PROCEDURE QUE CALCULA A FUNCAO SINAL, RETORNANDO
 *                    -1, 0 OU 1 SE PARA UM ARGUMENTO NEGATIVO, ZERO OU
 *                    POSITIVO, RESPECTIVAMENTE.
 **********************************************************************/


   PROCEDURE CALCSINAL (ARG1   TAPSMOUARG
                        CONTEX   TPOSICAOARV
                        VAR INTCALC   INTEGER)

     VAR TIPOCALC1   TARITMET
         INTCALC1   INTEGER
         REALCALC1   REAL
         VALOR   REAL

     BEGIN CALCVALORARG (ARG1, CONTEX, TIPOCALC1, INTCALC1, REALCALC1)
           IF TIPOCALC1 = NCINT
             THEN VALOR  = INTCALC1
             ELSE VALOR  = REALCALC1
           IF VALOR = 0
             THEN INTCALC  = 0
             ELSE IF VALOR   0
                    THEN INTCALC  = 1
                    ELSE INTCALC  = -1
     END


/**********************************************************************
 * RESOLVACLAUSMETA - PROCEDURE QUE ENCONTRA O VALOR CALCULADO DE UMA
 *                    VARIAVEL DE UMA FUN  O EMBUTIDA.
 **********************************************************************/

   PROCEDURE CALCVALORVAR (INDICE   TINDVAR
                           CONTEX   TPOSICAOARV
                           VAR TIPOCALC   TARITMET
                           VAR INTCALC   INTEGER
                           VAR REALCALC  REAL)

     VAR CODIGO   TAPSMOUARG
         INSTANC   TPOSICAOARV

    BEGIN INSTANC  = INSTDOCONTEX (CONTEX, INDICE)
          CODIGO  = ARVPROVA(.CODINST(INSTANC).).APSMOUARG
          IF CODIGO = NIL
            THEN BEGIN WRITELN
                       WRITELN (
'ERRO NA RESOLU$'O DE CLAUSULA META  FOI TENTADO AVALIAR UMA FUN  O')
                       WRITELN (
'          ARITMETICA COM UMA VARIAVEL AINDA NAO INSTANCIADA')
                       GOTO 999 /* DESVIO PARA O PONTO DE RECUPERA  O
                                 * DE ERROS NA ROTINA RESOLVACLAUSMETA
```

```
                                        */
                    END
               ELSE CALCVALORARG (CODIGO, ARVPROVA(.CONTEXINST(INSTANC).)
                                    .POSICAOARV,TIPOCALC,INTCALC,REALCALC)
     END


/*****************************************************************
 * RESOLVACLAUSMETA - PROCEDURE QUE CALCULA UMA FUN  C EMBUTIDA
 *****************************************************************/

   PROCEDURE CALCFUNCEMBUT (FUNC    TAPPREDOUFUNC
                            CONTEX   TPOSICAOARV
                            VAR TIPOCALC    TARITMET
                            VAR INTCALC    INTEGER
                            VAR REALCALC    REAL)

     BEGIN CASE TIPODEFUNC (FUNC)
           OF FUNCSOMA  CALCSOMA (FUNC - . PRIMARG, CONTEX, TIPOCALC
                                                , INTCALC, REALCALC )
              FUNCSUBT  CALCSUBT (FUNC - . PRIMARG, CONTEX, TIPOCALC
                                                , INTCALC, REALCALC )
              FUNCMULT  CALCMULT (FUNC - . PRIMARG, CONTEX, TIPOCALC
                                                , INTCALC, REALCALC )
              FUNCDIVINT  BEGIN TIPOCALC  = NCINT
                                CALCDIVINT (FUNC - . PRIMARG, CONTEX
                                                        , INTCALC)
                          END
              FUNCDIVREAL  BEGIN TIPOCALC  = NCREAL
                                 CALCDIVREAL (FUNC - . PRIMARG, CONTEX
                                                        , REALCALC)
                           END
              FUNCMOD  BEGIN TIPOCALC  = NCINT
                             CALCMOD (FUNC - . PRIMARG, CONTEX
                                                        , INTCALC)
                       END
              FUNCSINAL  BEGIN TIPOCALC  = NCINT
                               CALCSINAL (FUNC - . PRIMARG, CONTEX
                                                        , INTCALC)
                         END
              FUNCCALC  CALCVALORARG (FUNC - . PRIMARG, CONTEX
                                          ,TIPOCALC,INTCALC,REALCALC)
              FUNCDEF  BEGIN WRITELN
                             WRITELN (
'ERRO NA RESOLU$'O DE CLAUSULA META  FOI TENTADO AVALIAR UMA FUN  O')
                             WRITELN (
'                                N O ARITMETICA')
                             GOTO 999 /* DESVIO PARA O PONTO DE RECUPE-
                                      * RA  O DE ERROS NA ROTINA
                                      * RESOLVACLAUSMETA
                                      */
                       END
           END
     END
```

UNIVERSIDADE FEDERAL DA PARAÍBA
Pró-Reitoria Para Assuntos do Interior
Coordenação Setorial de Pós-Graduação
Rua Aprigio Veloso, 882 - Tel (083) 321-7222-R 355
58.100 - Campina Grande - Paraíba

```
/**************************************************************
 * RESOLVACLAUSMETA - DEFINI  O DA PROCEDURE CALCVALORARG, DEFINIDA
 *                    ANTERIORMENTE COM  FORWARD .
 **************************************************************/

   PROCEDURE CALCVALORARG

     BEGIN WITH ARG -
           DO CASE TIPO
               OF NINT  BEGIN TIPOCALC  = NCINT
                              INTCALC  = VALORINTEIRO
                        END
                  NREAL  BEGIN TIPOCALC  = NCREAL
                               REALCALC  = VALORREAL
                         END
                  VARIAVEL  CALCVALORVAR (INDDAVARIAVEL, CONTEX
                                          ,TIPOCALC, INTCALC, REALCALC)
                  PREDOUFUNC  CALCFUNCEMBUT (APPREDOUFUNC, CONTEX
                                          ,TIPOCALC,INTCALC,REALCALC)
                  CARACSTR  BEGIN WRITELN
                                  WRITELN (
'ERRO NA RESOLU$'O DE CLAUSULA META  STRING COMO ARGUMENTO DE FUN  O')
                                  WRITELN (
'                                 ARITMETICA EM AVALIA  O')
                                  GOTO 999 /* DESVIO PARA O PONTO DE
                                           * RECUPERA  O DE ERROS NA
                                           * ROTINA RESOLVACLAUSMETA
                                           */
                            END
               END
     END

/**************************************************************
 * RESOLVACLAUSMETA  -  DEFINI  O DA PROCEDURE ESCREVAPREDOUFUNC DEFI-
 *                      NIDA ANTERIORMENTE COM  FORWARD
 **************************************************************/

   PROCEDURE ESCREVAPREDOUFUNC

     VAR I   TCOMPRSIMB


/**************************************************************
 * ESCREVAPREDFUNC - PROCEDURE QUE ESCREVE O VALOR CALCULADO DE UMA
 *                   VARIAVEL.
 **************************************************************/

   PROCEDURE ESCREVAVARCALC (INDICE   TINDVAR
                             CONTEX   TPOSICAOARV)

     FORWARD
```

```
/**************************************************************************
 * ESCREVAPREDOUFUNC - PROCEDURE QUE ESCREVE O VALOR CALCULADO DE UMA
 *                     FUN AO EMBUTIDA.
 **************************************************************************/

   PROCEDURE ESCREVAVALORCALC (ARGCALC    TAPSMOUARG
                               CONTEX   TPOSICAOARV)

   VAR TIPOCALC    TARITMET
       INTCALC    INTEGER
       REALCALC    REAL

     BEGIN WITH ARGCALC -
            DO CASE TIPO
                OF NINT  WRITE (VALORINTEIRO)
                   NREAL  WRITE(VALORREAL)
                   VARIAVEL  ESCREVAVARCALC (INDDAVARIAVEL, CONTEX)
                   PREDOUFUNC  BEGIN CALCFUNCEMBUT (APPREDOUFUNC, CONTEX
                                                    ,TIPOCALC, INTCALC
                                                          , REALCALC)
                                     IF TIPOCALC = NCINT
                                       THEN WRITE (INTCALC)
                                       ELSE WRITE (REALCALC)
                               END
                   CARACSTR  BEGIN WRITELN
                                   WRITELN (
'ERRO NA RESOLU$'O DE CLAUSULA META  STRING PARTICIPANDO DE FUN  O')
                                   WRITELN (
'                                    CALCULADA')
                                   GOTO 999 /* DESVIC PARA O PONTO DE
                                            * RECUPERA  O DE ERROS NA
                                            * ROTINA RESOLVACLAUSMETA
                                            */
                             END
                END
           IF ARGCALC - . PROXSMOUARG )  NIL
             THEN ESCREVAVALORCALC (ARGCALC - . PROXSMOUARG, CONTEX)
     END


/**************************************************************************
 * ESCREVAPREDFUNC - DEFINI AO DA PROCEDURE ESCREVAVARÇALC, DEFINIDA
 *                   ANTERIORMENTE COM  FORWARD
 **************************************************************************/

   PROCEDURE ESCREVAVARCALC

     VAR CODIGO    TAPSMOUARG
         INSTANC    TPOSICAOARV

     BEGIN INSTANC  = INSTDOCONTEX (CONTEX, INDICE)
           CODIGO  = ARVPROVA(.CODINST(INSTANC).).APSMOUARG
           IF CODIGO = NIL
             THEN BEGIN WRITELN
                        WRITELN (
```

```
'ERRO NA RESOLU$'O DE CLAUSULA META  FOI TENTADO AVALIAR UMA FUN  O')
                        WRITELN (
'       ARITMETICA COM UMA VARIAVEL AINDA N O INSTANCIADA.')
                        GOTO 999 /* DESVIO PARA O PONTO DE RECUPERA  O
                                  * DE ERROS SINTATICOS NA ROTINA
                                  * RESOLVACLAUSMETA
                                  */
                END
            ELSE ESCREVAVALORCALC (CODIGO,ARVPROVA(.CONTEXINST(INSTANC)
                                                    .).POSICAOARV)
    END


/**************************************************************************
 * ESCREVAPREDFUNC - CORPO DA PROCEDURE
 **************************************************************************/

    BEGIN IF (TIPODEFUNC (PREDFUNC) = FUNCCALC) AND
             (PREDFUNC - . PRIMARG )  NIL)
          THEN ESCREVAVALORCALC (PREDFUNC - . PRIMARG, CONTEX)
          ELSE BEGIN WITH PREDFUNC - . NOME -
                    DO FOR I  = 1 TO COMPRNOME
                        DO WRITE (NOME(.I.))
                    IF PREDFUNC - . PRIMARG )  NIL
                      THEN BEGIN WRITE (' (')
                                 ESCREVAARGS (PREDFUNC - . PRIMARG
                                                , CONTEX ,TRUE)
                                 WRITE (')')
                           END
               END
    END


/**************************************************************************
 * RESOLVACLAUSMETA - PROCEDURE QUE COMPLETA A CRIA  O DE UM NOVO NO
 **************************************************************************/

 PROCEDURE COMPLETECRIACAONO (VAR NOEMCRIACAO   TPOSICAOARV)

   VAR INDVAR   TINDVAR
       INSTANC   TPOSICAOARV

   BEGIN ARVPROVA (.PAIDONO(NOEMCRIACAO).).POSICAOARV  = PAISMATUAL
         ARVPROVA (.SMETADONO(NOEMCRIACAO).).APSMOUARG  = SMATUAL
         INSTANC  = INSTDOCONTEX(CONTEXDONO(NOEMCRIACAO),1)
         FOR INDVAR  = 1 TO CLAUSSENDOTENTADA - . NVARS
           DO BEGIN ARVPROVA (.CODINST(INSTANC).).APSMOUARG  = NIL
                    ARVPROVA (.CONTEXINST(INSTANC).).POSICAOARV  = 0
                    INSTANC  = PROXINST (INSTANC)
              END
         POSICAOLIVRE  = POSICAOSEGAONO (NOEMCRIACAC,CLAUSSENDOTENTADA
                                                      - . NVARS)
   END
```

```
/**********************************************************************
 * RESOLVACLAUSMETA - PROCEDURE QUE CRIA UM NOVO NO DETERMINISTICO, O
 *                    QUAL PASSA A SER O PAI DA SUBMETA ATUAL
 **********************************************************************/

  PROCEDURE CRIENODETERM

    VAR NOEMCRIACAO   TPOSICAOARV

    BEGIN NOEMCRIACAO  = NOVONODETERM (POSICAOLIVRE)
          COMPLETECRIACAONO (NOEMCRIACAO)
          PAISMATUAL   = NOEMCRIACAO
    END


/**********************************************************************
 * RESOLVACLAUSMETA - PROCEDURE QUE CRIA UM NO N O DETERMINISTICO NA
 *      ARVORE DE PROVA, O QUAL PASSA A SER O PAI DA SUBMETA ATUAL E O
 *      NO PARA RETROCESSO.
 **********************************************************************/

      PROCEDURE CRIENONAODETERM

        VAR NOEMCRIACAO   TPOSICAOARV

        BEGIN NOEMCRIACAO  = NOVONONAODETERM (POSICAOLIVRE)
              ARVPROVA (.PTORETORNONO(NOEMCRIACAO).).POSICAOARV  =
                                                    NORETROCESSO
              ARVPROVA (.PROXCLAUSNO(NOEMCRIACAO).).APCLAUS  =
                                                  PROXCLAUSATENTAR
              ARVPROVA (.BASELGELIMNO(NOEMCRIACAO).).POSICAOARV  =
                                                  TOPOPILHALIGSELIM
              COMPLETECRIACAONO (NOEMCRIACAO)
              PAISMATUAL  = NOEMCRIACAO
              NORETROCESSO  = NOEMCRIACAO
        END


/**********************************************************************
 * RESOLVACLAUSMETA - PROCEDURE QUE DISRREFERENCIA UMA VARIAVEL
 **********************************************************************/

  PROCEDURE DISREFERENCIE (VAR CODIGO   TAPSMOUARG
                           VAR CONTEX   TPOSICAOARV)

    VAR INSTANC   TPOSICAOARV
        INDICE   TINDVAR

    BEGIN IF CODIGO - . TIPO = VARIAVEL
          THEN BEGIN INDICE  = CODIGO - . INDDAVARIAVEL
                     INSTANC  = INSTDOCONTEX (CONTEX,INDICE)
                     WHILE (CODIGO - . TIPO = VARIAVEL) AND
                           (INDICE )  0) AND
                           (ARVPROVA (.CODINST(INSTANC).).APSMOUARG )
                                                        NIL)
```

```
                    DO BEGIN CODIGO  = ARVPROVA(.CODINST(INSTANC).).
                                                          APSMOUARG
                             CONTEX  = ARVPROVA (.CONTEXINST(INSTANC)
                                                        .).POSICAOARV
                             IF CODIGO - . TIPO = VARIAVEL
                  THEN BEGIN INDICE  = CODIGO - . INDDAVARIAVEL
                             INSTANC  = INSTDOCONTEX (CONTEX,INDICE)
                       END
                       END
            END
    END

/**********************************************************************
 * RESOLVACLAUSMETA - PROCEDURE QUE ACHA A SUBMETA DE TRABALHO
 **********************************************************************/

  PROCEDURE ACHESMDETRAB

    BEGIN SMDETRAB  = SMATUAL
          CONTEXSMTRAB  = CONTEXSMATUAL
          DISREFERENCIE (SMDETRAB, CONTEXSMTRAB)
          CASE SMDETRAB - . TIPO
            OF NINT, NREAL  BEGIN WRITELN
                                  WRITELN (
'ERRO NA RESOLU$'O DE CLAUSULA META  SUBMETA N O PODE SER UM NUMERO'
                                          )
          GOTO 999  /* DESVIA PARA O  PONTO DE RECUPERA  O DE ERROS
            * NA ROTINA  RESOLVACLAUSMETA .
            */
                              END
         VARIAVEL  BEGIN WRITELN
                         WRITELN (
'ERRO NA RESOLU$'O DE CLAUSULA META  META-VARIAVEL NAC INSTANCIADA FOI'
                                  )
                         WRITELN (
'                                 OBTIDA NA CLAUSULA META OU EM AL-'
                                  )
                         WRITELN (
'                                 GUMA DAS CLAUSULAS UTILIZADAS NA'
                                  )
                         WRITELN (
'                                 RESOLUCAO DA MESMA'
                                  )
                         GOTO 999 /* DESVIO PARA O PONTO DE RECUPRA  O
                                   * DE ERROS NA ROTINA RESOLVACLSMETA
                                   */
                  END
          PREDOUFUNC
          END
    END

/**********************************************************************
 * RESOLVACLAUSMETA - PROCEDURE QUE BUSCA UMA INSTANCIA NO GRUPO DE
```

```
*                    CLAUSULAS DA SUBMETA ATUAL. SE N O FOR ENCONTRADA
*                    UMA CLAUSULA QUE POSSA INSTANCIAR A SUBMETA ATUAL,
*                    A SUBMETA ATUAL PASSARA A SER A SUBMETA DO RETRO-
*                    CESSO, ASSIM,  JA TENDO SIDO DESFEITA A INSTANCIA
*                    ANTERIOR, S O TENTADAS OUTRAS CLAUSULAS DO GRUPO.
*************************************************************************/

 PROCEDURE BUSQINSTANCNOGPO

   VAR GPOATUAL   TAPGPOCLS


/*************************************************************************
 * RESOLVECLAUSMETA/BUSQINSTANCNOGPO - PROCEDURE QUE FAZ O RETROCESSO
 *                                     (BACKTRACKING) APOS UMA FALHA
 *************************************************************************/

   PROCEDURE RETROCEDA


/*************************************************************************
 * RESOLVACLAUSMETA/BUSQINSTANCNOGPO/RETROCEDA - PROCEDURE QUE ELIMINA
 *                                               LIGA  ES FALHAS, OU
 *                                               SEJA, FEITAS POR INS-
 *                                               TANCIA OES ELIMINADAS
 *                                               NO RETROCESSO.
 *************************************************************************/

     PROCEDURE ELIMINELIGSFALHAS (NO   TPOSICAOARV)

       VAR INSTAANULAR
          ,BASEELIM   TPOSICAOARV

       BEGIN BASEELIM  = ARVPROVA (.BASELGELIMNO(NO).).POSICAOARV
             WHILE TOPOPILHALIGSELIM ) BASEELIM
               DO BEGIN TOPOPILHALIGSELIM  = TOPOPILHALIGSELIM + 1
                        INSTAANULAR  = ARVPROVA (.TOPOPILHALIGSELIM.)
                                                          .POSICAOARV
                        ARVPROVA (.CODINST(INSTAANULAR).).APSMOUARG  =
                                                                NIL
                        ARVPROVA (.CONTEXINST(INSTAANULAR).).POSICAOARV
                                                                = 0
                  END
       END


/*************************************************************************
 * RESOLVACLAUSMETA/BUSQINSTANCNOGPO/RETROCEDA - CORPO DA PROCEDURE
 *************************************************************************/

     BEGIN SMATUAL  = ARVPROVA (.SMETADONO(NORETROCESSO).).APSMOUARG
           PAISMATUAL  = ARVPROVA(.PAIDONO(NORETROCESSO).).POSICAOARV
           CONTEXSMATUAL  = ARVPROVA (.CONTEXDONO(NORETROCESSO).).
                                                          POSICAOARV

           ACHESMDETRAB
```

```
            PROXCLAUSATENTAR  = ARVPROVA (.PROXCLAUSNC(NORETROCESSO).).
                                                              APCLAUS
            ELIMINELIGSFALHAS (NORETROCESSO)
            POSICAOLIVRE  = POSICAOLIBRETROC (NORETROCESSO)
            NORETROCESSO  = ARVPROVA (.PTORETORNONO(NORETROCESSO).).
                                                          POSICAOARV
      END


/*****************************************************************
 * RESOLVACLAUSMETA/BUSQINSTANCNOGPO - PROCEDURE QUE TENTA INSTANCIAR
 *                                     UMA SUBMETA CUJO PREDICADO TENHA
 *                                     SIDO RECONHECIDO COMO EMBUTIDO
 *                                     COM A CLAUSULA EMBUTIDA CORRES-
 *                                     PONDENTE.
 *****************************************************************/

    PROCEDURE TENTECLAUSEMBUT

      VAR TIPOPRED   TTIPOPRED

      BEGIN TIPOPRED  = TIPODOPRED (PROXCLAUSATENTAR - . CABECA)
            CASE TIPOPRED
             OF PREDWRITE  BEGIN ESCREVAARGS (SMDETRAB - .
                                              APPREDCUFUNC - . PRIMARG
                                              , CONTEXSMTRAB, FALSE )
                                 IF SMDETRAB - . APPREDCUFUNC - . NOME =
                                                PREDSEMBUT(.PREDWRITEFALHA.)
                                   THEN UNIFPOSSIVEL  = FALSE
                                   ELSE UNIFPOSSIVEL  = TRUE
                                 PROXCLAUSATENTAR  = NIL
                                 SMATUAL  = SMATUAL - . PROXSMOUARG
                           END
                PREDWRITEFALHA  BEGIN UNIFPOSSIVEL  = TRUE
                                      PROXCLAUSATENTAR  = PREDSEMBUT (.
                                         PREDWRITE.) - . GPCCLS - .
                                                              PRIMCLAUS
                                      CRIENCNAODETERM
                                      SMATUAL  = NIL
                                END
                PREDCORTE  BEGIN WHILE NORETROCESSO  = PAISMATUAL
                                   DO NORETROCESSO  = ARVPROVA (.
                                            PTORETORNONO (NORETROCESSO)
                                                          .).POSICAOARV
                                 UNIFPOSSIVEL  = TRUE
                                 SMATUAL  = SMATUAL - .PROXSMOUARG
                           END
            END
      END


/*****************************************************************
 * RESOLVACLAUSMETA/BUSQINSTANCNOGPO - PROCEDURE QUE TENTA INSTANCIAR
 *                                     UMA SUBMETA COM UMA CLAUSULA DE-
 *                                     FINIDA DO GRUPO.
 *****************************************************************/
```

```
    PROCEDURE TENTECLAUSDEF


/**********************************************************************
 * RESOLVACLAUSMETA/BUSQINSTANCNOGPO/TENTECLAUSDEF - PROCEDURE QUE
 *       TENTA UNIFICAR SUBMETA COM CLAUSULA
 **********************************************************************/

      PROCEDURE TENTEUNIFSMCOMCLS


/**********************************************************************
 * RESOLVACLAUSMETA/BUSQINSTANCNOGPO/TENTECLAUSDEF/TENTEUNIFSMCOMCLAUS
 *       PROCEDURE EM QUE, DADOS OS ARGUMENTOS DE DOIS PREDICADOS, TEN-
 *       TA UNIFICAR OS ARGUMENTOS CORRESPONDENTES.
 **********************************************************************/

        PROCEDURE TENTEUNIFARGS (     PRIMARG1    TAPSMCUARG
                                      CONTEX1    TPOSICACARV
                                      PRIMARG2    TAPSMCUARG
                                      CONTEX2    TPOSICACARV)

          FORWARD


/**********************************************************************
 * RESOLVACLAUSMETA/BUSQINSTANCNOGPO/TENTECLAUSDEF/TENTEUNIFSMCOMCLAUS
 *       PROCEDURE QUE TENTA UNIFICAR FUN  O COM FUN  O
 **********************************************************************/

    PROCEDURE TENTEUNIFFUNCCOMFUNC (FUNC1, FUNC2    TAPPREDOUFUNC
                                    VAR CONTEX1, CONTEX2
                                                  TPOSICAOARV)

       BEGIN IF FUNC1 - . NOME )  FUNC2 - . NOME
               THEN UNIFPOSSIVEL   = FALSE
               ELSE IF (FUNC1 - . PRIMARG = NIL) AND
                       (FUNC2 - . PRIMARG = NIL)

                     THEN UNIFPOSSIVEL   = TRUE
                     ELSE TENTEUNIFARGS (FUNC1 - . PRIMARG, CONTEX1
                                        ,FUNC2 - . PRIMARG, CONTEX2)
       END


/**********************************************************************
 * RESOLVACLAUSMETA/BUSQINSTANCNOGPO/TENTECLAUSDEF/TENTEUNIFSMCOMCLAUS
 *       DEFINI  O DA PROCEDURE TENTEUNIFARGS, (DEFINIDA ANTERIORMENTE
 *       COM  FORWARD ).
 **********************************************************************/

   PROCEDURE TENTEUNIFARGS
```

```
    VAR ARG1, ARG2   TAPSMOUARG

/**************************************************************
 * RESOLVACLAUSMETA/BUSQINSTANCNOGPO/TENTECLAUSDEF/TENTEUNIFSMCOMCLAUS
 *      /TENTEUNIFARGS - PROCEDURE QUE TENTA UNIFICAR DOIS ARGUMENTOS.
 **************************************************************/

   PROCEDURE TENTEUNIFARGCOMARG (ARG1, ARG2   TAPSMOUARG
                                 CONTEX1, CONTEX2   TPOSICACARV)

     VAR TIPOCALC1, TIPOCALC2   TARITMET
         INTCALC1, INTCALC2   INTEGER
         REALCALC1,REALCALC2   REAL


/**************************************************************
 * RESOLVACLAUSMETA/BUSQINSTANCNOGPO/TENTECLAUSDEF/TENTEUNIFSMCOMCLAUS
 *      /TENTEUNIFARGS/TENTEUNIFARGCOMARG - PROCEDURE QUE INSTANCIA UMA
 *      VARIAVEL, OU SEJA, CRIA UMA LIGA  O ENTRE A VARIAVEL E SEU TER-
 *      MO ASSOCIADO.
 **************************************************************/

          PROCEDURE INSTANCIE (VAR VARIAV   TAPSMOUARG
                               VAR CONTEXVAR   TPOSICAOARV
                               TERMOASSOC   TAPSMOUARG
                               CONTEXTASSOC   TPOSICACARV)

            VAR INDICE   TINDVAR
                INSTANC   TPOSICAOARV

            BEGIN INDICE  = VARIAV - . INDDAVARIAVEL
                  IF INDICE )  0 /* SE N 0 FOR VARIAVEL ANONIMA */
                    THEN BEGIN INSTANC  = INSTDOCONTEX (CONTEXVAR,
                                                        INDICE)
                               ARVPROVA (.CODINST(INSTANC).).APSMOUARG
                                                        = TERMOASSOC
                               ARVPROVA (.CONTEXINST(INSTANC).)
                                         .POSICAOARV  = CONTEXTASSOC
                               IF INSTANC ) NORETROCESSO
                                 THEN /* A POSI AO DESTA INSTANCIA
                                       * PRECISA SER GUARDADA NA PILHA
                                       * DE LIGA  ES A ELIMINAR.
                                       */
                                  BEGIN ARVPROVA (.TOPOPILHALIGSELIM.)
                                                .POSICAOARV  = INSTANC
                                        TOPOPILHALIGSELIM  =
                                                TOPOPILHALIGSELIM - 1
                                        IF TOPOPILHALIGSELIM )=
                                                        POSICAOLIVRE
                                          THEN BEGIN WRITELN
                                                     WRITELN (
'ERRO NA RESOLUCAO DE CLAUS META  A AREA DE MEMORIA RESERVADA PARA A'
                                                             )
                                                     WRITELN (
```

```
                                ARVORE DE PROVA ESGOTOU'
                                                        )
                                                  GOTO 999
                                /* DESVIO PARA O PONTO DE RECUPERACAO
                                 * DE ERROS NA ROTINA RESOLVACLAUSMETA
                                 */
                                                  END
                                    END
                            END
                        UNIFPOSSIVEL  = TRUE
                END

/*****************************************************************
 * RESOLVACLAUSMETA/BUSQINSTANCNOGPO/TENTECLAUSDEF/TENTEUNIFSMCOMCLAUS
 *      /TENTEUNIFARGS/TENTEUNIFARGCOMARG - PROCEDURE QUE INSTANCIA UMA
 *      VARIAVEL COM OUTRA VARIAVEL NAO INSTANCIADA.
 *****************************************************************/

        PROCEDURE INSTANCVARCOMVAR (VAR VARIAV1, VARIAV2  TAPSMOUARG
                                    VAR CONTEX1,CONTEX2 TPOSICAOARV)

          BEGIN IF (VARIAV1 - . INDDAVARIAVEL )  0) AND
                   (VARIAV2 - . INDDAVARIAVEL )  0)
                   /* SE UMA DAS VARIAVEIS FOR ANONIMA, NAO
                    * NECESSARIO INSTANCIAR.
                    */
                  THEN IF CONTEX1 ) CONTEX2
                          THEN INSTANCIE (VARIAV2, CONTEX2
                                         ,VARIAV1, CONTEX1)
                          ELSE IF CONTEX1   CONTEX2
                                  THEN INSTANCIE (VARIAV1, CONTEX1
                                                 ,VARIAV2, CONTEX2)
                                  ELSE IF VARIAV1 )  VARIAV2
                                          THEN INSTANCIE (VARIAV1
                                                         ,CONTEX1
                                                         ,VARIAV2
                                                         ,CONTEX2)
                       UNIFPOSSIVEL  = TRUE
                END

/*****************************************************************
 * RESOLVACLAUSMETE/BUSQINSTANCNOGPO/TENTECLAUSDEF/TENTEUNIFSMCOMCLAUS
 *      /TENTEUNIFARGS/TENTEUNIFARGCOMARG - CORPO DA PROCEDURE
 *****************************************************************/

     BEGIN DISREFERENCIE (ARG1, CONTEX1)
           DISREFERENCIE (ARG2, CONTEX2)
           CASE ARG1 - . TIPO
            OF NINT  CASE ARG2 - . TIPO
                      OF NINT  UNIFPOSSIVEL  = (ARG1 - . VALORINTEIRO =
                                                ARG2 - . VALORINTEIRO)
                         VARIAVEL  INSTANCIE (ARG2, CONTEX2
                                             ,ARG1, CONTEX1)
```

```
                NREAL  UNIFPOSSIVEL  = FALSE
                PREDOUFUNC
                    IF TIPODEFUNC (ARG2 - . APPREDOUFUNC)
                                                  = FUNCCALC
                      THEN BEGIN CALCVALORARG(ARG2, CONTEX2
                                 ,TIPOCALC2,INTCALC2,REALCALC2)
                                 IF TIPOCALC2 = NCINT
                                   THEN UNIFPOSSIVEL  =
                                       (ARG1 - . VALORINTEIRO
                                                  = INTCALC2)
                                   ELSE UNIFPOSSIVEL  = FALSE
                           END
                      ELSE UNIFPOSSIVEL  = FALSE
            END
NREAL   CASE ARG2 - . TIPO
            OF NREAL  UNIFPOSSIVEL  = (ARG1 - . VALORREAL =
                                       ARG2 - . VALORREAL)
               VARIAVEL  INSTANCIE (ARG2, CONTEX2
                                   ,ARG1, CONTEX1)
               NINT  UNIFPOSSIVEL  = FALSE
               PREDOUFUNC
                 IF TIPODEFUNC (ARG2 - . APPREDOUFUNC) =
                                                  FUNCCALC
                  THEN BEGIN CALCVALORARG (ARG2, CONTEX2
                             ,TIPOCALC2,INTCALC2,REALCALC2)
                           IF TIPOCALC2 = NCREAL
                             THEN UNIFPOSSIVEL  = (ARG1 -
                                  .VALORREAL = REALCALC2)
                             ELSE UNIFPOSSIVEL  = FALSE
                       END
                  ELSE UNIFPOSSIVEL  = FALSE
            END
VARIAVEL  CASE ARG2 - . TIPO
            OF NINT, NREAL, PREDOUFUNC
                                 INSTANCIE (ARG1, CONTEX1
                                           ,ARG2, CONTEX2)
               VARIAVEL  INSTANCVARCOMVAR (ARG1, ARG2
                                          ,CONTEX1,CONTEX2)
            END
PREDOUFUNC
    IF TIPODEFUNC (ARG1 - . APPREDOUFUNC) )  FUNCCALC
      THEN  CASE ARG2 - . TIPO
              OF PREDOUFUNC  TENTEUNIFFUNCCOMFUNC
                                 (ARG1 - . APPREDOUFUNC
                                 ,ARG2 - . APPREDOUFUNC
                                 ,CONTEX1, CONTEX2 )
                 VARIAVEL  INSTANCIE (ARG2, CONTEX2
                                     ,ARG1, CONTEX1)
                 NINT, NREAL  UNIFPOSSIVEL  = FALSE
              END
      ELSE BEGIN CALCVALORARG (ARG1, CONTEX1, TIPOCALC1
                                     ,INTCALC1, REALCALC2)
                 CASE ARG2 - . TIPO
                  OF NINT  IF TIPOCALC1 = NCINT
                             THEN UNIFPOSSIVEL  = (ARG2 -
```

```
                                        .VALORINTEIRO = INTCALC1)
                                    ELSE UNIFPOSSIVEL  = FALSE
                                NREAL  IF TIPOCALC1 = NCREAL
                                        THEN UNIFPOSSIVEL  = (ARG2
                                        - .VALORREAL = REALCALC1)
                                        ELSE UNIFPOSSIVEL  = FALSE
                                VARIAVEL  INSTANCIE (ARG2, CONTEX2
                                                    ,ARG1, CONTEX1)
                                PREDOUFUNC
                                 IF TIPODEFUNC (ARG2 - .APPREDOUFUNC)
                                                    = FUNCCALC
                                   THEN BEGIN CALCVALORARG (ARG2
                                              ,CONTEX2, TIPOCALC2
                                              ,INTCALC2, REALCALC2)
                                         IF TIPOCALC1 = TIPOCALC2
                                           THEN IF TIPOCALC1 =
                                                        NCINT
                                                THEN UNIFPOSSIVEL
                                                     = (INTCALC1
                                                     = INTCALC2)
                                                ELSE UNIFPOSSIVEL
                                                     = (REALCALC1
                                                     = REALCALC2)
                                           ELSE UNIFPOSSIVEL  =
                                                        FALSE
                                        END
                                   ELSE UNIFPOSSIVEL  = FALSE
                            END
                    END
        END
    END

/**********************************************************************
 * RESOLVACLAUSMETA/BUSQINSTANCNOGPO/TENTECLAUSDEF/TENTEUNIFSMCOMCLAUS
 *      /TENTEUNIFARGS - CORPO DA PROCEDURE
 **********************************************************************/

        BEGIN ARG1  = PRIMARG1
              ARG2  = PRIMARG2
              UNIFPOSSIVEL  = TRUE
              WHILE (ARG1 )  NIL) AND (ARG2 )  NIL) AND UNIFPOSSIVEL
                DO BEGIN TENTEUNIFARGCOMARG (  ARG1 ,   ARG2
                                             ,CONTEX1, CONTEX2)
                         ARG1  = ARG1 - . PROXSMOUARG
                         ARG2  = ARG2 - . PROXSMOUARG
                   END
              IF UNIFPOSSIVEL
                THEN UNIFPOSSIVEL  = (ARG1 = ARG2)
        END

/**********************************************************************
 * RESOLVACLAUSMETA/BUSQINSTANCNOGPO/TENTECLAUSDEF/TENTEUNIFSMCOMCLAUS
        CORPO DA PROCEDURE
```

```
*****************************************************************/

        BEGIN IF (SMDETRAB - . APPREDOUFUNC - . PRIMARG = NIL) AND
                 (CLAUSSENDOTENTADA - . CABECA - . PRIMARG = NIL)
              THEN UNIFPOSSIVEL  = TRUE
              ELSE TENTEUNIFARGS (SMDETRAB - . APPREDOUFUNC - .PRIMARG
                                 ,CONTEXSMTRAB
                                 ,CLAUSSENDOTENTADA - . CABECA - .
                                                          PRIMARG
                                 ,CONTEXCLSSENDOTENT)
        END


/*****************************************************************
 * RESOLVACLAUSMETA/BUSQINSTANCNOGPO/TENTECLAUSDEF - CORPO DA PROCEDURE
 *****************************************************************/

      BEGIN IF PROXCLAUSATENTAR = NIL
               THEN UNIFPOSSIVEL  = FALSE
               ELSE BEGIN CLAUSSENDOTENTADA  = PROXCLAUSATENTAR
                          PROXCLAUSATENTAR  = PROXCLAUSATENTAR - .
                                                          PROXCLAUS
                          IF PROXCLAUSATENTAR = NIL
                            THEN CRIENODETERM
                            ELSE CRIENONAODETERM
                          IF POSICAOLIVRE  = TOPOPILHALIGSELIM
                            THEN BEGIN WRITELN
                                       WRITELN (
'ERRO NA RESOLU$'O DE CLAUS META  A AREA DE MEMORIA RESERVADA PARA A '
                                               )
                                       WRITELN (
  '                                 ARVORE DE PROVA ESGOTOU'
                                               )
                                       GOTO 999
                                  /* DESVIO PARA O  PONTO DE RECUPERA  O
                                   * DE ERROS NA ROTINA RESOLVACLAUSMETA
                                   */
                                 END
                          CONTEXCLSSENDOTENT  = CONTEXDONO (PAISMATUAL)
                          TENTEUNIFSMCOMCLS
                          IF UNIFPOSSIVEL
                           THEN BEGIN SMATUAL  = CLAUSSENDOTENTADA - .
                                                          PRIMSMETA
                                      CONTEXSMATUAL  = CONTEXCLSSENDOTENT
                                END
                    END
      END


/*****************************************************************
 * RESOLVACLAUSMETA/BUSCINSTANCNOGPO - PROCEDURE QUE TENTA INSTANCIAR
 *      UMA SUBMETA COM A PROXIMA CLAUSULA DO GRUPO.
 *****************************************************************/

   PROCEDURE TENTEPROXCLAUS
```

```
    BEGIN IF SMDETRAB - . APPREDOUFUNC - . NOME - . GPOCLS - .
                                                TIPOPREDCABEC )  PREDDEF
            THEN TENTECLAUSEMBUT
            ELSE TENTECLAUSDEF
    END


/**********************************************************************
 * RESOLVACLAUSMETA/BUSQINSTANCNOGPO -CORPO DA PROCEDURE
 **********************************************************************/

   BEGIN GPOATUAL  = SMDETRAB - . APPREDOUFUNC - . NOME - . GPOCLS
         IF GPOATUAL = NIL
           THEN UNIFPOSSIVEL  = FALSE
           ELSE BEGIN PROXCLAUSATENTAR  = GPOATUAL - . PRIMCLAUS
                      TENTEPROXCLAUS
                END
         WHILE (NOT UNIFPOSSIVEL) AND (NORETROCESSO )  0)
          DO BEGIN RETROCEDA
                   TENTEPROXCLAUS
             END
   END


/**********************************************************************
 * RESOLVACLAUSMETA - PROCEDURE QUE BUSCA UMA SUBMETA N O INSTANCIADA.
 *    SE A SUBMETA ATUAL FOR VAZIA, OU SEJA, SE LIGADA AO PAI DA SUBMETA
 *    MAIS RECENTEMENTE INSTANCIADA, N O EXISTIR MAIS NENHUMA SUBMETA A
 *    INSTANCIAR, BUSCA UMA SUBMETA N O INSTANCIADA NO AVO, BISAVO, ...
 *    AT  ENCONTRAR , OU CHEGAR A RAIZ.
 **********************************************************************/

  PROCEDURE BUSQSMNAOINSTANC

   BEGIN WHILE (SMATUAL = NIL) AND (PAISMATUAL )  1)
           DO BEGIN SMATUAL  = ARVPROVA (.SMETADONO(PAISMATUAL).).
                                          APSMOUARG - . PROXSMOUARG
                    PAISMATUAL  = ARVPROVA(.PAIDONO(PAISMATUAL).).
                                                        POSICAOARV
                    CONTEXSMATUAL  = CONTEXDONO (PAISMATUAL)
              END
   END


/**********************************************************************
 * RESOLVACLAUSMETA - CORPO DA PROCEDURE
 **********************************************************************/

  BEGIN PAISMATUAL  = 0
        SMATUAL  = NIL
        POSICAOLIVRE  = 1
        TOPOPILHALIGSELIM  = COMPRARVPROVA
        CLAUSSENDOTENTADA  = CLAUSMETA
        PROXCLAUSATENTAR  = NIL
```

UNIVERSIDADE FEDERAL DA PARAÍBA
Pró-Reitoria Para Assuntos do Interior
Coordenação Setorial de Pós-Graduação
Rua Aprigio Veloso, 882 - Tel. (083) 321-7222-R 355
58.100 - Campina Grande - Paraíba

```
        CRIENODETERM /* CRIA NO RAIZ, RETORNADO EM PAISMATUAL */
        SMATUAL  = CLAUSMETA - . PRIMSMETA
        CONTEXSMATUAL  = CONTEXDONO(PAISMATUAL)
        NORETROCESSO  = 0
        UNIFPOSSIVEL  = TRUE
        WHILE (SMATUAL )  NIL) AND UNIFPOSSIVEL
          DO BEGIN ACHESMDETRAB
                   BUSQINSTANCNOGPO
                   BUSQSMNAOINSTANC
             END
       WRITELN
/*     IF SMATUAL = NIL
         THEN BEGIN WRITELN
                    WRITELN (
'PROLOGLACET - OBTIDO SUCESSO NA RESOLU  O DA CLAUSULA META'
                             )
              END
         ELSE BEGIN WRITELN
                    WRITELN (
'PROLOGLACET - OBTIDA FALHA NA RESOLU  O DA CLAUSULA META'
                             )
              END  */
   599  /* PONTO DE RECUPERA$'O DE ERROS  SEMPRE QUE FOR DETECTADO UM
         * ERRO DURANTE A RESOLU  O DE UMA CLAUSULA META, O PROCESSA-
         * MENTO SERA DESVIADO PARA ESTE PONTO.
         */
        ELIMINECLAUS (CLAUSMETA)
  END


/*****************************************************************
 * PROGRAMAPRINCIPAL - CRIA  O DOS PREDICADOS E FUN  ES EMBUTIDOS
 *****************************************************************/

   PROCEDURE CRIEEMBUTIDOS


/*****************************************************************
 * CRIEEMBUTIDOS - PROCEDURE QUE CRIA UM GRUPO DE CLAUSULAS NO ELEMENTO
 *                 DA TABELA DE SIMBOLOS DE UM PREDICADO EMBUTIDO
 *****************************************************************/

   PROCEDURE CRIEGPOEMBUT (NOMEPRED   TTIPOPRED)

     VAR PRED   TAPELEMTABSIMB

     BEGIN NEW (PREDSEMBUT (.NOMEPRED.) - . GPOCLS)
           PRED  = PREDSEMBUT(.NOMEPRED.)
           PRED - . GPOCLS - . TIPOPREDCABEC  = NOMEPRED
           NEW (PRED - . GPOCLS - . PRIMCLAUS)
           NEW (PRED - . GPOCLS - . PRIMCLAUS - . CABECA)
           PRED - . GPOCLS - . PRIMCLAUS - . CABECA - . NOME  = PRED
           PRED - . GPOCLS - . PRIMCLAUS - . PROXCLAUS  = NIL
     END
```

```
/**************************************************************************
 * CRIEEMBUTIDOS - PROCEDURE QUE INSERE UM PREDICADO EMBUTIDO NA TABELA
 *                 DE SIMBOLOS
 **************************************************************************/

   PROCEDURE INSIRANATABSIMB (VAR PRED   TAPELEMTABSIMB)

     VAR COLISOR   TAPELEMTABSIMB
         INDHASH   TINDTABHASH

    BEGIN INDHASH  = CALCULEINDHASH (PRED - . COMPRNOME, PRED - . NOME)
          COLISOR  = TABHASH (.INDHASH.)
          IF COLISOR = NIL
            THEN TABHASH (.INDHASH.)  = PRED
            ELSE BEGIN WHILE COLISOR - . APNOMECOLISOR )  NIL
                         DO COLISOR  = COLISOR - . APNOMECOLISOR
                       COLISOR - . APNOMECOLISOR  = PRED
                 END
    END


/**************************************************************************
 * CRIEEMBUTIDOS - PROCEDURE QUE CRIA O PREDICADO   (CORTE).
 **************************************************************************/

   PROCEDURE CRIEPREDCORTE

     BEGIN NEW (PREDSEMBUT (.PREDCORTE.))
           PREDSEMBUT (.PREDCORTE.) - . COMPRNOME  = 1
           PREDSEMBUT (.PREDCORTE.) - . NOME(.1.)  = ' '
           CRIEGPOEMBUT (PREDCORTE)
           INSIRANATABSIMB (PREDSEMBUT (.PREDCORTE.))
     END


/**************************************************************************
 * CRIEEMBUTIDOS   -    PROCEDURE QUE CRIA O PREDICADO WRITE.
 **************************************************************************/

   PROCEDURE CRIEPREDWRITE

     BEGIN NEW (PREDSEMBUT (.PREDWRITE.))
           WITH PREDSEMBUT (.PREDWRITE.) -
            DO BEGIN COMPRNOME  = 4
                     NOME(.1.)  = 'I'
                     NOME(.2.)  = 'M'
                     NOME(.3.)  = 'P'
                     NOME(.4.)  = 'R'
                END
           CRIEGPOEMBUT (PREDWRITE)
           INSIRANATABSIMB (PREDSEMBUT (.PREDWRITE.))
     END
```

```
/**********************************************************************
 * CRIEEMBUTIDOS - PROCEDURE QUE CRIA O PREDICADO  WRITEFALHA .
 **********************************************************************/

   PROCEDURE CRIEPREDWRTFALHA

     BEGIN NEW (PREDSEMBUT(.PREDWRITEFALHA.))
           WITH PREDSEMBUT(.PREDWRITEFALHA.) -
            DO BEGIN COMPRNOME   = 9
                     NOME(.1.)   = 'I'
                     NOME(.2.)   = 'M'
                     NOME(.3.)   = 'P'
                     NOME(.4.)   = 'R'
                     NOME(.5.)   = 'F'
                     NOME(.6.)   = 'A'
                     NOME(.7.)   = 'L'
                     NOME(.8.)   = 'H'
                     NOME(.9.)   = 'A'
               END
           CRIEGPOEMBUT(PREDWRITEFALHA)
           INSIRANATABSIMB(PREDSEMBUT(.PREDWRITEFALHA.))
     END


/**********************************************************************
 * CRIEEMBUTIDOS - PROCEDURE QUE CRIA A FUN  O SOMA (+).
 **********************************************************************/

   PROCEDURE CRIEFUNCSOMA

     BEGIN NEW (FUNCSEMBUT(.FUNCSOMA.))
           FUNCSEMBUT(.FUNCSOMA.) - . COMPRNOME   = 1
           FUNCSEMBUT(.FUNCSOMA.) - . NOME(.1.)   = '+'
           INSIRANATABSIMB (FUNCSEMBUT(.FUNCSOMA.))
     END


   PROCEDURE CRIEFUNCSUBT

     BEGIN NEW (FUNCSEMBUT(.FUNCSUBT.))
           FUNCSEMBUT (.FUNCSUBT.) - . COMPRNOME   = 1
           FUNCSEMBUT (.FUNCSUBT.) - . NOME(.1.)   = '-'
           INSIRANATABSIMB (FUNCSEMBUT(.FUNCSUBT.))
     END


   PROCEDURE CRIEFUNCMULT

     BEGIN NEW (FUNCSEMBUT(.FUNCMULT.))
           FUNCSEMBUT (.FUNCMULT.) - . COMPRNOME   = 1
           FUNCSEMBUT (.FUNCMULT.) - . NOME(.1.)   = '*'
           INSIRANATABSIMB (FUNCSEMBUT(.FUNCMULT.))
     END
```

```
  PROCEDURE CRIEFUNCDIVINT

    BEGIN NEW (FUNCSEMBUT(.FUNCDIVINT.))
          FUNCSEMBUT(.FUNCDIVINT.) - . COMPRNOME  = 1
          FUNCSEMBUT(.FUNCDIVINT.) - . NOME(.1.)  = ' '
          INSIRANATABSIMB (FUNCSEMBUT(.FUNCDIVINT.))
    END
  PROCEDURE CRIEFUNCDIVREAL

    BEGIN NEW (FUNCSEMBUT(.FUNCDIVREAL.))
          FUNCSEMBUT(.FUNCDIVREAL.) - . COMPRNOME  = 1
          FUNCSEMBUT(.FUNCDIVREAL.) - . NOME(.1.)  = '/'
          INSIRANATABSIMB (FUNCSEMBUT(.FUNCDIVREAL.))
    END


  PROCEDURE CRIEFUNCMOD

    BEGIN NEW (FUNCSEMBUT(.FUNCMOD.))
          WITH FUNCSEMBUT(.FUNCMOD.) -
           DO BEGIN COMPRNOME  = 3
                    NOME(.1.)  = 'M'
                    NOME(.2.)  = 'O'
                    NOME(.3.)  = 'D'
              END
          INSIRANATABSIMB (FUNCSEMBUT(.FUNCMOD.))
    END

/*****************************************************************
 * CRIEEMBUTIDOS - PROCEDURE QUE CRIA A FUNCAO SINAL.
 *****************************************************************/

  PROCEDURE CRIEFUNCSINAL

    BEGIN NEW (FUNCSEMBUT(.FUNCSINAL.))
          WITH FUNCSEMBUT(.FUNCSINAL.) -
           DO BEGIN COMPRNOME  = 5
                    NOME(.1.)  = 'S'
                    NOME(.2.)  = 'I'
                    NOME(.3.)  = 'N'
                    NOME(.4.)  = 'A'
                    NOME(.5.)  = 'L'
              END
          INSIRANATABSIMB (FUNCSEMBUT(.FUNCSINAL.))
    END

/*****************************************************************
 * CRIEEMBUTIDOS - PROCEDURE QUE CRIA A FUN  O CALCULADORA ( )
 *****************************************************************/

  PROCEDURE CRIEFUNCCALC

    BEGIN NEW (FUNCSEMBUT(.FUNCCALC.))
```

UNIVERSIDADE FEDERAL DA PARAÍBA
Pró-Reitoria Para Assuntos do Interior
Coordenação Setorial de Pós-Graduação
Rua Aprigio Veloso, 882 - Tel (083) 321-7222-R 355
58.100 - Campina Grande - Paraíba

```
        FUNCSEMBUT(.FUNCCALC.) - . COMPRNOME   = 1
        FUNCSEMBUT(.FUNCCALC.) - . NOME(.1.)   = ' '
        INSIRANATABSIMB (FUNCSEMBUT(.FUNCCALC.))
    END


/*****************************************************************
 * CRIEEMBUTIDOS   -   CORPO DA PROCEDURE
 *****************************************************************/

    BEGIN CRIEPREDWRITE
          CRIEPREDCORTE
          CRIEPREDWRTFALHA
          CRIEFUNCCALC
          CRIEFUNCSOMA
          CRIEFUNCSUBT
          CRIEFUNCMULT
          CRIEFUNCDIVINT
          CRIEFUNCDIVREAL
          CRIEFUNCMOD
          CRIEFUNCSINAL
    END


/*****************************************************************
 * PROGRAMA PRINCIPAL - CORPO DO PROGRAMA
 *****************************************************************/

  BEGIN FOR INDTABHASH  = 0 TO INDMAXTABHASH
          DO TABHASH (.INDTABHASH.)  = NIL
        CRIEEMBUTIDOS
        OBTENHACLAUS (CLAUS)
        WHILE CLAUS ) NIL
          DO BEGIN IF CLAUS - . CABECA )  NIL
                     THEN ARMAZENECLAUSDEF (CLAUS)
                     ELSE RESOLVACLAUSMETA (CLAUS)
                   OBTENHACLAUS (CLAUS)
             END
  END
```

APÊNDICE IV

PROCEDURES EXEC USADAS PARA ATIVAR O INTERPRETADOR

Uma procedure EXEC é uma sequência de comandos executáveis pelo sistema VM/CMS armazenados num arquivo do tipo EXEC. Cada vez que o nome de um arquivo do tipo exec é invocado, os comandos nele contidos são executados.

O arquivo PROLOG EXEC contendo o grupo de comandos que facilitam o uso do interpretador é dado a seguir.

```
&CONTROL OFF
PASCMOD PROLOG
&TYPE
&TYPE ******** ******** ******** ******** ******** ******** ******** ****
&TYPE ******** ******** ******** ******** ******** ******** ******** ****
&TYPE
&TYPE INTERP. PROLOG BASICO -------- -------- --------  UFPB / CCT / DSC
&TYPE
&TYPE DESENV. COMO TESE DE MESTRADO POR ERALDO CRUZ LACET - JUNHO 1985
&TYPE
&TYPE ******** ******** ******** ******** ******** ******** ******** ****
&TYPE ******** ******** ******** ******** ******** ******** ******** ****
&TYPE
&TYPE DIGITE OS NOMES DOS ARQUIVOS DE ENTRADA
&READ ARGS
&IF &INDEX EQ 0 &GOTO -CLSTERM
COPYFILE PREDEF PROLOG A1 ENTRADA PROLOG A1
&J = 1
-LACO STATE &&J PROLOG A1
   &IF &RETCODE   0 &GOTO -ARQINDEF
   COPYFILE &&J PROLOG A1 ENTRADA PROLOG A1 (AP)
   &J = 1 + &J
   &IF &J )= &INDEX &GOTO -LACO
FILEDEF INPUT DISK ENTRADA PROLOG A1
&SKIP 3
-CLSTERM FILEDEF INPUT TERM (RECFM V LRECL 80 LOWCASE
&TIPO = 0
&SKIP 1
&TIPO = 1
&TYPE DIGITE O NOME DO ARQUIVO DE SAIDA
&READ ARGS
&IF &INDEX   0 &SKIP 2
   FILEDEF OUTPUT TERM
   &SKIP 1
FILEDEF OUTPUT DISK &1 PROLOG A1
&IF &TIPO = 0 &TYPE DIGITE  CLAUS. DO PROGRAMA
PROLOG
&SKIP 1
-ARQINDEF &TYPE NAO EXISTE O ARQUIVO &&J PROLOG A1
&IF &TIPO = 1 ERASE ENTRADA PROLOG A1
```

APÊNDICE V

UTILIZAÇÃO DO INTERPRETADOR

a) Como usar o interpretador

O interpretador funciona sob o sistema VM/CMS. Foi definido um procedimento EXEC (ver apêndice IV) que permite sua ativação. Para ativar esta EXEC é necessário apenas teclar PROLOG no ambiente CMS. Feito isto aparece no terminal a mensagem:

DIGITE OS NOMES DOS ARQUIVOS DE ENTRADA

o usuário deve então introduzir os nomes dos arquivos que deseja analisar. Caso tecle o ENTER sem ter introduzido o nome de qualquer arquivo, o sistema entenderá que o arquivo de entrada será o console do terminal. Se o usuário desejar, pode introduzir juntamente com seus arquivos de entrada, o arquivo PREDEF, citado no capítulo 3. Ficando assim disponível ao seu programa os predicados ali definidos.

Definidos os arquivos de entrada, surge a mensagem:

DIGITE O NOME DO ARQUIVO DE SAIDA

o usuário deve introduzir o nome do arquivo no qual deseja que seja armazenada a saída de seu programa ou, se desejá-la no vídeo de seu terminal, simplesmente tecla ENTER. Neste

ponto, se o arquivo de entrada é a console do terminal, surge a mensagem:

```
DIGITE CLAUS DO PROGRAMA
```

O interpretador é ativado e passa a analisar as cláusulas dos arquivos ou as introduzidas via terminal.

b) EXEMPLOS DE PROGRAMAS

Programa 1: Torre de Hanoi:

```
Hanoi(n) <- Move(n,Direito,Central,Esquerdo).
Move(0,_,_,_) <- !.
Move(n,a,b,c) <- Move(?(-(n,1)),a,c,b) & Informa(a,b)
                 & Move(?(-(n,1)),c,b,a).
Informa(x,y) <- Write("\Mova um disco do pino ",x,
                                    " para o pino ",y).
? Hanoi(5).
```

Resultado obtido:

```
Mova um disco do pino Direito para o pino Central
Mova um disco do pino Direito para o pino Esquerdo
Mova um disco do pino Central para o pino Esquerdo
Mova um disco do pino Direito para o pino Central
Mova um disco do pino Esquerdo para o pino Direito
Mova um disco do pino Esquerdo para o pino Central
Mova um disco do pino Direito para o pino Central
Mova um disco do pino Direito para o pino Esquerdo
Mova um disco do pino Central para o pino Esquerdo
Mova um disco do pino Central para o pino Esquerdo
Mova um disco do pino Esquerdo para o pino Direito
Mova um disco do pino Central para o pino Esquerdo
Mova um disco do pino Direito para o pino Central
Mova um disco do pino Direito para o pino Esquerdo
Mova um disco do pino Central para o pino Esquerdo
Mova um disco do pino Direito para o pino Central
```

```
Mova um disco do pino Esquerdo para o pino Direito
Mova um disco do pino Esquerdo para o pino Central
Mova um disco do pino Direito para o pino Central
Mova um disco do pino Esquerdo para o pino Direito
Mova um disco do pino Central para o pino Esquerdo
Mova um disco do pino Central para o pino Direito
Mova um disco do pino Esquerdo para o pino Direito
Mova um disco do pino Esquerdo para o pino Central
Mova um disco do pino Direito para o pino Central
Mova um disco do pino Direito para o pino Esquerdo
Mova um disco do pino Central para o pino Esquerdo
Mova um disco do pino Direito para o pino Central
Mova um disco do pino Esquerdo para o pino Direito
Mova um disco do pino Esquerdo para o pino Central
Mova um disco do pino Direito para o pino Central
```

Programa 2: Fatorial.

```
Fat(0,1).
Fat(x,?(*(x,z))) <- Fat(?(-(x,1)),z).
Fatorial(x) <- Fat(x,y) & Write("\Fatorial de ",x," = ",y).
? Fatorial(4).
? Fatorial(10).
? Fatorial(16).
```

Resultado obtido:

```
Fatorial de          4 =          24
Fatorial de         10 =     3628800
Fatorial de         16 =  2004189184
```

c) Instalação do Interpretador:

Estando o código fonte armazenado no arquivo "PROLOG PASCAL A1", compila-se no ambiente "CMS" através do comando:

```
PASCALVS PROLOG
```

A seguir, estando a "procedure exec" mostrada no apêndice IV, armazenada no arquivo "PROLOG EXEC A1", bastará usá-lo, conforme descrito no ítem (a).

APÊNDICE VI

ALGUMAS CONSIDERAÇÕES ACERCA DA EFICIÊNCIA

A eficiência não foi incluida dentre os objetivos deste trabalho. Assim, buscou-se muito mais a clareza e modularidade do código fonte que a eficiência. Além disto, não se dispõe de um outro interpretador PROLOG que funcione na mesma máquina, nem sobre o mesmo sistema operacional nem implementado na mesma linguagem que o deste trabalho. Por este motivo seria difícil obter uma medida comparativa precisa da eficiência deste interpretador em relação a outros interpretadores. Mesmo assim será mostrado um estudo comparativo, embora impreciso e grosseiro, entre este interpretador e uma cópia do interpretador desenvolvido por Ferguson (1981) na linguagem C que funciona no pdp-11/34/UNIX da UFPb.

Para se dar uma idéia da relação entre os tempos de CPU destes dois interpretadores, tomou-se o programa do caminho no grafo para o grafo definido através das seguintes cláusulas:

```
ExisteArco(A,B).
ExisteArco(B,C).
ExisteArco(A,C).
ExisteArco(C,D).
ExisteArco(D,E).
ExisteArco(C,E).
ExisteArco(E,F).
ExisteArco(F,G).
ExisteArco(E,G).
ExisteArco(G,H).
ExisteArco(H,I).
ExisteArco(G,I).
ExisteArco(I,J).
ExisteArco(J,L).
ExisteArco(I,L).
ExisteArco(L,M).
ExisteArco(M,N).
ExisteArco(L,N).
ExisteArco(N,O).
```

```
ExisteArco(O,P).
ExisteArco(N,P).
ExisteArco(P,Q).
ExisteArco(Q,R).
ExisteArco(P,R).
ExisteArco(R,S).
ExisteArco(S,T).
ExisteArco(R,T).
ExisteArco(T,U).
ExisteArco(U,V).
ExisteArco(T,V).
ExisteArco(A,X).
ExisteArco(X,Z).
?Caminho(A,Z,qual).
```

Propositadamente as cláusulas que fornecem o caminho procurado foram posicionadas ao final dos programas. Assim antes de encontrar o caminho de A a Z são tentados todos os outros caminhos do grafo.

Tomando como medidas equivalentes de tempos de CPU, aquelas definidas no VM/CMS do IBM como "Virtual time" e no PDP-11/UNIX como "user", o que, a rigor, não poderia ser afirmado, temos os seguintes tempos de CPU para cada uma das implementações:

PROLOG DESENVOLVIDO NESTE TRABALHO:
"Virtual time" = 409,4 seg

PROLOG DE FERGUSON:
"User" = 181,4 seg

O programa C:

```
#include <stdio.h>
main ()
{
     register int i, j, k;

     for (i = 1; i < 500; i++)
          for (j = 1; j < 100; j++)
               for (k = 1 ; k < 100; k++)
                    f (i, j, k);
```

```
}

f (i, j, k)
int i, j, k;
{}
```

E o programa Pascal:

```
program laco

 ;procedure f (i, j, k : integer)

   ;begin
   ;end

 ;var i, j, k : integer

 ;begin for i := 1 to 500
         do for j := 1 to 100
             do for k := 1 to 100
                 do f (i, j, k)
 ;end
```

foram usados para obter um fator que, embora de maneira grosseira corrija as discrepâncias ocasionadas pelo fato dos tempos referirem-se a interpretadores implementados em linguagens diferentes, em máquinas diferentes com sistemas operacionas diferentes.

Estes programas forneceram os seguintes tempos:

PROGRAMA C NO PDP/11/34/UNIX:
"User" = 360,2 seg

PROGRAMA PASCAL NO IBM 4341/VM/CMS:
"Virtual time" = 96,0 seg

Tomando-se como fator de correção a razão "User/Virtual_time" destes dois programas, que é aproximadamente 3,7, tem-se o interpretador desenvolvido

neste trabalho 8,5 vêzes mais lento que o de Ferguson.