

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Universidade do Porto
Faculdade de Engenharia

FEUP

EXERCÍCIOS DE PROGRAMAÇÃO EM LÓGICA

LUÍS PAULO REIS
DANIEL CASTRO SILVA

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA E
COMPUTAÇÃO

PROGRAMAÇÃO EM LÓGICA - 3º ANO
SETEMBRO DE 2007



Faculdade de Engenharia da Universidade do Porto
Licenciatura em Engenharia Informática e Computação
Programação em Lógica

2003/2004
LEIC
(3º Ano)
1º Sem

Exercícios – Meta-Programação e Meta-Interpretores

Exercício MP1. Utilização do Operador =..

É frequente desejarmos realizar uma dada transformação em todos os elementos de uma lista. Para o efeito vamos recorrer a um predicado de aridade 2. A esta transformação chama-se também mapeamento numa lista. Construa um predicado de mapeamento utilizando o operador =.. na sua definição.

Exemplo1:

Tendo

```
f(X,Y) :- Y is X*X.
```

vem

```
?-map([2,4,8],f,L).
```

```
L=[4,16,64]
```

Exemplo2:

Tendo

```
duplica(X,Y) :- Y is 2*X.
```

vem

```
?-map([1,2,3],duplica,L).
```

```
L=[2,4,6]
```

Solução:

```
map([],_,[]).
```

```
map([C|R],Transfor,[TC|CR]) :-  
    aplica(Transfor, [C,TC]),  
    map(R,Transfor,CR).
```

```
aplica(P,LArgs) :- G =.. [P|LArgs], G.
```

Exercício MP2. Lista de Elementos que tornam Predicado Verdadeiro

Implemente o predicado separa(+L,+Pred,-Lista) que dada uma lista L e um nome de um predicado de aridade 1, devolve a lista com exactamente os mesmos elementos mas em que primeiro aparecem todos aqueles que tornam verdadeiro o predicado.

```
separa(L,P,Res) :- sepDL(L,P,Res-Nots,Nots-[]).
```

```
sepDL([],_,P-P,N-N).
```

```
sepDL([V|L],P,[V|Y]-DY,N) :- aplica(P,[V]), !, sepDL(L,P,Y-DY,N).
sepDL([V|L],P,Y,[V|N]-DN) :- sepDL(L,P,Y,N-DN).
```

Exercício MP3. Idades Mais Próximas

Implemente utilizando o setof/3, o predicado mais_proximos(+Idade,-ListaProximos) que, assumindo a existência de factos idade(Nome,Idade) para representar que um dado indivíduo chamado Nome tem idade Idade, devolve em ListaProximos o nome dos indivíduos cuja idade é mais próxima de Idade.

Solução:

```
mais_proximos(I,[N1|Prox]) :-
    setof(Dif-Nome,prox(I,Dif,Nome),[D1-N1|L]),
    primeiros(D1,L,Prox).

prox(I,Dif,Nome) :- idade(Nome,Id), dif(I,Id,Dif).

dif(A,B,D) :- A > B, !, D is A - B.
dif(A,B,D) :- D is B - A.

primeiros(_,[],[]).
primeiros(D1,[D-_|_],[]) :- D > D1, !.
primeiros(D1,[_N|L],[N|NL]) :- primeiros(D1,L,NL).

%Dados para teste:
idade(maria,30).
idade(pedro,25).
idade(jose,25).
idade(rita,18).
```

Exercício MP4. Definição de functor(Term,F,N) e arg(N,Term,Arg) em termos do operador =..

a) Defina o predicado functor2(Term,F,Arity) que é verdadeiro se Term é um termo cujo functor principal tem o nome F e a aridade Arity.

Solução:


```
functor_(Term,F,N) :- Term=..[F|Args], length(Args,N).
```

b) Defina o predicado arg(N,Term,Arg) que é verdadeiro se Arg é o N-ésimo argumento do termo Term.

Solução:

```
arg_(N,Term,Arg) :- Term=..[F|Args], position(N,Args,Arg).

position(1,[X|_],X).
position(N,[_|Xs],Y) :- N>1, N1 is N-1, position(N1,Xs,Y).
```

 Universidade do Porto Faculdade de Engenharia FEUP	Faculdade de Engenharia da Universidade do Porto Licenciatura em Engenharia Informática e Computação Programação em Lógica	2003/2004 LEIC (3º Ano) 1º Sem
Docentes: Luís Paulo Reis e Eugénio da Costa Oliveira		
Exercícios OPA – Operadores e Aritmética		

Exercício OPA 1. Utilização de Operadores

Suponha que temos definidos os seguintes operadores:

```
:- op(500,xfx,na) .
:- op(500,xfy,ad) .
:- op(500,yfx,ae) .
```

Mostre como seriam representadas em PROLOG as seguintes expressões se não tivéssemos as directivas acima (indique os casos em que o PROLOG assinalaria um erro sintáctico):

- a na b ae c.
- a na b ad c.
- a ad b na c.
- a na b na c.
- a ad b ad c.
- a ae b ae c.
- a ad b ad c na d ae e ae f.

Solução:

- ae(na(a,b),c) .
- Erro.
- ad(a,na(b,c)) .
- Erro.
- ad(a,ad(b,c)) .
- ae(ae(a,b),c) .
- ad(a,ad(b,ae(ae(na(c,d),e),f))) .

Exercício OPA 2. Definição de Operadores Diversos

Crie as directivas que tornam termos abaixo sintacticamente válidos:

- se X entao Y senao Z.
- Y gostaria_de X se X fosse bom e X fosse inteligente .

Solução:

- :-op(500, xfx, entao) .
- :-op(400, fx, se) .
- :-op(400, xfx, senao) .

```

b) :-op(800, xfx, se).
   :-op(600, xfx, gostaria_de).
   :-op(500, xfy, e).
   :-op(400, xfx, fosse).

```

Exercício OPA 3. Definição de Operadores para Voos

Suponha que temos definidos os seguintes operadores:

```

:-op (700, xfx, \\\).
:-op (600, xfx, //).
:-op (600, xfy, ':' :').
:-op (400, yfx, para).
:-op (400, xfx, de).

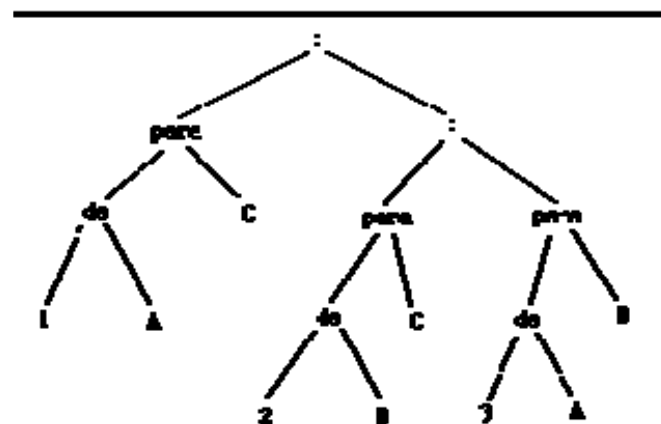
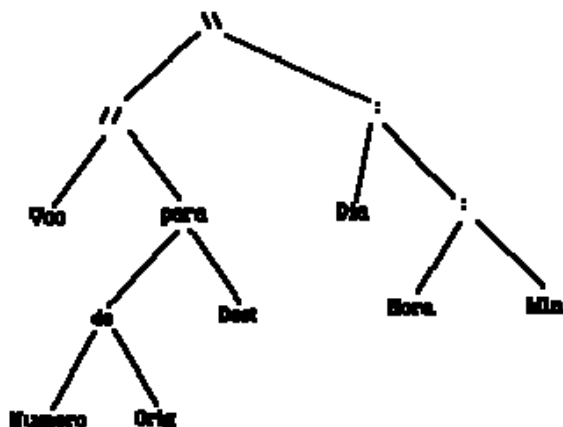
```

Construa uma representação gráfica para os termos:

a) Voo // Número de Orig para Dest \\\ Dia: Hora: min.

b) 1 de A para C: 2 de B para C: 3 de A para B.

Solução:



a) _____ b) _____

Exercício OPA 4. Definição de Operadores para Operações com Listas

Algumas das relações que envolvem listas foram anteriormente escritas no seguinte formato:

```

member(Elemento,Lista),
concatena(Lista1,Lista2,Lista),
delete(Elemento,Lista,NovaLista),
...

```

Suponha que preferíamos escrever estas relações no seguinte formato:

```

Elemento existe_em Lista ,
concatena Lista1 e Lista2 da Lista

```

apaga Elemento a Lista da NovaLista.

Declare existe_em, concatenar, e, etc. como operadores de modo a tornar este formato possível. Redefina as correspondentes relações de acordo com as alterações realizadas.

Solução:

```
:- op(200,xfx,existe_em).
X existe_em [X|_].
X existe_em [_|L]:-
X existe_em L.
:- op(200,fx,concatena).
:- op(150,xfx,da).
:- op(100,xfx,e).
concatena [] e L da L.
concatena [X|L1] e L2 da [X|L3] :-
concatena L1 e L2 da L3.
:- op(200,fx,apaga).
:- op(100,xfx,a).
apaga X a [X|L] da L.
apaga X a [Y|L] da [Y|L1] :-
apaga X a L da L1.
```

Exercício OPA 5. Definição de Operadores – Joga e E

Assumindo as seguintes definições de operadores:

```
:- op(300,xfx,joga).
:- op(200,xfy,e).
```

então os dois termos seguintes possuem sintaxe válida:

```
T1 = marcelo joga futebol e squash.
T2 = renata joga tenis e basquete e volei.
```

Como estes termos são interpretados pelo Prolog? Qual é o functor principal de cada termo e qual a sua estrutura?

Exercício OPA 6. Definição de Operadores – Era e Do

Sugira uma apropriada definição dos operadores "era" e "do" para que seja possível a escrita de cláusulas como:

```
vera era secretária do departamento.
e
paulo era professor do curso.
```

Exercício OPA 7. Definição de Operadores – Operador +

Considere o seguinte programa Prolog:

```
t(0+1, 1+0).
t(X+0+1, X+1+0).
```

```
t(X+1+1, Z) :-
    t(X+1, X1),
    t(X1+1, Z).
```

Como irá este programa responder as seguintes questões, considerando ser + um operador infixo do tipo yfx (como usual).

- a) ?-t(0+1, A).
- b) ?-t(0+1+1, B).
- c) ?-t(1+0+1+1+1, C).
- d) ?-t(D, 1+1+1+0).

Exercício OPA 8. Definição de Operadores – Se, Então e Senão

Defina os operadores "se", "então", "senão" e "==" de modo que seja válido o termo:

```
se X>Y então Z := X senão Z := Y
```

Escolha a precedência dos operadores de modo que "se" venha a ser o functor principal. Depois defina a relação "se" como um mini-interpretador para um tipo de comando se-então da forma:

```
se V1>V2 então Var:=V3 senão Var:=V4
```

onde V1, V2, V3 e V4 são números (ou variáveis instanciadas com números) e Var é uma variável. O significado da relação "se" deve ser: "se o valor de V1 é maior que o valor de V2, então Var é instanciada com V3, senão Var é instanciada com V4. Um exemplo do uso do mini-interpretador seria:

```
?-X=2, Y=3, V2 is 2*X, V4 is 4*X,
    se Y > V2 então Z:=Y senão Z:=V4,
    se Z > 5 então W=1 senão W=0.
X=2 Y=3 Z=8 W=1
```

Exercício OPA 9. Definição de Operadores – Entre

Defina o procedimento

```
entre(N1, N2, X)
```

que, para dois inteiros dados, N1 e N2, produz através de backtracking todos os inteiros X que satisfazem a restrição

```
N1 >= X >= N2
```

Exercício OPA 10. Definição de Operadores – Polígonos

Estude a definição de um "mundo de polígonos" onde os objectos são definidos em função das coordenadas de seus vértices no plano. Indivíduos desse universo seriam triângulos, rectângulos, quadrados, etc. Por exemplo o termo:

```
triângulo((1,1), (1,2), (2,2))
```

definiria um triângulo cujos vértices seriam os pontos (1,1), (1,2) e (2, 2) em um sistema de coordenadas cartesianas.

Formule as propriedades básicas de cada objecto através de relações unárias, tais como:

`isósceles(X)`

Formule relações entre diferentes indivíduos, representando assertivas tais como:

`"Uma casa é um quadrado com um triângulo em cima".`

ou

`"D é distância entre os centros geométricos de A e B".`

Pense numa versão deste programa para gerar trajectórias de figuras planas ao longo de curvas de equações dadas.