Conteúdo

1 Representação de dados estruturados **1**

2 Listas **2**

3 Matrizes e outros **4**

1 Representação de dados estruturados

Representação de dados estruturados

Variáveis : X Pais Nome

Constantes :a antonio 78

Termos complexos :

f(a,X) nome(angela)
pais(nome(portugal),lingua(portugues),continente(europa))
pais(portugal,portugues,europa)
s(np(det(o),nome(gato)),vp(vt(mordeu),np(det(a),pn(joana))))
0 s(0) s(s(0)) s(s(s(0))) ...
arv(a,arv(b,vazia,vazia),arv(c,vazia,vazia))

Exemplos

Inteiros como termos:

```
inteiro(0).
inteiro(s(N)):- inteiro(N).
```

Exemplo: adição de inteiros em lógica

```
soma(0,Y,Y).
soma(s(X),Y,s(Z)):- soma(X,Y,Z).
```

Exemplo: verificar se um termo é uma arvore binária

```
arvore(vazia).
arvore(arv(_,Esq,Dir)) :- arvore(Esq),arvore(Dir).
```

Predicados não-lógicos

Permitem:

- Entrada/Saída: ler e escrever (`write`, `get`, etc)
- Numéricos: para obter um aritmética mais eficiente
- etc...

Como tem efeitos secundários interfere com a resolução e o retrocesso!!!

is

`X is Expression`

`X is Y+1` se Y é um número

Não permite unificação...

Aritmética (predicados não lógicos)

`X is <expressao aritmetica>`

`X is 2+2*2`

Comparações aritméticas

`X =\= Y` `X` avalia para um valor diferente de `Y`

`X =:= Y` `X` avalia para um valor igual a `Y`

`X > Y` `X` avalia para um valor maior do que `Y`

`X < Y` `X` avalia para um valor menor do que `Y`

`X =< Y` `X` avalia para um valor menor ou igual do que `Y`

2 Listas

Listas

- [ ]
- $.(x, y)$ (ou $[x|y]$)
- $[x_1, \ldots, x_n|[]]$ ou $[x_1, \ldots, x_n]$

Listas

É possível unificar as listas seguintes?

$[X, Y, Z]$	$[john, likes, fish]$
$[cat]$	$[X \mid Y]$
$[X, Y \mid Z]$	$[mary, likes, wine]$
$[[the, Y] \mid Z]$	$[[X, answer], [is, here]]$
$[X, Y, X]$	$[a, Z, Z]$
$[[X], [Y], [X]]$	$[[a], [Z] \mid Z]$
$.(.(a, .(b, [])), .(c, []))$	$[X \mid Y]$

Listas

Exercícios:

Defina os predicados seguintes:

- `member/2`
- `list/1`
- `length/2`
- `prod_esc/3`

Volte a definir os dois últimos predicados utilizando uma variável para acumular resultados intermédios.

Supondo listas de inteiros:

- `max/2`
- `sumlist/2`

Listas

Mais Exercícios:

- `append/3`
- `naive_reverse/2`
- `reverse/2`
- `delete/3`

Definir com `append`:

- `prefix/2`
- `suffix/2`
- `member/2`
- `last/2`

Caminhos em digrafos - com ciclos

```
a(g,h).
a(g,d).
a(d,a).
a(e,d).
a(h,f).
a(h,f).
a(a,e).
a(e,f).
a(a,b).
a(b,f).
```

- `pathCic/2`

Aplicações de listas

Transformar uma lista noutra com uma dada propriedade

Exercícios:

Defina os predicados seguintes:

- `square/2`
- O mesmo do anterior com `transform/2`
- `pares/2`
- `sem_dup/2`

3 Matrizes e outros

Exercícios:

Defina os predicados seguintes:

- `transposta/2` (Ex: `transp([[1,2,3],[4,5,6]],[[1,4],[2,5],[3,6]])`)
- `occ_rep/4` (Ex: `occ_rep([1,1,2,a,a,a,3],C,0,[2*1,1*2,3*a,1*3])`)
- `sum_cum/2` (Ex: `sum_cum([1,3,2,5,4],[1,4,6,11,15])`)
- `split_int/3` (Ex: `split_int([1,3,2,5,4],[1,3,5],[2,4])`)