Lógica em Programação

Lista de Exercícios Selecionados

1. Escreva programas em Prolog para calcular:

i) o fatorial de um número 'n'.

ii) a função de Ackermann a(m,n) definida por:

$$
\begin{aligned}
a(0,n) &= n + 1 \\
a(m,0) &= a(m-1,1) \\
a(m,n) &= a(m-1,a(m,n-1)) \text{ se } m > 0,\ n > 0.
\end{aligned}
$$

(Nota: Teste esta função para valores de 'm' menores do que 4...)

2. Escreva uma versão mais eficiente do programa Ackermann, utilizando 'assert'.

3. O arquivo 'cricket' contém informação sobre partidas de cricket entre Inglaterra e Austrália:

```
century(Name,Year,Score,Ground).
australia(Ground).
england(Ground).
australian(Name).
english(Name).
```

Escreva programas em Prolog para:

(i) imprimir uma lista de todos os batedores que fizeram um 'double century' (200 corridas ou mais) nos anos 20.

(ii) encontrar todos os batedores que fizeram 'centuries' em três ou mais 'grounds' diferentes, e listar a informação completa em cada caso (nome, ano, número de 'centuries' e 'ground').

(iii) encontrar qual/quais batedor/es fizeram o maior número de 'centuries' em qualquer 'ground'.

4. Escreva um programa Prolog que descreva o circuito da figura 0.1. Utilize este programa para encontrar as saídas D,E para todas as possíveis combinações de entradas A, B e C.

Repita a computação para o caso em que o dispositivo marcado está com defeito e sempre produz saída '0' para qualquer combinação de entrada.

5. Usando o arquivo 'england', escreva programas para:

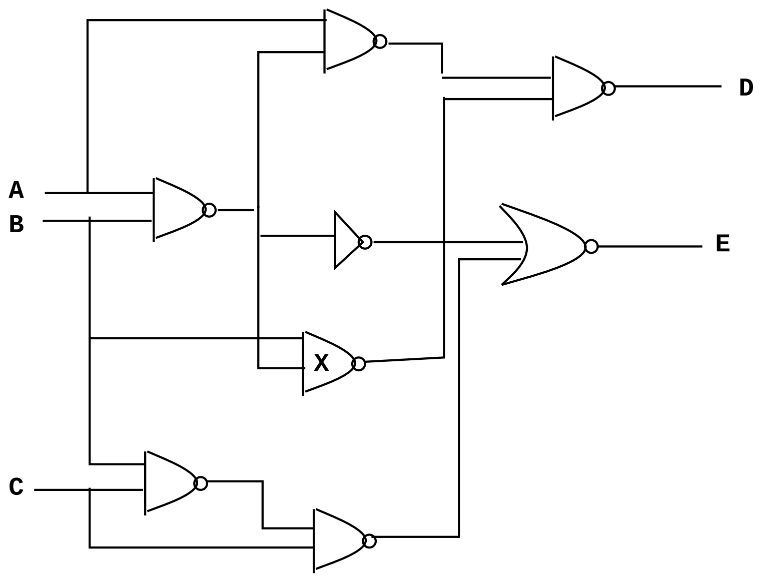

Figure 0.1: Figura para exercício 2

(i) imprimir uma lista de todos os condados que são adjacentes a três ou mais condados.

(ii) imprimir uma lista de todas as cidades que estão a menos de 65 milhas de Bristol, listando-as por condado.

(iii) imprimir uma lista de condados que você atravessaria para ir de Kent a Cornwall.

6. O arquivo 'tennis' contém informação sobre alguns jogadores de tênis.

champion(Name).
male(Name).
female(Name).
child(Child,Parent).
older(Name1,Name2). Name1 é mais velho que Name2.
born_before(Name1,Name2). Name1 nasceu antes de Name2.

Escreva programas em Prolog que responda as seguintes questões:

(i) Qual tenista mulher é mais velha que Kelly?

(ii) Quais campeões masculinos são mais jovens que Kelly?

(iii) Borg é mais velho que Evert?

[Dica: Talvez fique mais fácil definir um novo predicado mais_velho_que(X,Y).]

7. Faça uma **breve** comparação entre linguagens declarativas (p.e., Prolog) e linguagens imperativas (p.e., C), destacando os pontos mais importantes.

8. Escreva de forma concisa o algoritmo de execução de Prolog.

9. Considere o seguinte programa:

```
top(X,Y) :- p(X,Y).
top(X,X) :- s(X).
p(X,Y) :- q(X), r(Y).
p(X,Y) :- s(X), r(Y).
```

```
q(a)
q(b).
r(c).
r(d).
s(e).
```

Faça um esboço da árvore de execução para a consulta ?- top(X,Y) com as respectivas soluções para cada ramo da árvore.

10. Considere o seguinte programa:

```
p(Y) :- q(X,Y), r(Y).
p(X) :- q(X,X).
q(a,a).
q(a,b).
r(b).
```

Introduza o operador de corte (cut, !) em lugares diferentes dete programa e diga quantas e quais respostas você obteria para a consulta `p(Z)`.

11. Qual a função do programa abaixo?

```
c(_, [], 0).
c(X, [X|L], N):-
        c(X, L, R), !,
        N is R+1.
c(X, [_|L], N):-
        c(X, L, N).
```

12. Considere um grafo com custos associados aos arcos que conectam os nós. Escreva um programa Prolog para encontrar o custo mínimo para ir de um nó A para um nó B do grafo.

13. Defina o predicado `height(BinaryTree,Height)` para calcular a altura de uma árvore binária. Assuma que a altura da árvore vazia é 0, e que a altura de uma árvore de um elemento é 1.

14. Defina um predicado para reconhecer se um termo Prolog é uma lista.

15. Os predicados `sub1`, `sub2` e `sub3`, abaixo, implementam uma relação de listas. O predicado `sub1` tem uma definição mais procedural do que os predicados `sub2` e `sub3`. Estes dois últimos são escritos em estilo mais declarativo. Estude o comportamento, no que diz respeito à eficiência, destes três predicados. Dois deles têm mais ou menos a mesma eficiência. Quais deles? Por que o outro é mais ineficiente?

```
sub1(List,Sublist) :- prefix(List,Sublist).
sub1([_|Tail],Sublist) :- sub1(Tail,Sublist).

prefix(_,[]).
prefix([X|List1],[X|List2]) :- prefix(List1,List2).

sub2(List,Sublist) :- conc(List1,List2,List),
                      conc(List3,Sublist,List1).

sub3(List,Sublist) :- conc(List1,List2,List),
                      conc(Sublist,_,List2).
```

16. Defina a relação `reverse(List,RevList)`, onde os argumentos são representados por diferença de listas.

17. Utilize o predicado **bagof/3** para definir a relação `powerset(Set,Subsets)` que compute o conjunto de todos os subconjuntos de um determinado conjunto. (represente todos os conjuntos por listas).

18. Defina a relação `alv(Tree)` para testar se uma árvore binária é AVL, isto é, todas subárvores devem diferir em altura de no máximo 1. Represente a árvore binária através do termo: `t(Left,Root,Right)` ou `nil` se a subárvore estiver vazia.

19. Escreva um programa em Prolog que retorne os 'n' primeiros termos de uma série de Fibonacci e sua soma, i.e., `?- fib(3,L,S)` deveria retornar `L = [0,1,1]` e `S = 2`.