Serviço Público Federal
Universidade Federal de Sergipe
Centro de Ciências Exatas e Tecnologia
Departamento de Ciência da Computação

Avaliação de Lógica para Computação - 2017/2

Aluno(a):
Professor: Jânio Coutinho Canuto

1) **(3,0)** Considere os seguintes fatos onde `pai(X,Y)` é verdade se `X` é pai de `Y`:

```
pai(a, b).
pai(a, c).
pai(b, d).
pai(b, e).
pai(c, f).
```

a) Defina um predicado `irmao(X,Y)` que é verdade se `X` é irmão de `Y`.

b) Defina um predicado `primo(X,Y)` que é verdade se `X` é primo de `Y`.

c) Defina um predicado `neto(X,Y)` que é verdade se `X` é neto de `Y`.

d) Defina um predicado `descendente(X,Y)` que é verdade se `X` é descendente de `Y`.

e) Defina um predicado `avo(X)` que é verdade se `X` é avô.

f) Faça a árvore de busca (algoritmo da resolução) para um dos predicados definidos anteriormente. Expanda a árvore até encontrar a segunda solução (se houver). Você pode desenhar a árvore ou descrever os passos.

2) **(2,0)** Um mapa de um certo jogo é composto por várias salas numeradas, conectadas por portais. Alguns pares de salas estão conectados, outros não. Os portais permitem que o jogador passe de uma sala para outra, mas apenas em um sentido, ou seja, o portal que leva da sala 1 para a sala 2 não permite que o jogador retorne da sala 2 para a sala 1. Considere a seguinte lista de conexões, onde `c(X,Y)` é verdade se é possível ir de `X` para `Y`:

```
c(1, 2).  c(3, 4).  c(5, 6).  c(7, 8).  c(9, 10).  c(12, 13).
c(13, 14).  c(15, 16).  c(17, 18).  c(19, 20).  c(4, 1).  c(6, 3).
c(4, 7).  c(6, 11).  c(14, 9).  c(11, 15).  c(16, 12).  c(14, 17).  c(16, 19).
```

a) Defina um predicado `vai(X,Y)` que é verdade se existe um caminho, direto ou indireto, que leva o jogador da sala `X` para a sala `Y`.

b) Há alguma sala inalcançável a partir das demais? Qual?

c) Quais salas deixam o jogador preso?

d) Qual o menor número de sala a partir da qual é possível chegar na sala 20?

e) Suponha agora que as conexões são bi-direcionais. O predicado que você definiu no primeiro item ainda funciona? Por que?

3) **(2,0)** Uma empresa de viagens tem as seguintes informações:

```
onibus(saoPaulo, campinas).
onibus(campinas, paulinea).
onibus(valmont, saarbruecken).
onibus(valmont, metz).

trem(metz, frankfurt).
trem(saarbruecken, frankfurt).
trem(metz, paris).
trem(saarbruecken, paris).

aviao(frankfurt, bangkok).
aviao(frankfurt, singapura).
aviao(paris, losAngeles).
aviao(bangkok, saoPaulo).
aviao(singapura, saoPaulo).
aviao(losAngeles, saoPaulo).
```

onde `onibus(X,Y)` é verdade se há uma linha de ônibus que leva da cidade `X` à cidade `Y`, `trem(X,Y)` é verdade se há uma linha de trem que leva da cidade `X` à cidade `Y` e `aviao(X,Y)` é verdade se há uma vôo que leva da cidade `X` à cidade `Y`.

a) Defina um predicado `viagem(X,Y)` que é verdade se existe um caminho, direto ou indireto, que leva da cidade `X` à cidade `Y`. Ex: `viagem(paris, paulinea)` deve retornar Verdadeiro.

b) Defina um predicado `viagem(X,Y,Z)` que, além de dizer se há um caminho, diz quais as cidades visitadas. Ex: Para a consulta `viagem(paris, campinas, Z)` devemos ter `Z = vai(paris, losAngeles, vai(losAngeles, saoPaulo, vai(saoPaulo, campinas)))`.

4) **(2,0)** O clássico problema das torres de Hanoi consiste em, dado um conjunto de N discos de tamanhos distintos, empilhados em ordem descrescente de tamanho (com o menor embaixo) em um dado local A, movê-los para um certo local B tendo um local C como auxiliar. Esse processo de transferência deve obedecer às seguintes restrições:

- apenas um disco pode ser movido por vez;
- o movimento consistem em retirar um disco do topo de uma das pilhas (A, B ou C) e colocá-lo no topo de outra pilha;
- um disco maior nunca pode ser empilhado sobre um disco menor.

Uma forma interessante de pensar no raciocínio recursivo é simplesmente assumir que é possível resolver uma instância mais simples do seu problema. Para o caso do problema das torres de Hanoi, vamos assumir que sabemos resolver o problema com $(N-1)$ discos. Sendo assim, a solução do problema para N discos é simples: transfira $(N-1)$ discos da posição A para a posição C usando B como auxiliar, mova o disco restante de A para B e, por fim, mova os $(N-1)$ discos que estão em C para B usando A como auxiliar. Obviamente, se N é 1 basta movê-lo diretamente para onde você quiser.

a) Defina um predicado `hanoi(N,A,B,C)` que resolve o problema das torres de Hanoi com N discos. O movimento de um disco de `X` para `Y` deve ser escrito na tela usando o predicado `move(X,Y) :- write('Move da haste'), write(X), write('para a haste '), write(Y), nl.`

5) **(1,0)** Quando aplicamos o método da resolução a uma fórmula lógica, tanto proposicional quanto de predicados, e chegamos em uma cláusula vazia, sabemos que a fórmula é insatisfatível. No entanto, na programação lógica (que é baseada no método da resolução) dizemos que a consulta é consequência lógica do programa quando encontramos a cláusula vazia. Em um caso a cláusula vazia representa contradição e no outro representa tautologia. Mostre que não há incoerência entre as duas interpretações.