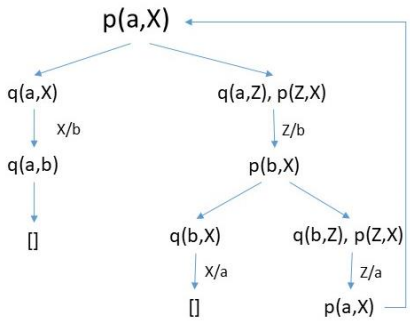
Gabriel Augusto Requena dos Reis – 16.2.8105
Lista 01 – CSI419 - LINGUAGENS DE PROGRAMACAO - Turma 21

1. Mostre no que resulta a unificação dos seguintes pares de termos. Mostre as substituições obtidas em cada unificação.
   (a) p(X,f(X)) ≡ p(Y,f(a))
   {Y/X, X/a}
   (b) p(f(X),Y,g(Y)) ≡ p(Y,f(a),g(a))
   Não tem unificação pois Y tenta se tornar dois predicados diferentes (Y/f(a) e Y/a).
   (c) p(X,Y,X) ≡ p(X,f(X))
   Não tem unificação, pois a quantidade de elementos é diferente.

2. Quantas provas para o objetivo ← p(a,X) podem ser obtidas a partir do seguinte programa ? Produza a arvore SLD do problema.



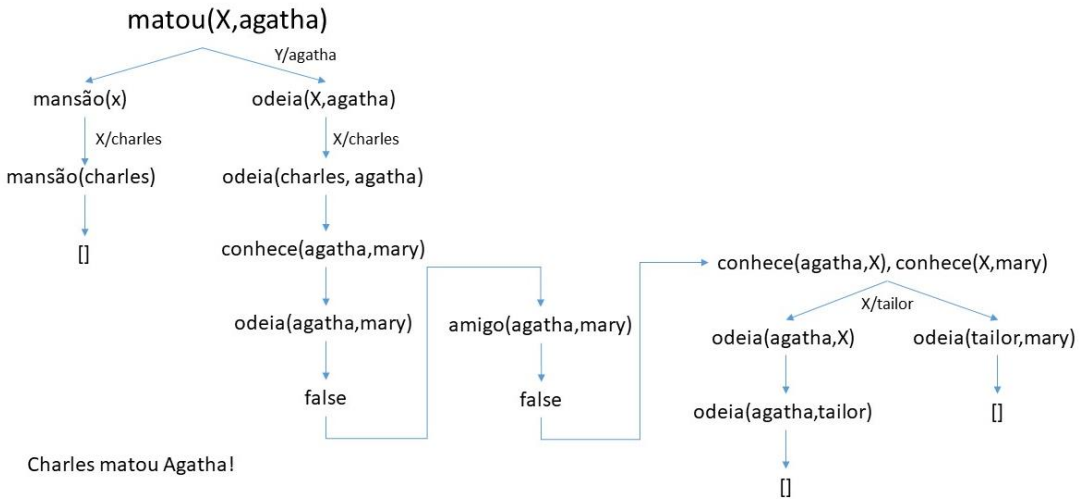
3. Houve um assassinato na mansão Rose Red. Os investigadores reuniram todos os fatos que conhecem acerca do crime e das pessoas que estavam na mansão na hora do crime.

   a) Modele o conjunto de afirmações acima na forma de cláusulas. Algumas destas afirmações podem não ser relevantes para solução do problema.
   b) Como base no seu modelo, como você pode descobrir o assassino? Defina uma clausula objetivo que permita determinar o assassino.

A e b)

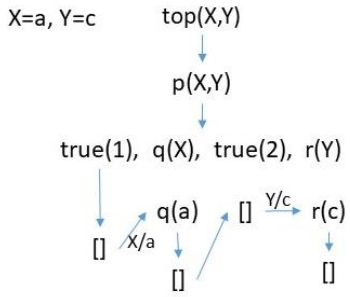
```
matou(X,Y) :- mansao(X), odeia(X,Y).
        mansao(charles).
        mansao(agatha).
        mansao(mary).
        mansao(tailor).
        mansao(mordomo).

        conhece(A,B) :- odeia(A,B).
        conhece(A,B) :- amigo(A,B).
        conhece(A,C) :- conhece(A,B), conhece(B,C).

        odeia(tailor,mary).
        odeia(agatha,tailor).
        odeia(mary,tailor).
        odeia(mary,mordomo).
        odeia(charles, X) :- conhece(X,mary).

        amigo(mordomo,agatha).
        amigo(charles,mary).
        amigo(agatha,mordomo).
```

c) Usando seu programa lógico obtido em e a) clausula objetivo obtida em b), mostre a resolução SLD para o problema.

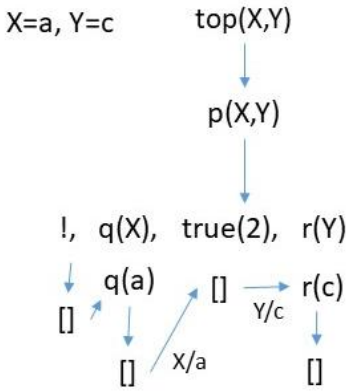


4. Considere o seguinte programa:

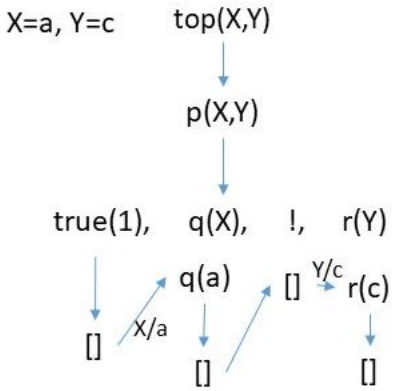
Mostre a resolução SLD para o objetivo ← top(X,Y) e mostre os ramos cortados quando:



(a) true(1) é substituído por um corte.



(b) true(2) é substituído por um corte.



5. Considere a seguinte implementação para representação de números naturais:

nat(zero).

nat(s(X)) ← nat(X).

(a) Defina um predicado que realiza soma de dois números naturais.

```
plus(zero,X,X)
plus(s(X),Y,s(Z)) :- plus(X,Y,Z)
```

(b) Defina um predicado para a multiplicação.

```
mult(zero,_,_) :- zero.
mult(_,zero,_) :- zero.
mult(X,s(zero), Z is plus(X,zero,R)).
mult(X,Y,Z) :- mult(X,Y is Y-1, Z is plus(X,Z,R) ).
```

6. Listas podem ser utilizadas para implementar conjuntos. Defina os seguintes predicados que permite manipular uma lista como um conjunto.

a) elem(X, XS) : Sucede se X pertencer a XS.

```
elem(X,[X|_]) :- !.

elem(X,[_|C]) :- elem(X,C).
```

b) union(XS, YS, ZS) : Sucede se ZS for a união de XS e YS.

```
uni([],L,L).

uni([X|Xs], Ys, [X,Zs] :- uni(Xs, Ys, Zs).
```

c) inter(Xs, Ys, Zs) : Sucede se Zs for a interseção Xs e Ys.

```
inter([X|Y], L, [X|Z]) :- elem(X,L), inter(Y,L,Z).
inter([_|X], L, Y) :- inter(X,L,Y).
inter(_,_,[]).
```

d) nat2int(X, Y) : Sucede se Y for a representação do número X em número arábicos.

```
nat2int(zero, X) :- X.
nat2int(s(Y), X) :- nat2int(Y, X is X+1).
```

7. Defina um predicado que diz se uma lista é um palíndromo. Um palíndromo é uma sequência de símbolos que idêntica se lida em da esquerda para direita ou em ordem reversa. (Para resolução desta questão é proibido usar o predicado reverse ou similares.)

```
inverter([],[]).

inverter([X|L1], L):- inverter(L1,Y), append(Y, [X], L).

palin(X):- inverter(X, L), X==L.
```

8. Considere a definição formal de substituição. Mostre substituições θ, σ e γ, sendo θ, σ, γ 6= {}, tal que:

(a) θ = θθ
{X/B}

(b) θσ = σθ
θ{X/J}, σ{X/J}

(c) θ(σγ) = (θσ)γ
θ{X/Z}, σ{X/Z}, γ{X/Z}

9. Classifique as linguagens de programação que você usa quanto ao paradigma de programação.

Java	Fraca	Dinâmica
Java script	Fraca	Dinâmica
R	Fraca	Dinâmica
C	Fraca	Estática
C#	Forte	Estática

10. Escreva funções, em uma linguagem imperativa de sua escolha, que resolvam cada um dos problemas apresentados a seguir. (Não é permitido utilizar comandos de repetição.)

a. Somar todos os elementos de em array de números inteiros.

```
public static int soma(ArrayList<Integer> vetor){
        if(vetor.size() <= 0)
                return 0;
        int primeiro = vetor.get(0);
        vetor.remove(0);
        return (primeiro+soma(vetor));
}
```

b. Fazer o produto de todos os elementos de um array de números inteiros.

```
public static int prod(ArrayList<Integer> vetor){
        if(vetor.size() <= 0)
                return 1;
        int primeiro = vetor.get(0);
        vetor.remove(0);
        return (primeiro * prod(vetor));
}
```

c. Calcular o fatorial de um número inteiro.

```
public static int fatorial(int x) {
        if(x < 1)
                return -1;
        if(x == 1)
                return x;
        return x*(fatorial(x-1));
}
```

d. Encontrar o maior elemento em um vetor de inteiros.

```
public static int maior(ArrayList<Integer> vetor) {
        int neg = (int) Float.NEGATIVE_INFINITY;
        return (maior(vetor, neg));
}
public static int maior(ArrayList<Integer> vetor, int maior) {
        if(vetor.size() <=0)
                return maior;

        int aux = vetor.get(0);
        vetor.remove(0);

        if(aux > maior)
                return (maior(vetor, aux));
        else
                return (maior(vetor, maior));
}
```

e. Encontrar o menor elemento em um vetor de inteiros.

```
public static int menor(ArrayList<Integer> vetor) {
        int pos = (int) Float.POSITIVE_INFINITY;
        return (menor(vetor, pos));
}
public static int menor(ArrayList<Integer> vetor, int menor) {
        if(vetor.size() <=0)
                return menor;

        int aux = vetor.get(0);
        vetor.remove(0);

        if(aux < menor)
                return (menor(vetor, aux));
        else
                return (menor(vetor, menor));
}
```

f. Encontrar dois valores x e y, tais que x + y = z, onde z é valor inteiro dado.

```
public static void EncontraSoma(int Z) {
        if(Z>=0)
        EncontraSoma(Z,0,0,true);
}
public static void EncontraSoma(int Z, int X, int Y,boolean div) {
        if(X+Y == Z) {
                System.out.println(X +"+" +Y +"=" +Z);
                return;
        }

        if(div == true)
                EncontraSoma(Z, X+1, Y, false);
        else
                EncontraSoma(Z, X, Y+1, true);
}
```

g. Obter o n-ésimo termo da série de Fibonacci.

```
public static int NesimoFibonacci(int n) {
        if (n < 2) {
                return n;
        }
        return NesimoFibonacci(n - 1) + NesimoFibonacci(n - 2);
}
```

h. Obter a lista de fatores primos de um número

```
public static ArrayList<Integer> fatoresprimos(int valor, ArrayList<Integer>
resposta) {
	if (valor == 1) {
		return resposta;
	}
	else {
		int num = menornumeroprimo(valor,2);
		resposta.add(num);
		valor = valor/resposta.get(resposta.size()-1);
		fatoresprimos(valor, resposta);
	}
	return resposta;
}

public static int menornumeroprimo(int valor, int divisor){
	if(divisor == valor)
		return valor;

	if(EPrimo(divisor,2) && (valor%divisor==0))
		return divisor;
	else
		return menornumeroprimo(valor,divisor+1);
}

private static boolean EPrimo(int num, int aux) {
	if(aux == num)
		return true;
	if(num%aux == 0)
		return false;

	return (EPrimo(num,aux+1));
}
```