

Índice:

Tema	página
1. Cálculo proposicional	2
2. Termos constantes	2
3. Forma sentencial	2
4. Tautologia	3 – 4
5. Fato	4
6. Contradição	4 – 5
7. Introdução ao Prolog	5 – 7
8. Regras recursivas	7 – 9
9. Estruturas	10 – 12
10. Termo	12 – 16
11. Lista	17 – 21
12. Máximo de uma lista	21
13. União	21 – 22
14. Intersecção	22
15. AFN	22

Cálculo Proposicional

Termos Constantes

F / V

Letra Proposicionais

Conectivos Lógicos

...

...

...

- verdadeiro e falso.

- a , b , a1 , b1 , letras maiúsculas.

- conjunção (E) \wedge

- disjunção (OU) \vee

- negação (NÃO) \neg

- implicação \Rightarrow

- equivalência \Leftrightarrow

Prioridade (ordem): $\{\{\{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow\}\}\}$

TABELA VERDADE

a	$\neg a$
V	F
F	V

conjunção (E) \wedge

a	b	(a \wedge b)
F	F	F
F	V	F
V	F	F
V	V	V

disjunção (OU) \vee

a	b	(a \vee b)
F	F	F
F	V	V
V	F	V
V	V	V

implicação \Rightarrow

a	b	(a \Rightarrow b)
F	F	V
F	V	F
V	F	V
V	V	F

equivalência \Leftrightarrow

a	b	(a \Leftrightarrow b)
F	F	V
F	V	F
V	F	F
V	V	V

Sintaticamente é correto ter símbolos e conectivos: NÃO pode = $P \wedge \wedge Q \Rightarrow \Rightarrow R$, isto chama-se **FBF (Fórmula Bem Formada)**

a	b	a \wedge b	a \vee b	a \Rightarrow b	a \Leftrightarrow b
F	F	F	F	V	V
F	V	F	V	V	F
V	F	F	V	F	F
V	V	V	V	V	V

Forma Sentencial

1. Toda letra proposicional é uma forma sentencial.
2. Se A e B são formas sentenciais, então:

Veja abaixo, exemplo de formas sentenciais:

$A \wedge B$
 $A \vee B$
 $\neg A$
 $A \Rightarrow B$
 $A \Leftrightarrow B$

Exemplo:

- a
- $(a \wedge b)$
- $((a \vee b) \wedge c)$
- $(a \vee b) \wedge c$

Prioridade cresce na ordem da esquerda p/direita: $\Leftrightarrow, \Rightarrow, \vee, \wedge, \neg$

$a \vee b \wedge c \Rightarrow (a \vee (b \wedge c))$
 $a \vee b \Rightarrow c \wedge d \Rightarrow ((a \vee b) \Rightarrow (c \wedge d))$
 $a \Rightarrow b \Rightarrow c \Rightarrow ((a \Rightarrow b) \Rightarrow c)$

((a \Rightarrow b) \Rightarrow c)

		V	V	V		V	V	
		V	V	V		F	F	
		V	F	F		V	V	
		V	F	F		V	F	
		F	V	V		V	V	
		F	V	V		F	F	
		F	V	F		F	V	
		F		F			F	

(a \Rightarrow (b \Rightarrow c))

	V	V		V	V	V		
	V	F		V	F	F		
	V	V		F	V	V		
	V	V		F	V	F		
	F	V		V	V	V		
	F	V		V	F	F		
	F			F		V		
	F			F		F		

n letras proposicionais

linhas da tabela é 2^n

• TAUTOLOGIA:

Uma forma sentencial é uma tautologia se, independente dos valores verdade atribuídos as letras sentenciais, que o valor verdade da forma sentencial é V.

A	→	B	↔	¬ A	∨	B
V	V	V	V	F	V	V
V	F	F	V	F	F	F
F	V	V	V	V	V	F
F	F	F	V	V	V	V

Coluna que representa o valor verdade da forma seqüencial.

∴ $A \Rightarrow B \Leftrightarrow \neg A \vee B$ é uma TAUTOLOGIA.

$A \vee \neg A$ é uma TAUTOLOGIA.

$A \Rightarrow A$ é uma TAUTOLOGIA

$A \Rightarrow B \Leftrightarrow \neg B \Rightarrow \neg A$

$a \Rightarrow b \Leftrightarrow \neg a \vee b$

$(a \vee b) \Rightarrow b \Leftrightarrow \neg (a \vee b) \vee b$

- FATO:**

Se uma forma sentencial é uma tautologia, então, **qualquer forma sentencial obtida da original** trocando-se cada ocorrência de uma letra sentencial também é uma tautologia.

- CONTRADIÇÃO:**

Uma forma sentencial é uma contradição se, **seu valor verdade é F**, para toda **atribuição de verdade** às suas letras sentenciais.

FATO: A é uma tautologia se e somente se $\neg A$ é uma contradição.

FATO: Suponha que A e B são formas sentenciais. Se A e $A \Rightarrow B$ são tautologias, então B é uma tautologia.

Verificação: Como $A \Rightarrow B$ são tautologias, então qualquer atribuição de verdade as letras sentenciais que ocorrem em A e $A \Rightarrow B$, tal que, o valor verdade de A e $A \Rightarrow B$ é V (verdade).
Suponho que, por contradição, que exista uma atribuição de verdade às letras seqüenciais de B que tornam B falsa.

Como A é V (verdade), para toda atribuição de verdade, segue que, $A \Rightarrow B$ é falsa para a atribuição considerada para B . Isto contraria $A \Rightarrow B$ ser uma tautologia.

Definições: Se $A \Rightarrow B$ é uma tautologia, então dizemos que, **B** é uma consequência lógica de **A**.
Se $A \Rightarrow B$ é uma tautologia, então dizemos que, **A** é logicamente equivalente a **B**.

$$A \Rightarrow B \quad \Leftrightarrow \quad \neg A \vee B$$

Verificar que: $\neg, \Rightarrow, \Leftrightarrow, \vee, \wedge$

\neg, \Rightarrow são suficientes para escrever qualquer forma sentencial envolvendo $\neg, \Rightarrow, \Leftrightarrow, \vee, \wedge$

- conjunção (**E**) \wedge
- disjunção (**OU**) \vee
- negação (**NÃO**) \neg
- implicação \Rightarrow
- equivalência \Leftrightarrow

Introdução ao PROLOG

Fatos:

progenit (parent)

parent (tom , bob)
relação ou predicado átomos

parent (tom , bob)
parent (pam , bob)
parent (tom , liz)
parent (bob , ana)
parent (bob , pat)
parent (pat , jim)

❖ Consultas: QUERYS

?- parent (bob , pat)
yes

?- parent (liz , pat)
no

❖ Variáveis Lógicas (letra maiúscula)

Uma VAR representa um objeto não específico.

?- parent (X , liz)

X = tom

?- parent (bob , X)

X = ana

X = pat

?- parent (X , Y)

X = tom Y = bob

X = pam Y = bob

❖ Consultas Compostas

Quem é um dos Avós de jim ?

(1) X é um progenitor de Y e

(2) Y é um progenitor de jim

?- parent (X , Y) , parent (Y , jim)

X = bob Y = pat

Será que ANA e PAT têm um progenitor comum ?

?- parent (X , pat) , parent (X , ana)

X = bob

❖ Regras

A é dito a cabeça da regra e B1, B2, ... , Bn é o corpo da regra. (veja abaixo a sequência do Prolog.)

female (pam)

male (tom)

male (bob)

female (liz)

female (pat)

female (ana)

male (jim)

Queremos deduzir a Relação MOTHER (mãe):

$\text{mother}(X) := \text{female}(X), \text{parent}(X, Y)$

tal que, X é mulher então X é não.

.....para todo X, se existe um Y

?- $\text{mother}(X)$quem é a mãe de ?

X = bob ;

X = pam;

Queremos deduzir a Relação FATHER (pai):

$\text{father}(X) := \text{male}(X), \text{parent}(X, Y)$.

$\text{father}(X, Y) := \text{male}(X), \text{parent}(X, Y)$.

$\text{father}(X) := \text{father}(X, Y)$.

Queremos deduzir a Relação UNCLE (tio):

$\text{uncle}(X, Y) := \text{male}(X), \text{parent}(Z, Y), \text{twins}(X, Z)$.

$\text{twins}(X, Y), \text{parent}(Z, Y), \text{parent}(Z, X), X \neq Y$.

?- $\text{twins}(X, Y)$quem é a mãe de ?

X = bob Y = liz

X = ana Y = pat

?- $\text{twins}(\text{bob}, \text{bob})$errado

yes

Queremos deduzir a Relação SON (filho):

$\text{son}(X, Y) := \text{male}(X), \text{parent}(Y, X)$.

Exercício: Escreva o predicado $\text{CONSIN} : (X, Y)$ que é verdadeiro sempre que X for primo de Y.

$\text{consinparent}(X, Y) := \text{parent}(Z, Y), \text{parent}(W, Y), \text{twins}(Z, W)$.

$\text{grandparent}(X, Y) := \text{parent}(X, Z), \text{parent}(Z, Y)$.

$\text{grandgrandparent}(X, Y) := \text{parent}(X, Z), \text{grandparent}(Z, Y)$.

$\text{ancestral}(X, Y) := \text{parent}(X, Y)$.

$\text{ancestral}(X, Y) := \text{parent}(X, Z), \text{ancestral}(Z, Y)$.

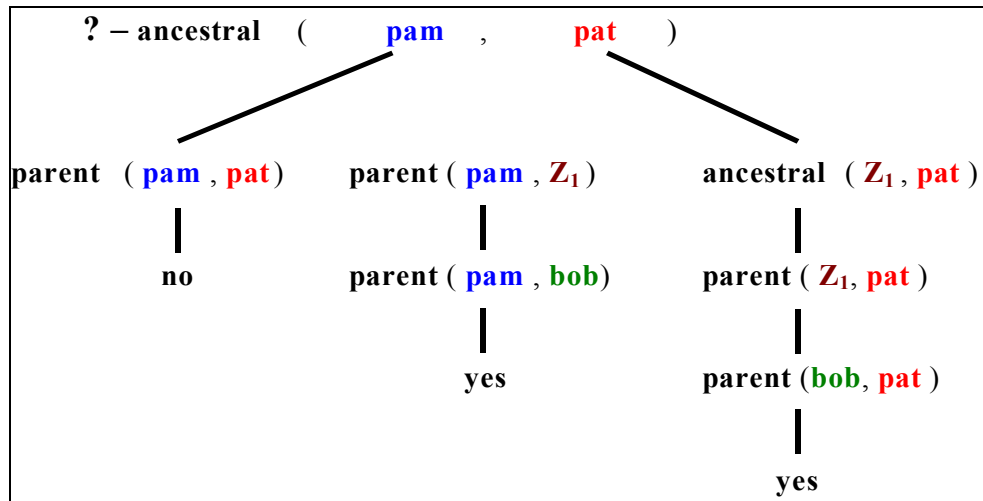
Regras Recursivas:

$\text{parent}(\text{pam}, \text{bob})$

$\text{parent}(\text{bob}, \text{pat})$

$\text{parent}(\text{pat}, \text{jack})$

$\text{parent}(\text{tom}, \text{bob})$



ESTRUTURAS

? bordate (**pam**, date (1 , may , 1870)),

‘
‘
‘

? bordate (**pam**, X),

X = date (1 , may , 1870).

- Como chamar todas as pessoas nascidas em MAIO?

? bordate (X, date (Y , may , Z)),

X = **pam** (Y =1 , Z = 1870),

‘
‘
‘

? bordate (X, date (____ , may , ____)),

X = **pam**,

X = **jack**.

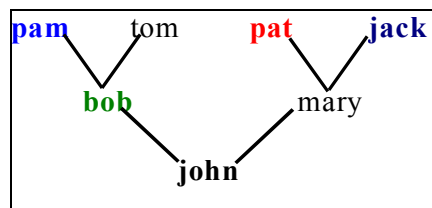
Significa qualquer coisa

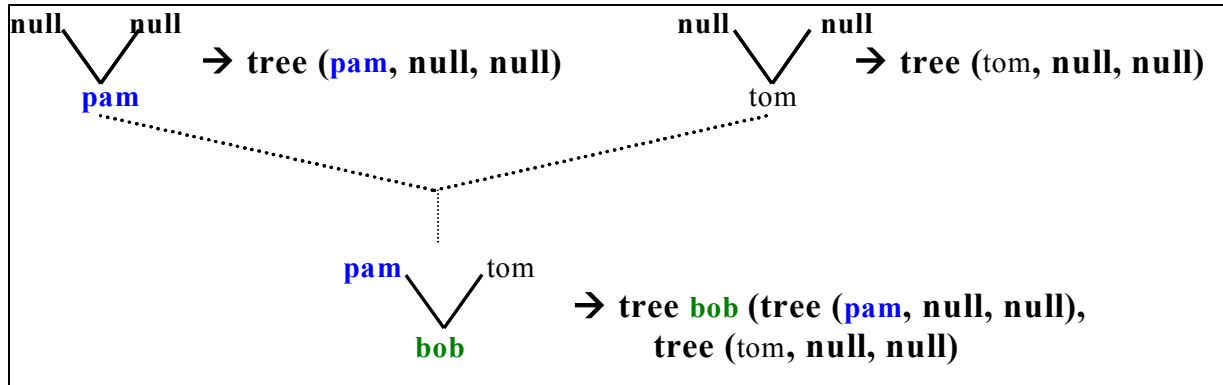
- Pessoas que nasceram no mesmo DIA e MÊS?

? bordate (X, date (D , M , ____)),

? bordate (Y, date (D , M , ____)), X \ = Y.

Significa qualquer coisa





Última TREE:

$\rightarrow \text{tree}(\text{john}, \text{tree}(\text{bob}, \text{tree}(\text{pam}, \text{null}, \text{null}), \text{tree}(\text{tom}, \text{null}, \text{null})), \text{tree}(\text{mary}, \text{tree}(\text{pat}, \text{null}, \text{null}), \text{tree}(\text{jack}, \text{null}, \text{null})))$

Tree member

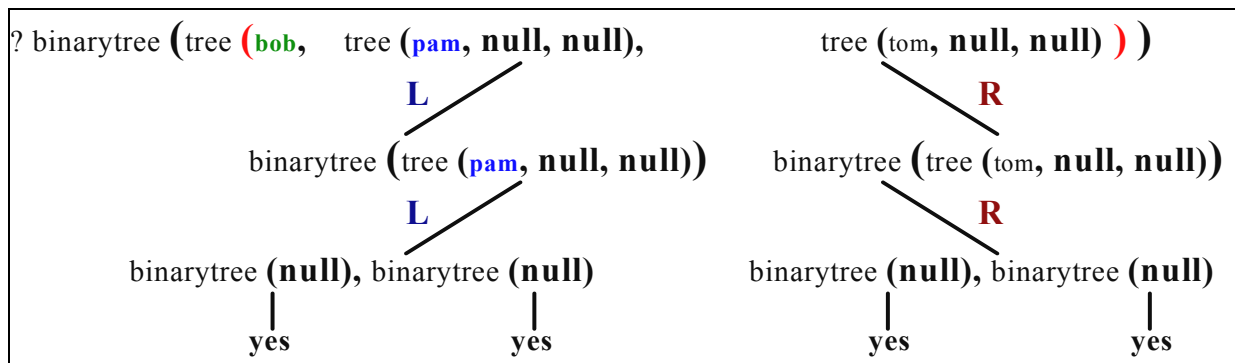
L = esquerda, **R** = direita, **E** = elemento

binarytree (null)

binarytree (tree (**E**, **L**, **R**)):-
binarytree (**L**), binarytree (**R**).

? binarytree (tree (**pam**, **null**, **null**)).

yes



Uma substituição é um conjunto de pares da forma $X_i = t_i$, onde X_i é uma variável e t_i é um termo, e $X_i \neq X_j$ para todo $i \neq j$, e X_i não ocorre em t_j , para nenhum j .

Se A é um termo e Θ é uma substituição, então $A\Theta$ denota o termo obtido de A , pela troca de cada por X_i por t_i em A , onde $\Theta = \{ X_1 = t_1, \dots, X_n = t_n \}$

Exemplo:

- parent (X , bob)
 $\{ X = \text{pat} \}$
 parent (X , bob) $\{ X = \text{pat} \}$ é o termo parent (pat , bob)

- tree (X , tree (pam , null , null) , Y) =: A
 $\{ X = \text{bob} , Y = \text{tree}(\text{bob} , \text{null} , \text{null}) \} =: \Theta$

$A\Theta$ é o termo tree (bob , tree (pam , null , null) , tree (tom , null , null)

para t (X , bob) =: A
 $\{ X = Y \} =: \Theta$

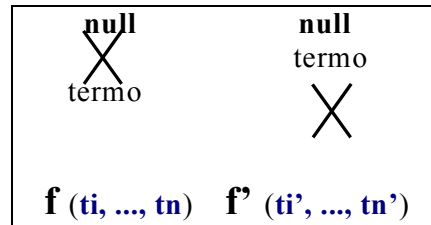
$A\Theta$ é o termo parent (Y , bob)
 [é uma variante de A]

Um termo C é uma instância comum de termos A e B, se existem substituições Θ_1 e Θ_2 , tais que, $C = A\Theta_1$ e $C = B\Theta_2$.

Exemplo:

- A = parent (X , tom) , B = parent (pam , Y)
 C = parent (pam , tom)
 $\Theta_1 = \{ X , \text{pam} \}$ $\Theta_2 = \{ Y , \text{tom} \}$

Entrada: Dois termos **A** e **B** unificam se possuem uma instância comum.



Entrada: dois termos **A** e **B**.

Saída: **SIM** se unificam e **NÃO**, caso contrário:

- Se **A** e **B** são átomos, então devolva:
SIM se **A** e **B** são iguais e **NÃO** em caso contrário.
- Se **A** é uma variável e **B** é um termo, então **A** e **B** se unificam pela substituição **A = B**, devolva **SIM** neste caso.
- Se **A** é da forma **f (t₁, ..., t_n)** e **B** é da forma **f' (t₁', ..., t_n')**, então **A** e **B** unificam se **f** e **f'** são os mesmos átomos **n = m**, **t_i** para **1 ≤ i ≤ n**; neste caso, devolva **SIM**, caso contrário devolva **NÃO**.

Este algoritmo, tomando-se as substituições calculadas, elas determinam um unificador **mais geral** dos termos **A** e **B**.

parent (X , tom) , parent (Y , tom)
{ X = pam , Y =tom }

Um unificador **mais geral** é { X = Y }

Entrada: Uma lista **G₁, ..., G_n** de objetivos e um programa **P**.

Saída: **SIM (YES)** se **G₁, ..., G_n** é uma conseqüência lógica de **P** e **NÃO (NO)**, caso contrário:

- Se **n = 0**, então devolva **YES**.
- Se **n ≥ 1**, então considere objetivo **G₁**.

Encontre fazendo uma busca de “uma para baixo” a primeira cláusula **H: - B₁ ... B_n**, tal que,

exista uma substituição Θ , tal que $G_1 \Theta = H \Theta$.

Renomeie as variáveis de $H: - B_1 \dots B_n$, de tal forma que, nenhuma apareça em G_2, \dots

G_n . Denote por $H': - B'_1 \dots B'_n$, a variante sintática obtida.

Seja α um unificador mais geral de H' de G_1 , submeta a lista de objetivos $B'_1 \alpha \dots B'_n \alpha$,

$G_2 \alpha \dots G_n \alpha$, ao algoritmo que estamos definindo. Caso a chamada recursiva devolva **YES**,

então devolva **YES**. Caso contrário, volte para o passo de busca na próxima cláusula que segue

$H: - B_1 \dots B_n$, com a lista de objetivos $G_1, \dots G_n$.

- Caso não exista nenhuma cláusula cuja cabeça unifique com G_1 devolva **NO**.

Considere o programa:

1. $\text{path}(X, Z) :- \text{arc}(X, Y), \text{arc}(Y, Z)$
2. $\text{path}(X, X).$
3. $\text{arc}(b, c).$

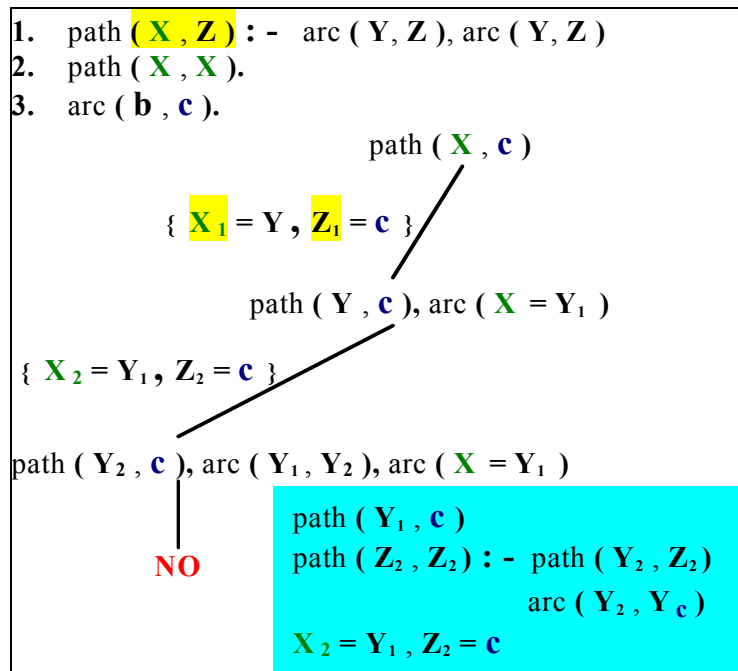
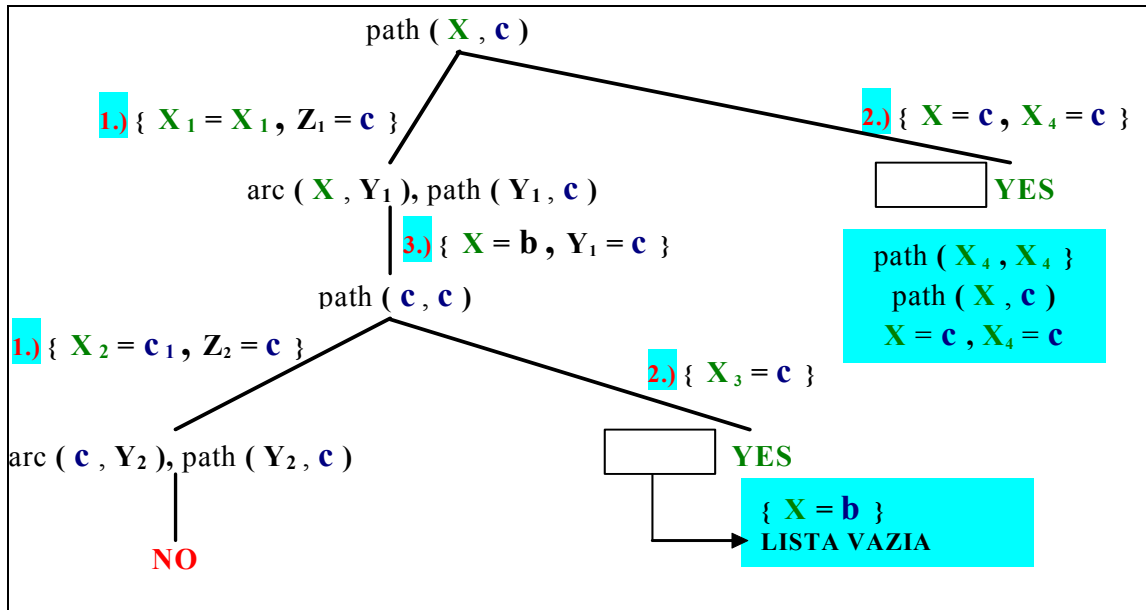
Chamada
recursiva

Considere a consulta:

? $\text{path}(X, c)$

Existe X , tal que, haja um caminho de X até c .

ÁRVORE DE REFUTACÃO



$\cdot (\cdot (\text{pam}, \cdot (\text{pat} []))) , \cdot (\cdot (\text{pam}, \cdot (\text{bob} [])))$

Notação Visual

[]

$\cdot (\text{pam}, [])$

Notação Colchetes

[]

[(pam / [])]

[(pam / [pat / []])]

Notação Simplificada

[]

[pam]

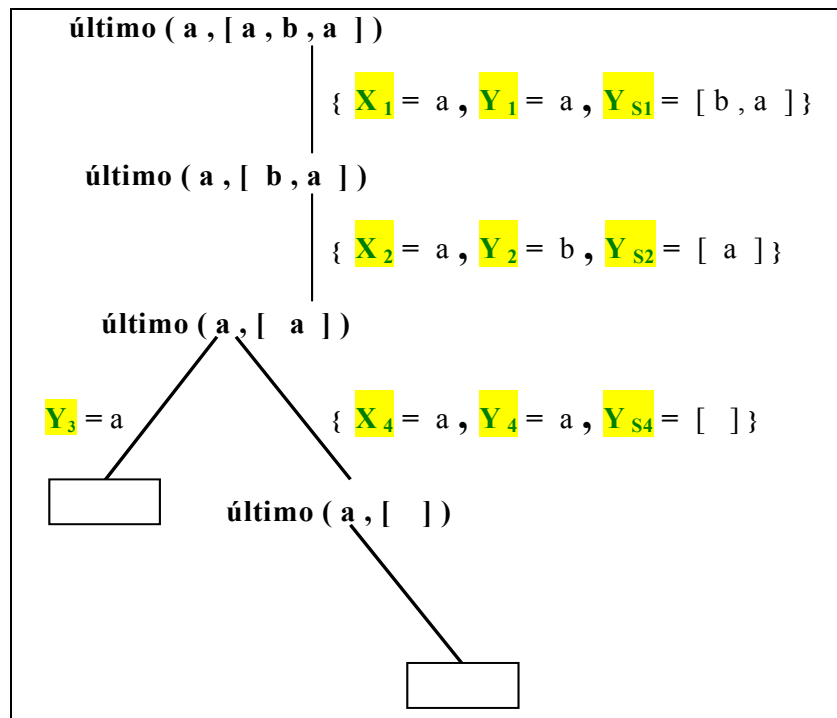
[pam , pat]

% último (X , Xs) : X é o último elemento de Xs.

1º) último (X , [X]) .

2º) último (X , [Y , Ys]) := (X , Ys) .

? último (a , [a , b , a]) .



% prefixo (Xs1 , Xs) : Xs é um prefixo da lista Ys.

% prefixo ([a , b] , [a , b , c]) yes

% prefixo ([] , [a , b , c]) yes

```
% prefixo ( [ a , c ] , [ a , c ] )      yes
% prefixo ( [ ] , Y s )
% prefixo ( X / X s ) , ( Y / Y s ) := prefixo ( Xs / Y s )
```

% sufixo (Xs , Ys): a lista Xs é um sufixo da lista Ys.

```
% sufixo ( [ ] , Y s )      .....não precisa está linha no processo.
% sufixo ( Xs / Y s )
% sufixo ( X s ) , [ Y / Y s ] := sufixo ( Xs / Y s ) .
```

Exercício:

1) Escreva um **predicado ordenada** (L , S), cujo significado pretendido é a lista S, e o resultado de ordem é a lista L. Por isso, use a relação “= <”(menor ou igual) de Prolog.

```
% ordenada ( [ 3 , 4 , 1 , 2 ] , [ 1 , 2 , 3 , 4 ] )
yes.
```

```
% ordenada ( [ 3 , 1 , 2 ] , [ 1 , 2 , 3 , 4 ] )
```

2) Escreva um **predicado ordenada** (L), que responda YES, sempre que a lista L estiver ordenada.

```
% ordenada ( [ 1 , 3 , 5 , 7 ] )
yes.
```

```
% ordenada ( [ 1 , 2 , 4 , 1 ] )
no.
```

```
1: ordenada ( [ ] )
2: ordenada ( [ x ] )
3: ordenada ( [ x , y / L ] ) :- x = < y , ordena ( [ y / L ] )
```

3) Escreva um **predicado intercala** (L1 , L2 , S), cujo significado é se L1 e L2 são listas ordenadas, então S é o resultado da ordem de concatenação de L1 e L2.

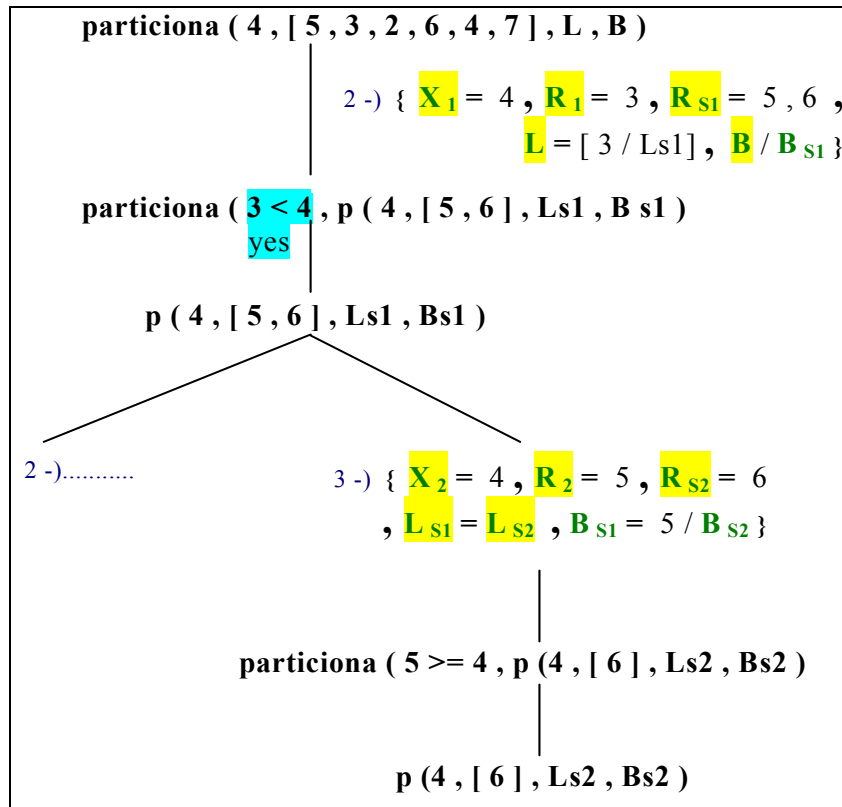
```
% intercala ( [ 2 , 4 , 7 ] , [ 1 , 3 , 6 , 9 ] , S )
```

```
% S = [ 1 , 2 , 3 , 4 , 6 , 7 , 9 ]
```

- 1: intercala ([], A , A)
- 2: intercala (A , [], A)
- 3: intercala ([X , Xs] , [Y , Ys] , [X , Zs]) : - $x \leq y$, intercala (Xs , [Y / Ys] , Zs)
- 4: intercala ([X , Xs] , [Y , Ys] , [X , Zs]) : - $x \geq y$, intercala ([X , Xs] , Ys , Zs)

Exercício:

- 1) Escreva um programa Prolog para máximo (X , L), onde X é o maior elemento da Lista L.
Assim, por exemplo para a consulta
?- máximo (X , [3 , 7 , 4 , 5])
X = 7
- 2) Para o programa do exercício 1, determinar a árvore S/d para a consulta:
?- máximo (X , [3 , 7 , 4 , 5])
- 3) Determine se existe uma unificação.
 - a) $\cdot (Q \cdot (\cdot (R , S) , \cdot ())) = \cdot (\cdot (t , w) , X)$
 - b) $\text{sat} (\text{and} (\text{or} (x , y) , z)) = \text{sat} (\text{and} (z , w))$
- 4) Considere o programa:
edge (a , b) .
edge (b , c) .
path (x , x) .
path (x , y) := (X , Z , path , (Z , Y) .
Qual as respostas para as seguintes perguntas:
 - a) ?- path (a , a)
 - b) ?- path (a , c)
- 5) Escreva um programa em prlog para particionar (X , R , L , B), onde L é a lista constituída dos elementos da lista R, que são estritamente menor que X . B é a lista constituída dos elementos de L que são maiores ou iguais a X.
Por exemplo, para a consulta
?- particiona (4 , [5 , 3 , 2 , 6 , 4 , 7] , L , B), devemos ter como possíveis respostas.....L = [3,2] e R = [5 , 6 , 4 , 7]
 - 1-) particiona (X , [], [], [])
 - 2-) particiona (X , [R / Rs] , [R / Ls] , Bs) : - $r < X$, particiona (X , Rs , Ls , Bs) .
 - 3-) particiona (X , [R / Rs] , Ls , [R / Bs]) : - $r \geq X$, particiona (X , Rs , Ls , Bs) .



6) Usando particionamento do exercício 5, escreva uma versão do algoritmo quicksort em prolog.

MÁXIMO DE UMA LISTA:

% se/máx (X , Y) : Y é o máximo de Xs>

% se/máx (X , [X]).

% se/máx (X , [X / Ys]) : - se/máx (Y / Xs) , X >= Y.

% se/máx (Y , [X / Ys]) : - se/máx (Y / Xs) , Y > X.

UNIÃO de Ys:

% união (Xs , Ys , \cup s) : \cup s é a união de Xs e Ys.

% união ([] , Ys , Ys).

% união ([X / Ys] , Ys , \cup s) : - membro (X / Ys) , união (Xs , Ys , \cup s).

% união ([X / Ys], Ys, [X / \cup s]):- não_membro (X / Ys), união (Xs1 , Ys , \cup s).

INTERSECCÃO:

% inter (Xs, Ys, Is): Is é a intersecção de Xs e Ys.

% inter ([], Ys, []).

% inter ([X / Xs], Ys, [X / Is]):- membro (X / Ys), inter (Xs, Ys, Is).

% inter ([X / Xs], Ys, Is):- não_membro (X / Ys), inter (Xs, Ys, Is).

Um **AFN** é um quintúpla (**Q** , **Σ** , **δ** , **q₀** , **F**), em que >

Q: é o conjunto Finito de Estados,

Σ : é um Alfabeto,

δ : $Q \times \Sigma$:

q₀: é o conjunto de Estado Inicial,

F: é o conjunto de Estados Finais.

