



Lógica de Predicados



Rogério Eduardo da Silva

Proposições não são suficientes

- ▶ A lógica proposicional é uma forma sistemática de se representar conhecimento (proposições) que nos permite inferir novos conhecimentos através de regras de inferência.
 - A capital de Minas Gerais é Belo Horizonte ?
- ▶ Existem diversas proposições que não conseguem ser representadas pela lógica proposicional:
 - Qual a capital do Brasil ?

Representação de Predicados

► Exemplos:

- 2 é um número par
- As flores são perfumadas
- Rosa é uma flor
- O homem é um mamífero

► Representação Algébrica

- `par(2)`
- `perfumado(flora)`
- `flor(rosa)`
- `mamifero(homem)`

Representação de Predicados

► Exemplos:

- Todo número inteiro par é divisível por 2
- Todo coala come folhas de eucalipto
- Existem pessoas fumantes

► Representação Algébrica

- $\forall x, \text{par}(x) \rightarrow \text{divisivel}(x, 2)$
- $\forall x, \text{coala}(x) \rightarrow \text{comer}(x, \text{folha_eucalipto})$
- $\exists x, \text{pessoa}(x) \rightarrow \text{fumante}(x)$

Representação de Predicados

► Dados:

- $A(x)$ = x é um animal
- $B(x)$ = x é um mamífero
- $C(x)$ = x é um homem

► Traduz para linguagem corrente:

- $[(\forall x)(C(x) \rightarrow B(x)) \wedge (\forall x)(B(x) \rightarrow A(x))] \rightarrow [(\forall x)(C(x) \rightarrow A(x))]$

► *“Se todos os homens são mamíferos e todos mamíferos são animais então todo homem é um animal”*

Exercícios

- ▶ Traduzir para representação de predicados
 - Todos são felizes
 - Algumas pessoas são felizes
 - João não é feliz
 - Alguns animais são carnívoros
 - Nem todos os animais são carnívoros
 - Simba é um animal carnívoro
 - Todas as moscas voam
 - Não existem moscas que não voam
 - Romeu ama Julieta
 - Romeu ama Maria e não ama Julieta
 - Todos amam Romeu mas ele não ama ninguém
 - Ana é irmã de José
 - Pedro é pai de Maria e de José
 - Ana é irmã de Maria e José é irmão de Ana então José e Maria são irmãos
 - Todo animal é mortal

Definições - termo

- ▶ Qualquer variável é um termo
- ▶ Se t_1, t_2, \dots, t_n são termos e f uma função n -ária então $f(t_1, t_2, \dots, t_n)$ é um termo
- ▶ Exemplos:
 - X (variável)
 - `starwars`
 - `mortal(homem)`
 - `pai(darthvader,luke)`
 - `mestre(quigon_jinn,mestre(obiwan_kenobi,anakin_skywalker))`
- ▶ O resultado de qualquer termo é sempre um objeto

Definições - átomo

- ▶ A constante lógica *false* é um átomo
- ▶ Se t_1, t_2, \dots, t_n são termos e P uma proposição n -ária então $P(t_1, t_2, \dots, t_n)$ é um átomo
- ▶ Exemplos:
 - P (proposição)
 - *false*
 - $\sim(\text{false})$
 - $>(8, 5)$
 - $=(8, +(4, 2))$
- ▶ O resultado de qualquer átomo é sempre um valor lógico

Fórmulas

- ▶ Todo átomo é uma fórmula
- ▶ Se H é uma fórmula então $\sim(H)$ também é
- ▶ Se H e G são fórmulas então $H \sqcap G$ também é
- ▶ Se H é uma fórmula e x uma variável então $((\sqcap x)H)$ e $((\sqcup x)H)$

Ordem de Precedência

- ▶ Maior precedência: \sim
- ▶ Prec. Intermediária Superior: \square , \square
- ▶ Prec. Intermediária Inferior: \rightarrow , \leftrightarrow
- ▶ Menor precedência: \square , \square

Introdução à Lógica de Predicados

► Considere os seguintes fatos

- Marcos era um homem
 - $\text{homem}(\text{Marcos})$
- Marcos nasceu em Pompéia
 - $\text{Pompeano}(\text{Marcos})$
- Todos os que nasceram em Pompéia eram romanos
 - $\forall x: \text{Pompeano}(x) \rightarrow \text{Romano}(x)$
- César era um soberano
 - $\text{soberano}(\text{César})$
- Todos os romanos eram leais a César ou o odiavam
 - $\forall x: \text{Romano}(x) \rightarrow \text{leal-a}(x, \text{César}) \vee \text{odeia}(x, \text{César})$
- Todo mundo é leal a alguém
 - $\forall x: \exists y: \text{leal-a}(x, y)$
- As pessoas só tentam assassinar soberanos aos quais não são leais
 - $\forall x: \forall y: \text{pessoa}(x) \wedge \text{soberano}(y) \wedge \text{tenta-assassinar}(x, y) \rightarrow \sim \text{leal-a}(x, y)$
- Marcos tentou assassinar César
 - $\text{tenta-assassinar}(\text{Marcos}, \text{César})$

Introdução à Lógica de Predicados

- ▶ Responda: Marcos era leal a César ?
 - $\sim \text{leal-a}(\text{Marcos}, \text{César})$
- ▶ Adicionamos o novo predicado “Todos os homens são pessoas”
 - $\forall x: \text{homem}(x) \rightarrow \text{pessoa}(x)$

Introdução à Lógica de Predicados

- ▶ $\sim \text{leal-a}(\text{Marcos}, \text{César})$
- ▶ $\sim \text{leal-a}(x, y)$
- ▶ $\text{pessoa}(x) \wedge \text{soberano}(y) \wedge \text{tenta-assassinar}(x, y)$
- ▶ $\text{homem}(x) \wedge \text{soberano}(y) \wedge \text{tenta-assassinar}(x, y)$
- ▶ $\text{homem}(\text{Marcos}) \wedge \text{soberano}(y) \wedge \text{tenta-assassinar}(x, y)$
- ▶ $\text{homem}(\text{Marcos}) \wedge \text{soberano}(\text{César}) \wedge \text{tenta-assassinar}(x, y)$
- ▶ $\text{homem}(\text{Marcos}) \wedge \text{soberano}(\text{César}) \wedge \text{tenta-assassinar}(\text{Marcos}, \text{César})$

Predicados Computáveis

- ▶ Servem para representar domínios incontáveis
 - maior_que(X, Y)
 - menor_que($A+B, C$)

Unificação

- ▶ Substituição de variáveis em predicados estruturalmente equivalentes.
- ▶ Método: construção de uma árvore de alternativas com percurso em profundidade com retrocesso
- ▶ Critérios:
 - Mesmo nome de predicado
 - Mesmo número de argumentos
 - Mesmo valor literal (se forem conhecidos) para os argumentos (na ordem)
- ▶ Exemplos
 - homem(Joao) e homem(X) = MATCH
 - predicado(Joao,Marcos) e predicado(X) = NOT MATCH
- ▶ $P(x, x)$ e $P(y, z)$
 - substitui-se y por x (y/x)
 - obtem-se então $P(y,y)$ e $P(y,z)$
 - substitui-se z/y
 - Representação: $(z/y)(y/x)$

Unificação

- ▶ odeia(Marcos, z)
- ▶ odeia(Marcos, Pedro)
- ▶ odeia(Marcos, Joao)
- ▶ Unificando odeia(x, y) com odeia(Marcos, z):
 - (Marcos/x, Pedro/z)(Marcos/x, z/y)
 - (Marcos/x, Joao/z)(Marcos/x, z/y)
- ▶ Unificando odeia(x, y) com odeia(Marcos, Pedro):
 - (Marcos/x, Pedro/y)
- ▶ Unificando odeia(x, y) com odeia(Marcos, Joao):
 - (Marcos/x, Joao/y)

Unificação de Predicados

► Considere os seguintes fatos

- homem(Marcos)
- Pompeano(Marcos)
- $\forall x: \text{Pompeano}(x) \rightarrow \text{Romano}(x)$
- soberano(César)
- $\forall x: \text{Romano}(x) \rightarrow \text{leal-a}(x, \text{César}) \wedge \text{odeia}(x, \text{César})$
- $\forall x: \forall y: \text{leal-a}(x, y)$
- $\forall x: \forall y: \text{pessoa}(x) \wedge \text{soberano}(y) \wedge \text{tenta-assassinar}(x, y) \rightarrow \sim \text{leal-a}(x, y)$
- tenta-assassinar(Marcos, César)
- $\forall x: \text{homem}(x) \rightarrow \text{pessoa}(x)$

$\sim \text{leal-a}(x, y)$

Unificação de Predicados

- ▶ Considere os seguintes fatos
 - homem(Marcos)
 - homem(Pedro)
 - Pompeano(Marcos)
 - $\forall x: \text{Pompeano}(x) \rightarrow \text{Romano}(x)$
 - soberano(César)
 - $\forall x: \text{Romano}(x) \rightarrow \text{leal-a}(x, \text{César}) \wedge \text{odeia}(x, \text{César})$
 - $\forall x: \exists y: \text{leal-a}(x, y)$
 - $\forall x: \exists y: \text{pessoa}(x) \wedge \text{soberano}(y) \wedge \text{tenta-assassinar}(x, y) \rightarrow \sim \text{leal-a}(x, y)$
 - tenta-assassinar(Marcos, César)
 - $\forall x: \text{homem}(x) \rightarrow \text{pessoa}(x)$

Unificação de Predicados

► Considere os seguintes fatos

- homem(Marcos)
- homem(Pedro)
- Pompeano(Marcos)
- $\forall x: \text{Pompeano}(x) \rightarrow \text{Romano}(x)$
- soberano(César)
- soberano(Nero)
- $\forall x: \text{Romano}(x) \rightarrow \text{leal-a}(x, \text{César}) \wedge \text{odeia}(x, \text{César})$
- $\forall x: \forall y: \text{leal-a}(x, y)$
- $\forall x: \forall y: \text{pessoa}(x) \wedge \text{soberano}(y) \wedge \text{tenta-assassinar}(x, y) \rightarrow \sim \text{leal-a}(x, y)$
- tenta-assassinar(Marcos, César)
- $\forall x: \text{homem}(x) \rightarrow \text{pessoa}(x)$

Unificação de Predicados

► Considere os seguintes fatos

- homem(Marcos)
- homem(Pedro)
- Pompeano(Marcos)
- $\forall x: \text{Pompeano}(x) \rightarrow \text{Romano}(x)$
- soberano(César)
- soberano(Nero)
- $\forall x: \text{Romano}(x) \rightarrow \text{leal-a}(x, \text{César}) \wedge \text{odeia}(x, \text{César})$
- $\forall x: \exists y: \text{leal-a}(x, y)$
- $\forall x: \exists y: \text{pessoa}(x) \wedge \text{soberano}(y) \wedge \text{tenta-assassinar}(x, y) \rightarrow \sim \text{leal-a}(x, y)$
- tenta-assassinar(Marcos, César)
- tenta-assassinar(Pedro, Nero)
- $\forall x: \text{homem}(x) \rightarrow \text{pessoa}(x)$

Conhecimento como Regras

- ▶ Conhecimento ***declarativo*** é todo conhecimento especificado mas cujo uso não foi definido
- ▶ Conhecimento ***procedimental*** são informações de controle sobre o uso do conhecimento declarativo
- ▶ Exemplo:
 - homem(Marcos)
 - homem(César)
 - pessoa(Cleópatra)
 - $\forall x: \text{homem}(x) \rightarrow \text{pessoa}(x)$
- ▶ Pergunta: $\exists y: \text{pessoa}(y)$

Conhecimento como Regras

- ▶ Resposta: $Y = \text{Cleópatra}$
- ▶ Exemplo:
 - $\text{homem}(\text{Marcos})$
 - $\text{homem}(\text{César})$
 - $\forall x: \text{homem}(x) \rightarrow \text{pessoa}(x)$
 - $\text{pessoa}(\text{Cleópatra})$
- ▶ Resposta: $Y = \text{Marcos}$

Exercício

- ▶ Montar uma base de conhecimento para descrever a árvore genealógica da sua família:
 - Enumeração de membros por gênero
 - “homem” e “mulher”
 - Relações de parentesco direto
 - “pai de” e “mãe de”
 - “irmãos”
 - Relações de parentesco indireto
 - “avô de” e “avó de”
 - “tio de” e “tia de”
 - “primo de” e “prima de”
- ▶ Demonstrar o rastreio de uma consulta para o predicado “primo de” em função de variáveis não instanciadas

Árvore Genealógica

- ▶ homem(Carlos)
- ▶ homem(Joao)
- ▶ homem(Miguel)
- ▶ homem(Pedro)
- ▶ mulher(Maria)
- ▶ mulher(Joana)
- ▶ mulher(Carla)
- ▶ mulher(Adriana)
- ▶ pai(Carlos,Joao)
- ▶ pai(Carlos,Miguel)
- ▶ pai(Miguel,Pedro)
- ▶ mae(Maria,Joao)
- ▶ mae(Maria,Miguel)
- ▶ mae(Joana,Pedro)
- ▶ irmaos(X,Y) :- pai(Z,X) ∧ pai(Z,Y)
- ▶ irmaos(X,Y) :- mae(Z,X) ∧ mae(Z,Y)
- ▶ avo(X,Y) :- pai(X,Z) ∧ pai(Z,Y) ∧ homem(X)
- ▶ avo(X,Y) :- pai(X,Z) ∧ mae(Z,Y) ∧ homem(X)
- ▶ avó(X,Y) :- mae(X,Z) ∧ pai(Z,Y) ∧ mulher(X)
- ▶ avó(X,Y) :- mae(X,Z) ∧ mae(Z,Y) ∧ mulher(X)
- ▶ tio(X,Y) :- pai(Z,Y) ∧ irmaos(X,Z) ∧ homem(X)
- ▶ tio(X,Y) :- mae(Z,Y) ∧ irmaos(X,Z) ∧ homem(X)
- ▶ tia(X,Y) :- pai(Z,Y) ∧ irmaos(X,Z) ∧ mulher(X)
- ▶ tia(X,Y) :- mae(Z,Y) ∧ irmaos(X,Z) ∧ mulher(X)
- ▶ primo(X,Y) :- pai(Z,X) ∧ pai(W,Y) ∧ irmaos(Z,W) ∧ homem(X)
- ▶ primo(X,Y) :- pai(Z,X) ∧ mae(W,Y) ∧ irmaos(Z,W) ∧ homem(X)
- ▶ prima(X,Y) :- pai(Z,X) ∧ pai(W,Y) ∧ irmaos(Z,W) ∧ mulher(X)
- ▶ prima(X,Y) :- pai(Z,X) ∧ mae(W,Y) ∧ irmaos(Z,W) ∧ mulher(X)

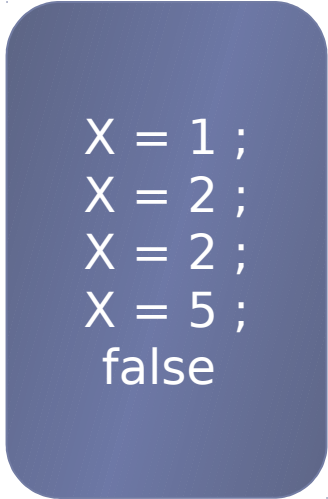
PROLOG

- ▶ Cláusulas de Horn
 - poodle(Fido)
 - $\forall x: \text{homem}(x) \rightarrow \text{pessoa}(x)$
- ▶ Em PROLOG
 - poodle(fido).
 - pessoa(X) :- homem(X).

http://www2.joinville.udesc.br/~coca/cursos/ic/material_de_prolog/

PROLOG

- ▶ Exercício: apresente todas as possibilidades (na ordem) para a consulta `a(X)`.
 - `a(X) :- b(X,Y),c(Y).`
 - `a(X) :- c(X).`
 - `b(1,2).`
 - `b(2,2).`
 - `b(3,3).`
 - `b(3,4).`
 - `c(2).`
 - `c(5).`



```
X = 1 ;  
X = 2 ;  
X = 2 ;  
X = 5 ;  
false
```

PROLOG

- ▶ Exercício: apresente todas as possibilidades (na ordem) para a consulta `a(1,X)`.
 - `a(X,Y) :- b(X,Y).`
 - `a(X,Y) :- c(X,Z), a(Z,Y).`
 - `b(1,2).`
 - `b(2,3).`
 - `c(1,2).`
 - `c(1,4).`
 - `c(2,4).`
 - `c(3,4).`

```
X = 2 ;  
X = 3 ;  
false
```

Recursividade

- ▶ `fala_sobre(A,B):- conhece(A,B).`
- ▶ `fala_sobre(P,R):- conhece(P,Q),
fala_sobre(Q,R).`
- ▶ `conhece(bill,jane).`
- ▶ `conhece(jane,pat).`
- ▶ `conhece(jane,fred).`
- ▶ `conhece(fred,bill).`

Recursividade

- ▶ Cálculo do fatorial de um número inteiro n é definido recursivamente por

$$\text{fat}(n) = n * \text{fat}(n-1), \text{ onde } \text{fat}(0) = 1$$

- ▶ Em PROLOG:
 - `fatorial(0,1).`
 - `fatorial(X,Y) :- N is X - 1, fatorial(N, F), Y is X * F.`

Recursividade

- ▶ Calcular o N-ésimo termo de uma série de Fibonacci.
- ▶ Séries de Fibonacci são dadas por:
 - $\text{fibonacci}(1) = \text{fibonacci}(2) = 1$
 - $\text{fibonacci}(N) = \text{fibonacci}(N - 2) + \text{fibonacci}(N - 1)$
- ▶ Resolução:
 - $\text{fibonacci}(0,1)$.
 - $\text{fibonacci}(1,1)$.
 - $\text{fibonacci}(N,X) :- N1 \text{ is } N-2, N2 \text{ is } N-1, \text{fibonacci}(N1,X1), \text{fibonacci}(N2,X2), X \text{ is } X1+X2$.

Recursividade

- ▶ Calcule a soma da série abaixo:
 - $X = 1 + 1/2 + 1/3 + \dots + 1/N$
- ▶ Resolução:
 - `soma(1,1).`
 - `soma(N,X) :- N1 is N-1, soma(N1,X1), X is 1/N + X1.`

Listas

- ▶ É a forma como PROLOG trata variáveis multi-valoradas (vetores).
- ▶ Exemplos:
 - []
 - [gremio, saopaulo, corinthians, flamengo, vasco, bahia]
 - [1,2,3,4,5,6,7,8,9,10]
 - [dog(fido), dog(snoopy), dog(scoobydoo)]
 - [abelha, aranha, [borboleta, mosca, barata]]

Listas

- ▶ Construção de listas em PROLOG
 - A lista [f,r,e,d] é representada por dois elementos:
[Head | Tail]
 - onde Head = f
 - e ainda Tail = [r,e,d]

Listas

► Unificações com listas

- $[b,a,d]$ e $[d,a,b]$ resulta em falha pois ordem importa
- $[X]$ e $[b,a,d]$ resulta em falha pois comprimento importa
- $[X|Y]$ e $[a,b,c,d,e,f]$ resulta em sucesso por cabeça e cauda
- $[X,Y,Z|W]$ e $[a,b,c,d,e,f]$ resulta em sucesso por cabeça e cauda
- $[X|Y]$ e $[]$ falha pois lista vazia não pode ser desconstruída
- $[X|Y]$ e $[[a,[b,c]],d]$ sucesso por cabeça e cauda
- $[X|Y]$ e $[a]$ sucesso por cabeça e cauda, com cauda vazia

Listas

- ▶ Faça um programa PROLOG para imprimir os itens de uma lista
- ▶ Resolução
 - `imprime([]):-write('.').`
 - `imprime([H|T]):-write(H), write(' '), imprime(T).`
- ▶ Faça um programa PROLOG para contar o tamanho de uma lista
- ▶ Resolução
 - `compr([],0).`
 - `compr([H|T],N) :- compr(T,N1), N is N1+1.`

Listas

► Exercícios

- Soma todos os elementos em uma lista
 - $[1,2,3,4,5] = 15$
- Determine a quantidade de zeros em uma lista binária
 - $[1,0,1,0,1] = 2$
- Imprima o vetor na ordem inversa
 - $[1,2,3,4] = 4,3,2,1$
- Some dois vetores de mesmo tamanho
 - $[1,2,3,4]$ e $[5,6,7,8] = [6,8,10,12]$

Listas

1. Soma todos os elementos em uma lista
 - `somalista([],0).`
 - `somalista([H|T], SOMA) :- somalista(T,SOMA1), SOMA is H+SOMA1.`
2. Determine a quantidade de zeros em uma lista binária
 - `contazeros([],0).`
 - `contazeros([0|T], QTDE) :- contazeros(T,QTDE1), QTDE is QTDE1+1.`
 - `contazeros([1|T], QTDE) :- contazeros(T,QTDE).`
3. Imprima o vetor na ordem inversa
 - `inverte([]).`
 - `inverte([H|T]) :- inverte(T), write(H), write(' ')`
4. Some dois vetores de mesmo tamanho
 - `somavet([], [], []).`
 - `somavet([H1|T1], [H2|T2], [HEAD|VET]) :- compr([H1|T1],TAM),compr([H2|T2],TAM),somavet(T1,T2,VET), HEAD is H1+H2.`
 - `somavet([H1|T1], [H2|T2], [HEAD|VET]) :- compr([H1|T1],TAM1),compr([H2|T2],TAM2),TAM1 \= TAM2, write('NO').`

Listas

► Exercícios

- Calcular a media dos elementos de uma lista
 - $L = [10, 8, 3]$ $M = 7$
 - `media(L, M) :- compr(L, T1), somalista(L, T2), M is T2/T1.`
- Determinar se um certo elemento X pertence a uma lista
 - $X=4$ e $L=[1,2,3,4,5]$ Yes
 - `member(H, [H | _]).`
 - `member(H, [_|T]) :- member(H,T).`
- Unir dois vetores
 - $[1,2,3,4]$ e $[5,6] = [1, 2, 3, 4, 5, 6]$
 - `uniao([], X, X).`
 - `uniao([H|Tail1], List2, [H|Tail2]) :- uniao(Tail1, List2, Tail2).`
- Determine o vetor na ordem inversa
 - $[1,2,3,4] = [4,3,2,1]$
 - `inverse([], X, X).`
 - `inverse([H|Tail], Accum, Z) :- inverse(Tail, [H|Accum], Z).`
 - `inverse2(Vet,Z) :- inverse(Vet, [], Z).`

Listas

► Exercícios

- Remover um elemento X da lista (se ele pertencer)
 - $L = [10, 8, 3]$ $X = 8$ $L2 = [10, 3]$
 - `apaga(X, [X|R], R).`
 - `apaga(X, [Y|R1], [Y|R2]) :- apaga(X, R1, R2).`
- Determinar se uma dada matriz é quadrada
 - $M = [[1,2,3],[4,5,6],[7,8,9]]$ Yes
 - `testrow([], _).`
 - `testrow([Head|Tail], Size) :- compr(Head, Size), testrow(Tail, Size).`
 - `squarematrix(Matrix) :- compr(Matrix, Size), testrow(Matrix, Size).`
- Determinar a soma de todos os elementos de uma matriz
 - $M = [[1,2,3],[4,5,6],[7,8,9]]$ $S = 45$
 - `summatrix([], 0).`
 - `summatrix([Head|Tail], Sum) :- somalista(Head, Sum1), summatrix(Tail, Sum2), Sum is Sum1+Sum2.`

Listas

► Exercícios

- Dadas duas matrizes quadradas, determinar sua soma
 - $M1 = [[1,2,3],[4,5,6],[7,8,9]]$ $M2 = [[9,8,7],[6,5,4],[3,2,1]]$
 - $R = [[10,10,10],[10,10,10],[10,10,10]]$
 - `addmatrix([], [], []).`
 - `addmatrix([Head1|Tail1], [Head2|Tail2],[Head3|Tail3]) :-`
`somavet(Head1, Head2, Head3), addmatrix(Tail1, Tail2, Tail3).`
- Determinar a soma da diagonal principal de uma matriz quadrada
 - $M = [[1,2,3],[4,5,6],[7,8,9]]$ $S = 15$
 - `selectdiagonal(0, [H|_], H).`
 - `selectdiagonal(C, [_|Tail], E) :- C1 is C-1, selectdiagonal(C1, Tail, E).`
 - `diagonal(_, [], 0).`
 - `diagonal(R, [H|T], S) :- selectdiagonal(R, H, S1), R1 is R+1,`
`diagonal(R1, T, S2), S is S1+S2.`
 - `sumdiagonal(M,S) :- diagonal(0, M, S).`

PROLOG

- ▶ Implemente um programa em PROLOG para diagnóstico de doenças com base em uma lista de sintomas = sistema especialista. Exemplo:
 - O paciente apresenta:
 - febre alta?
 - dor de cabeça?
 - dor nos olhos?
 - perda de paladar e apetite?
 - manchas na pele?
 - náuseas e vômitos?
 - tonturas?
 - extremo cansaço?
 - dores no corpo, ossos e articulações?
 - Quadro clínico: DENGUE
- ▶ Possíveis questionamentos a um SE:
 - Qual doença apresenta os sintomas X?
 - `doenca(X, [febre, dorcabeca, dorolhos, perdaapetite, manchas, vomito, tontura, cansaco, dorcorpo])`
 - Quais os sintomas da doença X?
 - `sintomas(dengue, X)`
 - Quais doenças apresentam o sintoma X?
 - `doencas(X, manchas)`

Sistema Especialista

- ▶ `bancodados([`
- ▶ `[dengue,[febre, dorcabeca, dorolhos, perdaapetite, manchas, vomito,`
`tontura, cansaco, dorcorpo]],`
- ▶ `[sarampo,[febre, dorcabeca, dorolhos, manchas, vomito, vozrouca, tosse,`
`cansaco, dorcorpo]],`
- ▶ `[variola,[febre, pustulas, vomito, tontura, cansaco, dorcorpo]],`
- ▶ `[malaria,[febre, dorcabeca, dorolhos, calafrios, sudorese, tosse, vomito,`
`tontura, cansaco, dorcorpo]],`
- ▶ `]).`
- ▶ `sintomas(Doenca, Sintomas) :- bancodados(BD), buscasintomas(Doenca,`
`BD, Sintomas).`
- ▶ `buscasintomas(_, [], []).`
- ▶ `buscasintomas(Doenca, [[Doenca, Sintomas]|_], Sintomas).`
- ▶ `buscasintomas(Doenca, [Head|Tail], Sintomas) :- buscasintomas(Doenca,`
`Tail, Sintomas).`

Resolução de Problemas

- ▶ Consiste na programação do conhecimento de um dado problema a fim de que se possa ser inferida a(s) solução(ões) do mesmo.
- ▶ Exemplo de um problema:
 - *Fui há uma festa e apresentado há três casais. Os maridos tinham profissões e esposas distintas. Após alguns goles me confundi quem era casado com quem, e suas profissões. Apenas lembro de alguns fatos, então me ajude descobrir quem são estes casais, com base nos seguintes dados:*
 - 1. O médico é casado com a Maria;
 - 2. O Paulo é advogado;
 - 3. Patrícia não é casada com Paulo;
 - 4. Carlos não é médico.
 - 5. Lucia era esposa de alguém
 - 6. Um dos homens era engenheiro
 - 7. Um dos homens se chamava Luis

(Retirado da revista Coquetel: Problemas de Lógica)

Resolução de Problemas

► Resolução:

- Descobrir três tuplas-3 da forma (Homem, Esposa, Profissão), ou seja, a solução pode ser representada por
 - solucao = ((H1, E1, P1), (H2, E2, P2), (H3, E3, P3))
- Escreva um programa PROLOG que represente as restrições descritas no enunciado do problema de forma a inferir possíveis soluções que satisfaçam a todas as restrições (CLP = *constraints logic programming*)

Resolução de Problemas

- ▶ solucao([(carlos, E1, P1),(luis, E2, P2),(paulo, E3, advogado)]) :-
 - esposa(E1), esposa(E2), esposa(E3),
 - alldifferent([E1, E2, E3]),
 - profissao(P1), profissao(P2),
 - alldifferent([P1, P2]),
 - ((E1==maria, P1==medico);(E2==maria, P2==medico)),
 - E3\==patricia,
 - P1\==medico.
- ▶ esposa(maria).
- ▶ esposa(patricia).
- ▶ esposa(lucia).
- ▶ profissao(medico).
- ▶ profissao(engenheiro).
- ▶ alldifferent([]).
- ▶ alldifferent([H|T]) :- not(member(H, T)), alldifferent(T).