# PROLOG
# PROgramming in LOGic

# Prolog

- Linguagem de programação lógica e declarativa
- Foi criado em 1970 por R. Kowalski e Maarten van Emden em Edimburgo e Alain Colmerauer em Marsailles

*algoritmo* = *lógica* + *controle*

- Junto com Lisp, são as linguagens de programação simbólica mais usadas em Inteligência Artificial

# Prolog

- Origem: pesquisas em provadores automáticos de teoremas e sistemas de dedução dos anos 60 e 70
- Faz inferência a partir de um processo de refutação (resolução).
  - Cláusulas de Horn com encadeamento para trás
- Usuários: *algumas centenas de milhares*

# Sintaxe: *símbolos*

- Variáveis são representadas por ***letras maiúsculas*** (A, B, …) ou "_" (chamada de variável anônima)
- Símbolos funcionais (funtores) e símbolos relacionais (predicados) são representados por ***letras minúsculas***. Exemplos:
  - p(X, Y, _) : aridade 3 (p/3)
  - q : aridade 0, i.e., constante

# Sintaxe: *termos*

- Termos:
  - Uma variável é um termo
  - Se f é um símbolo funcional n-ário e t1,…, tn são termos, então f(t1,…, tn) é um termo
- Átomo: se p é uma relação n-ária e t1,…, tn são termos então p(t1,…,tn) é um átomo
- Uma pergunta (**consulta**) é um conjunto finito de átomos.

# Sintaxe: *cláusula*

- São escritas na forma de uma implicação invertida:

  **H :- B**

  em que **H** é uma conclusão (átomo ou literal positivo chamado de cabeça), **B** é um conjunto de premissas (corpo), i.e., é uma conjunção de átomos e **:-** é a implicação clássica invertida **(←)**

  **H :-** (*cláusula unária ou fato*)

  **:- B** (*cláusula objetivo ou consulta*)
- Se uma variável aparecer só em B , ela é existencialmente quantificada
- Se uma variável aparecer em H, ela é universalmente quantificada

# Sintaxe

- Conjunção: ‘,’
- Disjunção: ‘;’
- Toda cláusula deve terminar com um ponto
- Elementos não lógicos:
  - negação: \+ ou not()
  - cut:  !
  - assert, retract, etc...

# Definição de programas

- Programas Prolog puros: conjuntos de cláusulas sem elementos extra-lógicos
- Programa prolog generalizado: conjunto de cláusulas com negação

# Utilizando o SWI Prolog

1 Editar o programa

2 Inicializar o SWI

- Linux: Digitar **pl** e carregar o programa, para isso digite **consult(nome_do_programa.pl)**

# Utilizando o Prolog

- O Prolog basicamente faz inferência sobre os fatos expressos no programa
- O sistema avalia uma *consulta* com relação ao programa lido e responde
  - **yes**: se existirem valores para as variáveis da consulta que a satisfaça
  - **no**: caso a consulta não seja (mais) satisfeita

  Digitando ';' o sistema busca uma nova solução e halt. termina o programa.

# Exemplos

- Família 1
  - construção, consultas, busca, unificação
- Família 2
  - regras, falha, cut
- Concatenação de listas
  - recursão, árvore de provas

# Listas

- Lista em Prolog: L= [H|T], onde H é um elemento da lista L (*cabeça*) e T é o resto da lista (*cauda*)
- Abreviações:
  - [ ] : lista vazia
  - [s0|[s1,…,sn|t]] : [s0,s1,…,sn|t]
  - [s0,s1,…,sn|t] : [s0,s1,…,sn,t]

  Usualmente distinguimos variáveis que recebem listas colocando um ‘s’ após o simbolo (ex. Xs)

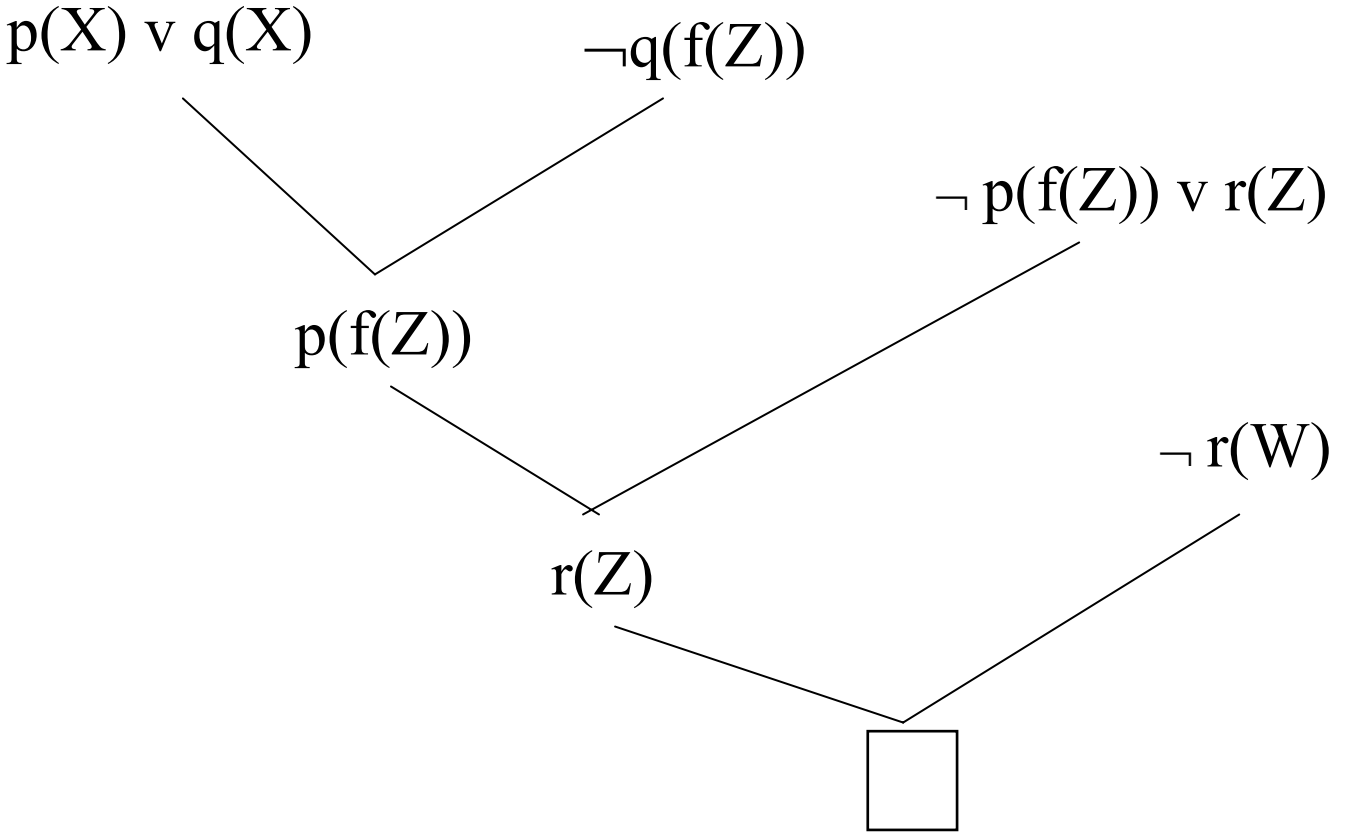
# O Processo Computacional

- Uma cláusula: H :- $A_1$, …, $A_n$ deve ser interpretada como “para que H seja verdade, todo $A_i$ deve ser verdade”
- Para provar $A_i$ é necessário encontrar um conjunto de valores para as variáveis de $A_i$ que satisfaça $A_i$. Este conjunto de valores é associado às variáveis por meio de substituições (unificação de termos)

# Computação como Resolução

- O que nos interessa é mostrar que uma determinada fórmula lógica é conseqüência de um conjunto de fórmulas, isso é feito por um processo de refutação
  - se queremos provar $\alpha$ a partir de um "programa lógico" P devemos considerar a união da **negação** de $\alpha$ com P e derivar uma **contradição**, isto é, mostrar que o conjunto P U $\neg \alpha$ é insatisfatível

# Árvore de prova por refutação



# SWI Prolog

- http://www.swi-prolog.org/
- Prolog prompt

?- _

- consulta Prolog (Prolog *goal*)

?- governador(serra).

yes