Universidade Federal de São Carlos
Campus Sorocaba

Uso/Representação de conhecimento

Sistemas Baseados em Conhecimento (SBC)

Prolog

Profa: Katti Faceli
katti@ufscar.br

Baseado nos slides de Ana Carolina Lorena – UFABC
e no livro Sistemas Inteligentes: Fundamentos e Aplicações

08/01/2012

Agentes com conhecimento

- Podem representar o mundo
- Usar um processo de inferência para derivar novas representações sobre o mundo
- Utilizar as novas representações para deduzir o que fazer
- Um agente com conhecimento precisa:
  - Representar o conhecimento
  - Raciocinar sobre o conhecimento

Agentes com conhecimento

- Os agentes que usam busca, por exemplo, contem conhecimento bastante específico e inflexível
  - Sabe as mudanças de estado, o custo do caminho para ir do estado inicial até algum outro estado, sabe estimar o custo de ir do estado atual até um objetivo...

Agentes com conhecimento

- Podemos representar o conhecimento de maneiras mais gerais, que permitam combinar informações para atender a diferentes propósitos – útil para lidar com ambientes parcialmente observáveis
- Ex: médico fazendo um diagnóstico deve deduzir qual é o estado (doença) – não diretamente observável – antes de escolher o tratamento
  - Conhecimento:
    - Regras aprendidas de livros e professores
    - Padrões de associação que o médico estabeleceu ao longo da carreira

Agentes com conhecimento

- Um agente baseado em conhecimento deve ter:
  - Uma **base de conhecimento** (BC): conjunto de **sentenças** que representam afirmações sobre o mundo (abstração do mundo)
    - Sentenças são expressas em uma linguagem de representação de conhecimento (descrição de conceitos)
  - Um **motor de inferência**: Mecanismos para adicionar novas sentenças e consultar a BC
    - Genericamente: TELL (informe) e ASK (pergunte)
    - Podem envolver **inferência** (derivar novas sentenças a partir das sentenças existentes na BC)

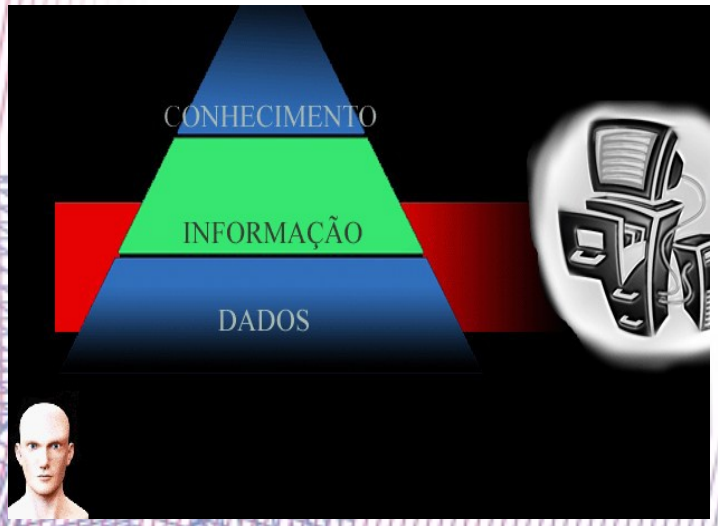
Agentes com conhecimento

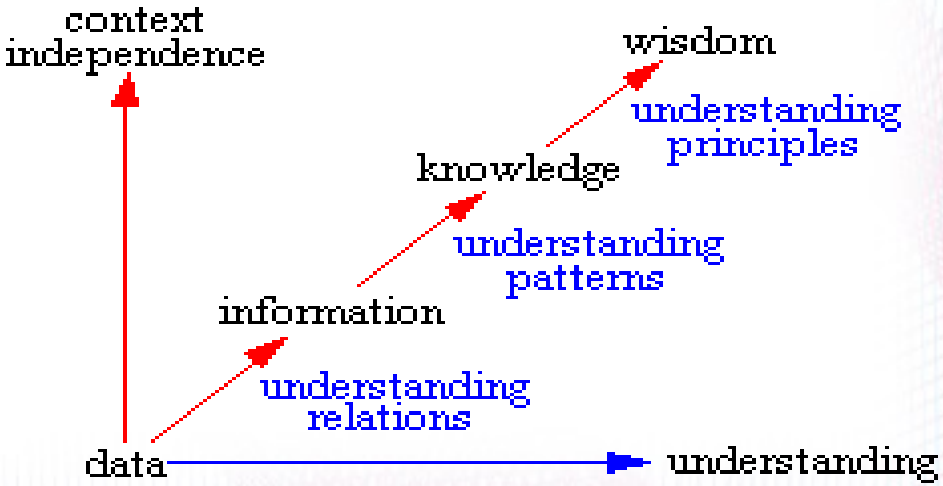
- O agente baseado em conhecimento trabalha no nível de conhecimento e não de implementação:
  - Precisamos especificar o que o agente sabe e quais são suas metas e não como ele funciona
  - Ex: motorista de taxi automatico
    - Meta: levar passageiro até Marin
    - Conhecimento: está em San Francisco, a ponte Golden Gates é a única ligação entre os dois locais
    - Não importam os detalhes de implementação: p. Ex: se o conhecimento geográfico é armazenado como uma lista ligada ou um mapa de pixels

Conhecimento

- Mas o que é conhecimento?

•**Dado**: elemento puro, quantificável sobre um determinado evento
  - Conjunto de descrições simples
  - Por si só não oferece embasamento para entendimento da situação
•**Informação**: dado analisado e contextualizado
•**Conhecimento**: modelo das relações existentes entre as informações contidas nos dados e do seu significado





http://www.systems-thinking.org/dikw/dikw.htm
http://www.systems-thinking.org/kmgmt/kmgmt.htm

Conhecimento

- **Dado**: capital de R$100, taxa de juros de 5%
- **Informação**: capital de R$100, taxa de juros de 5% em uma conta de poupança
  - R$100 é a quantidade de dinheiro na conta e 5% é o fator usado para calcular o juro que o capital vai render
- **Conhecimento**: se eu colocar R$100 na poupança e o banco paga 5% de juros anualmente, no fim do ano receberei R$5 de juros e meu capital será de R$105. Quanto mais dinheiro eu colocar na poupança, maior será meu rendimento.

Conhecimento

- O processo de gerar conhecimento resulta de:
  - Comparar informações e combiná-las em ligações úteis e com significado
    - conhecimento é dependente de nossos valores e experiências
    - conhecimento é sujeito às leis universalmente aceitas

Conhecimento

- Tipos de conhecimento:
  - **Declarativo** (conhecimento descritivo e genérico sobre fatos e eventos) - “o que é”
  - **Procedural** (conhecimento prescritivo difícil de expressar e explicar) - “como funciona”
  - **De senso comum** (Composto de conhecimento declarativo e procedural) - “o julgamento do certo e do errado”
  - **Heurístico** (Único para cada indivíduo, não pode ser obtido em nenhuma fonte, envolve avaliação sistemática e uso de regras heurísticas)

Procedimento heurístico: método de aproximação de soluções dos problemas, que não segue um percurso claro mas se baseia na intuição e nas circunstâncias a fim de gerar conhecimento novo. Oposto do procedimento algoritmico.

Sistemas convencionais x SBC

S. Convencionais	SBCs
Estruturas de Dados	Representação do Conhecimento
Dados e Relações entre Dados	Conceitos e Relações entre Conceitos
Algoritmos Não Heurísticos	Algoritmos Heurísticos
Conhecimento Embutido no Código	Conhecimento Representado Explicitamente e Separado do Código que o Manipula
Explicação do Raciocínio é Difícil	Podem e Devem explicar o Raciocínio

Sistemas convencionais x SBC

- Um SBC é basicamente um programa de computador que usa conhecimento representado explicitamente para resolver problemas

Base de conhecimento

- Contém a descrição do conhecimento necessário para resolver o problema de que o agente trata
- Uma BC pode ser composta por até dezenas de milhares de sentenças
  - Relações de causa e efeito
    - ”se temperatura > 37,5 então paciente tem febre”
  - Metaconhecimento - conhecimento sobre como guiar a busca da solução
    - ”se paciente é alcolatra, procure primeiro doenças hepáticas”,
    - ”procure a solução primeiro por caminhos com poucas alternativas”

Base de conhecimento

- Nem sempre o conhecimento em uma BC é completamente consistente e preciso (pode gerar soluções conflitantes)
- Nem sempre esse conhecimento é completo

Representação de Conhecimento (RC)

- Forma sistemática de estruturar e codificar o que se sabe sobre uma determinada aplicação
- Características desejáveis:
  - Ser compreensível ao ser humano (caso seja necessário avaliar o conhecimento do sistema)
  - Não depender de detalhes sobre o processador de conhecimento que a interpretará
  - Ser robusta, permitindo uso mesmo que não aborde todas as situações possíveis
  - Ser generalizável, para poder ser utilizada em diversas situações e interpretações

Representação de Conhecimento (RC)

- Representação deve ser:
  - Coerente com o senso comum
  - Suficientemente precisa
    - Permitir que um programa apresente um comportamento interessante tomando como base essa representação
    - Permitir raciocínio para obtenção de novos conhecimentos

RC

- Alguns formalismos de representação de conhecimento:
  - Lógica (Simbólica): Proposicional e Predicados
  - Regras de Produção
  - Redes Semânticas
  - Frames
  - Sistemas Fuzzy
  - ...

BC usando lógica

- Uma BC representada usando lógica é uma conjunção de fórmulas
- Prolog é uma linguagem que permite a representação de conhecimento e o raciocínio sobre esse conhecimento, com base na LPO

Engenharia de conhecimento

- Estabelece diretrizes para a construção de uma base de conhecimento

Engenharia de conhecimento

- Etapas do processo de engenharia de conhecimento:
  - **Identificar a tarefa**: delinear as questões que a BC permitirá e os tipos de fatos que estarão disponíveis – determinar que conhecimento é necessário representar para conseguir respostas para as várias instâncias do problema (parecido com a determinação do ambiente de tarefas)
  - **Agregar conhecimento relevante** (aquisição de conhecimento): pode ser necessário obter conhecimento de um especialista do domínio – deve-se compreender bem como o domínio realmente funciona

Engenharia de conhecimento

- Etapas do processo de engenharia de conhecimento:
  - **Definir um vocabulário de predicados, funções, constantes**: converter os conceitos do domínio na notação da lógica
    - determinar os tipos de itens que existem, não suas propriedades específicas e inter-relacionamentos
  - **Codificar o conhecimento geral sobre o domínio**: escrever os **axiomas** correspondentes a todos os símbolos do vocabulário – fixa o significado dos símbolos

Engenharia de conhecimento

- Etapas do processo de engenharia de conhecimento:
  - **Codificar descrição de uma instância específica do problema**: escrever sentenças atômicas simples sobre instâncias de conceitos presentes na ontologia (dados vindos das percepções, por exemplo)
  - **Formular consultas ao mecanismo de inferência e obter respostas**
  - **Depurar a base de conhecimento**: tenta identificar axiomas ausentes, conhecimento incorreto, etc com base em respostas inesperadas para consultas realizadas

Prolog

- Material
  - Learn Prolog Now
    - http://www.learnprolognow.org/
  - Manual de referencia do SWI-Prolog (no moodle e no site do SWI-Prolog)
  - Apostila ”Linguagem Prolog” de Sandra Cortinovi (no moodle)
  - Introdução à Programação PROLOG, Luiz A. M. Palazzo

Programação em Lógica

- Programação em Lógica (Logic Programming):
  - uso de lógica como linguagem de programação
- A Programação Lógica é um paradigma de programação baseado no raciocínio:
  - especifica o que se sabe sobre um problema
  - o foco está mais direcionado a conhecimento e menos direcionado a algoritmos
- Nesse paradigma, programar é fornecer:
  - axiomas, que indicam alguns fatos sobre o mundo
  - regras para derivação de outros fatos (inferência)

Programação em Lógica

- Programação Lógica (PL) é declarativa, em oposição à programação imperativa (procedural) ou à programação orientada a objetos
- Um programa em linguagem de programação procedural como C especifica detalhadamente o fluxo de execução do programa (seu algoritmo – **como** o problema é resolvido)
- Em PL especifica-se **o que** deve ser computado ao invés de **como** deve ser computado

Programação em Lógica

- Na programação em lógica a noção de computação está relacionada com a noção de dedução
- não há estruturas de controle tais como:
  - desvio condicional
  - estruturas de repetição
  - etc

Programação em Lógica

- Uma das principais ideias da PL é a de que um algoritmo é constituído por dois elementos:
  - Lógica: corresponde à definição do que deve ser solucionado
  - Controle: estabelece como a solução deve ser obtida
- A tarefa do programador em PL é **especificar o componente lógico** do algoritmo
- O controle da execução é exercido por um sistema de inferência ligado à linguagem lógica
  - não é o programador que estabelece como a inferência é feita

Programação em Lógica

- Um programa em PL é a representação de determinado problema ou situação expressa através de um conjunto finito de cláusulas

Programação em Lógica

- O conhecimento (programas e/ou dados) é expresso por meio de:
  - **Fatos**: cláusulas que denotam uma verdade incondicional
    - Ex: (1) João é uma criança. (2) Chocolate é doce.
  - **Regras**: cláusulas que definem condições que devem ser satisfeitas para que uma certa declaração seja considerada verdadeira
    - Ex: Crianças gostam de alimentos desde que eles sejam doces
- O processamento (controle) é realizado mediante **consultas** à BC, por meio do mecanismo de inferência
  - Ex: João gosta de chocolate?

PROLOG

- PROLOG (PROgramming in LOGic): primeira e mais conhecida das linguagens para programação em lógica
- Criada por Alain Colmerauer, na década de 70
- Base fundamental: Lógica de Primeira Ordem
  - Mas permite representar apenas **cláusulas de Horn**
- Proposta para resolver problemas envolvendo objetos e relações entre objetos
- Prolog é um provador automático de teoremas

Cláusulas de Horn

- **Fatos**: cláusulas com um único literal positivo:
  - p(X, Y).
    - $\forall X , \forall Y \; p(X, Y)$
- **Regras**: cláusulas com mais literais, sendo um positivo:
  - q(X) :- p(Y), r(Z).
    - $\forall X \; \forall Y \; \forall Z \;\; \neg \, p(Y) \vee \neg \, r(Z) \;\; \vee q(X)$
    - $\forall X \; \forall Y \; \forall Z \; p(Y) \wedge r(Z) \; \rightarrow \; q(X)$
- **Consultas** ou **Restrições**: são cláusulas sem nenhum literal positivo
  - p(X), q(Y), r(Z).
    - $\exists X \; \exists Y \; \exists Z \; \neg p(X) \vee \neg q(Y) \vee \neg r(Z)$

PROLOG

- Prolog é um nome geral para uma família de sistemas:
  - Prolog de Edimburgo
  - SWI-Prolog
  - IC-Prolog
  - microProlog
  - Quintus-Prolog
  - Arity Prolog
- Prolog geralmente é interpretada, mas há casos em que é compilada

PROLOG

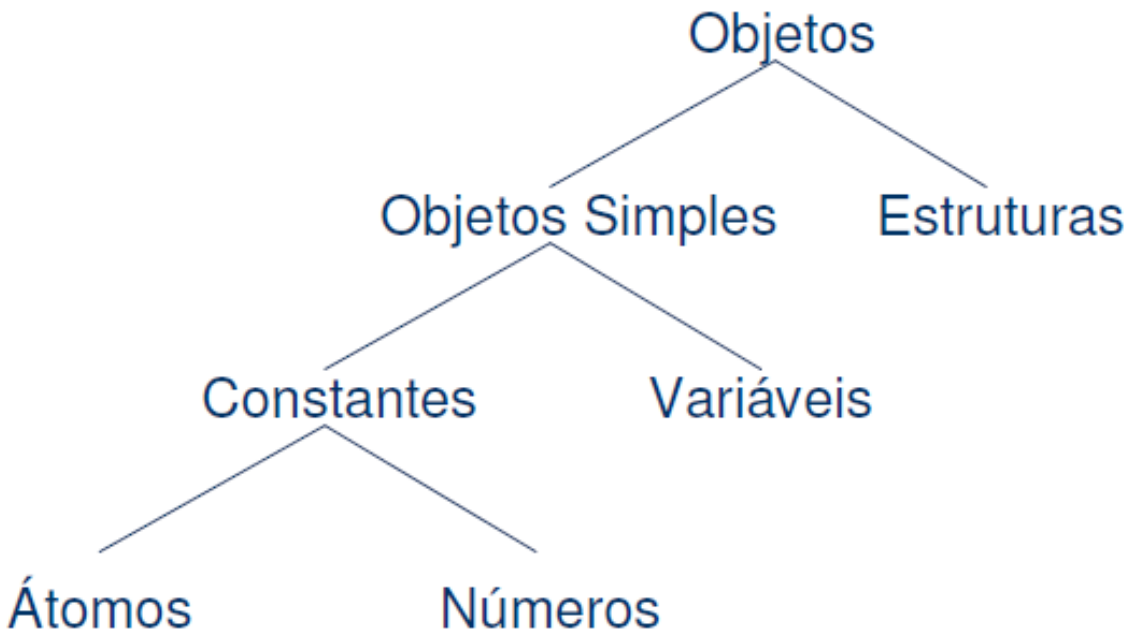
- Vamos usar o **SWI-Prolog**
  - http://www.swi-prolog.org/
- SWI-Prolog é um interpretador Prolog desenvolvido no departamento de Ciências Sociais e Informática da Universidade de Amsterdã
- Software livre, distribuído sob licença GPL
- Tem versões para windows e linux

PROLOG

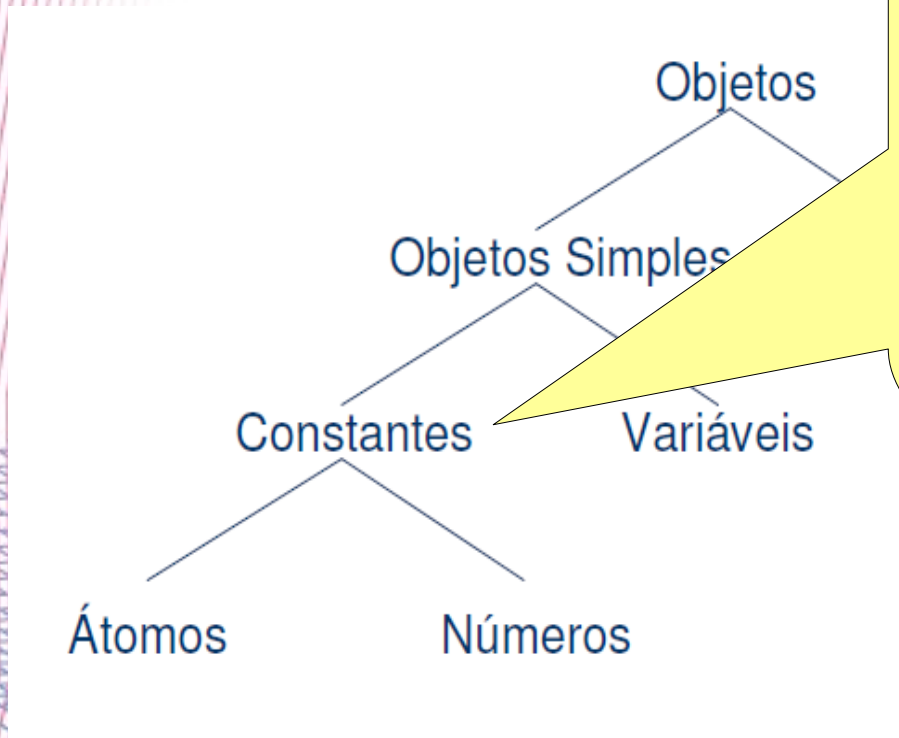
- Para rodar:
  - No windows: menu Iniciar
  - No linux:
    - Abrir console
    - Digitar swipl, pl ou swi-prolog
- Usando o help:
  - help.
  - help(consult).

PROLOG - termos

- Em prolog, os termos são classificados em:



PROLOG - termos



Constantes

Átomos:

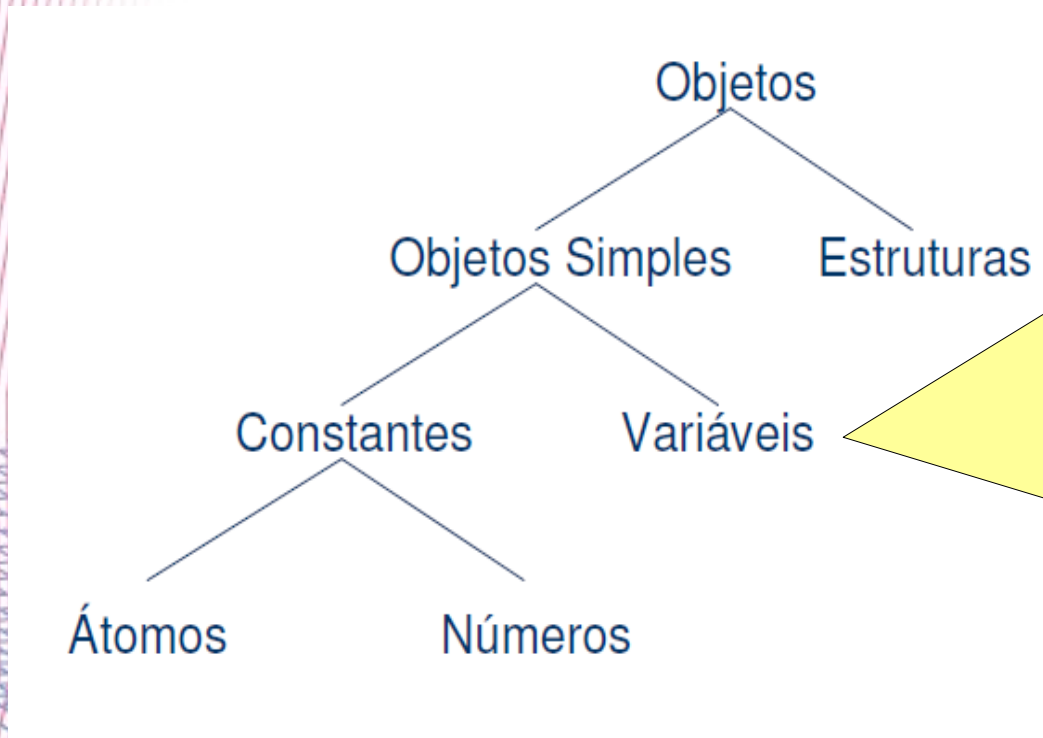
- cadeias compostas por letras, dígitos e caracteres especiais
- sempre começam com letra minúscula ou são uma string qualquer delimitada por apóstrofos

Ex: curso_de_IA, maria, x_y, abc, alguma_Coisa, ‘ABC’, ‘123’

Números: incluem os inteiros e reais

Ex: 1, -21, 11024, 1.22, -0.395

PROLOG - termos



Variáveis: cadeias de letras, dígitos ou _, que sempre iniciam com letra maiúscula ou com _

Ex: Pessoa, _pessoa, X, Y2a, _abc, Maria

O escopo de uma variável é dentro de uma mesma regra ou dentro de uma pergunta

Se a variável X ocorre em duas regras/perguntas, então são duas variáveis distintas

Mas a ocorrência de X dentro de uma mesma regra/pergunta significa a mesma variável

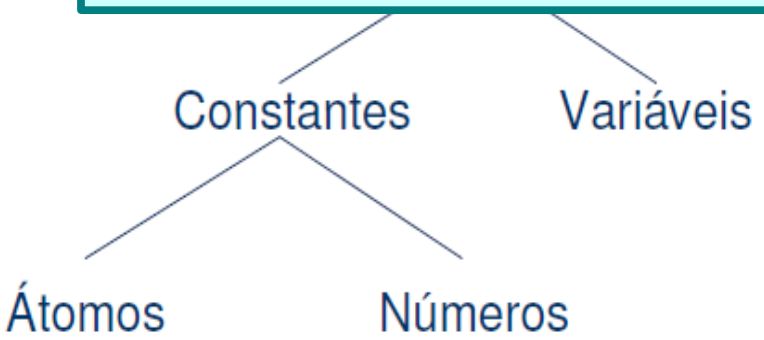
PROLOG - termos

Uma variável pode estar:

– Instanciada: quando a variável já referencia ou está **unificada** a algum objeto

– Livre ou não-instanciada: quando a variável não referencia (não está unificada a) um objeto

– Uma vez instanciada, somente Prolog pode torná-la não-instanciada através de seu mecanismo de inferência (não o programador)



de letras, dígitos ou
m com letra

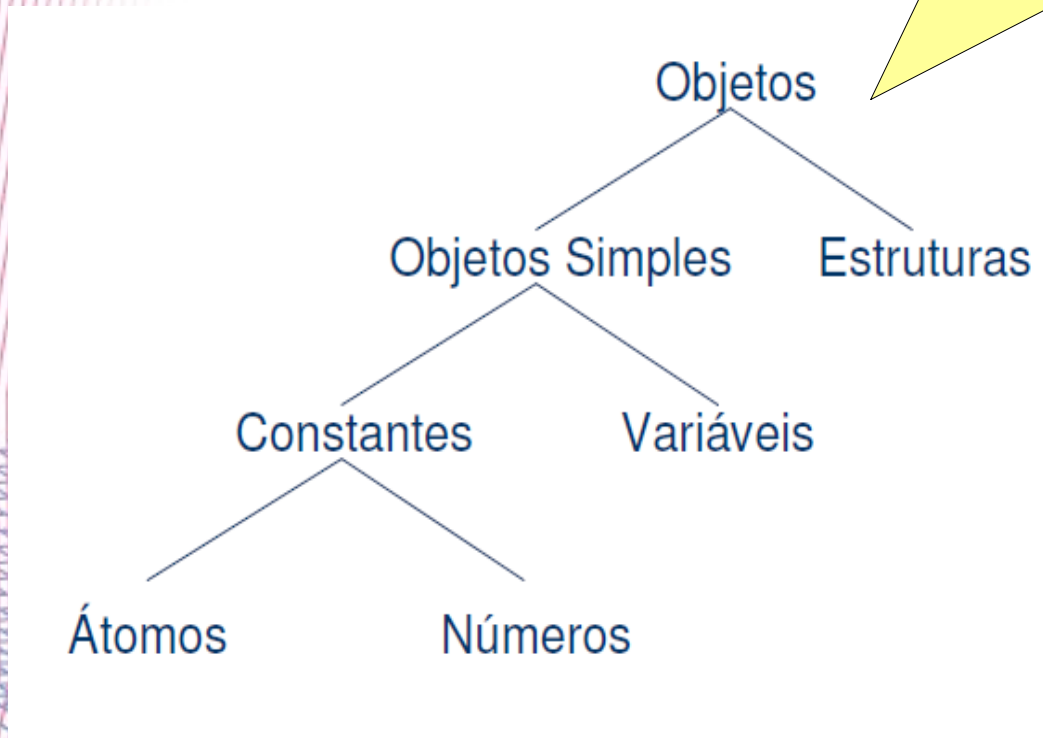
essoa, X, Y2a, _abc,

ariável é dentro de
u dentro de uma

ocorre em duas regras/perguntas, então são duas variáveis distintas

Mas a ocorrência de X dentro de uma mesma regra/pergunta significa a mesma variável

PROLOG - termos



Objetos: composição de termos (**simples** ou **estruturas**) – feita por meio de um funtor (o que chamavamos de função)

Funtor: definido por um nome (símbolo funcional, com mesma sintaxe dos átomos) e por uma aridade (nº de argumentos)

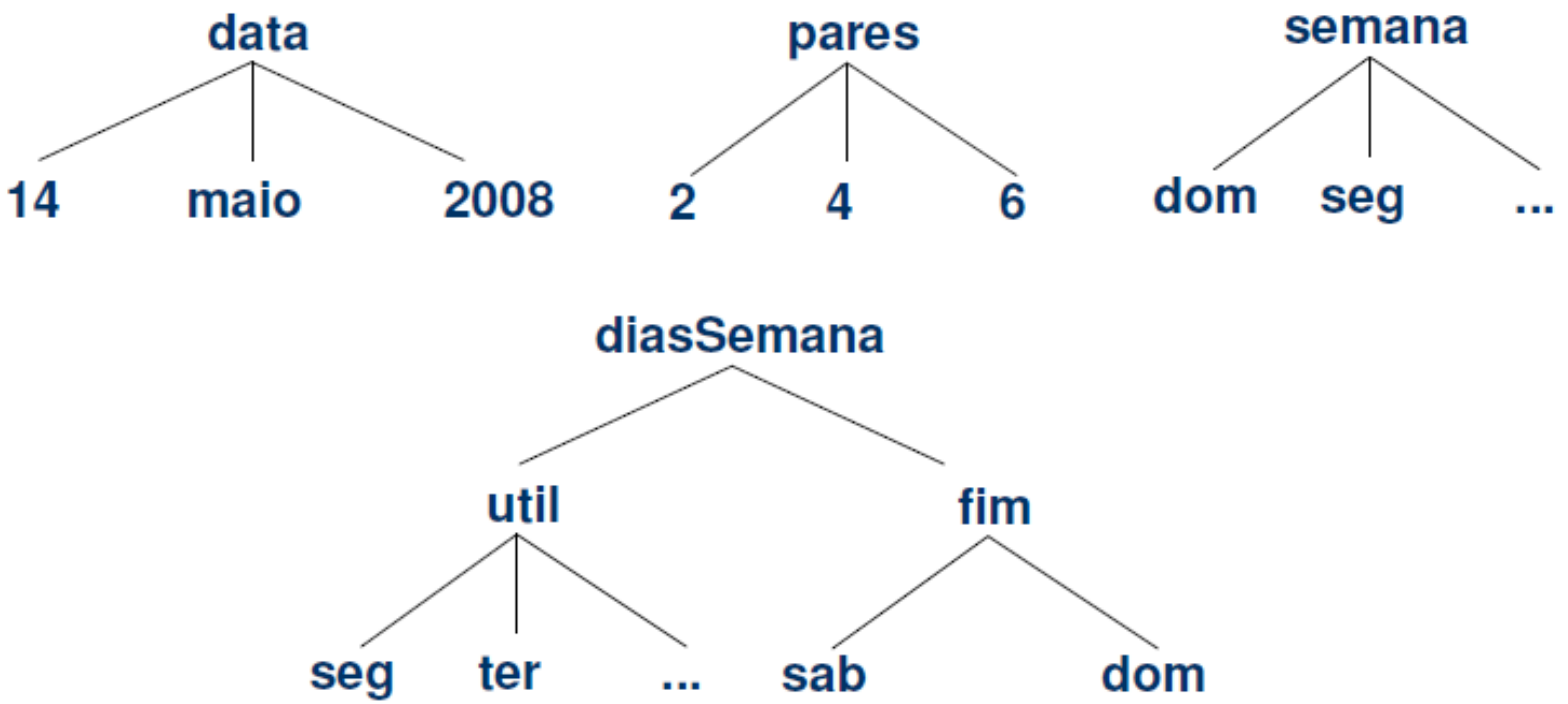
Ex:

```
data(14,maio,2008)
pares(2,4,6)
semana(dom, seg, ter, qua, qui, sex,
            sab)
diasSemana(util(seg, ter, qua, qui,
                sex), fim(sab, dom))
```

Funtor/aridade: data/3, pares/3, semana/7, diasSemana/2,
util/5, fim/2

PROLOG - termos

- Todo objeto estruturado pode ser representado como uma árvore, onde a raiz da árvore é o funtor e os filhos da raiz são os componentes

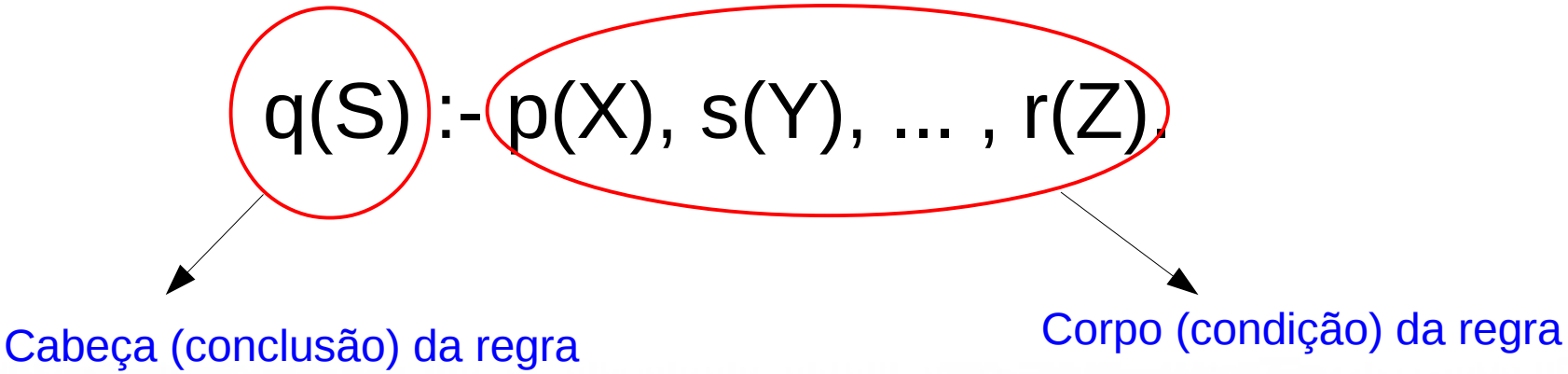


PROLOG - predicados

- Representam propriedades ou relações entre objetos – sintaxe é a mesma dos funtores
  - pai_de(joao,pedro) = “João é pai de Pedro.”
  - bonita(maria) = “Maria é bonita.”
  - gosta_de(ana,vinho) = “Ana gosta de vinho.”
  - ordem(1,3,4,8) = “1, 3, 4 e 8 estão numa relação de ordem.”
- Composição de predicados com base na LPO:
  - Conjunções: representadas por “,”
  - Disjunções: representadas por “;”
  - Implicações lógicas: “:-”

PROLOG - predicados

- A regra $p(X) \wedge s(Y) \wedge ... \wedge r(Z) \rightarrow q(S)$ seria representada em prolog como:
  - q(S) :- p(X), s(Y), ... , r(Z).
- Obs: toda cláusula termina sempre com .



PROLOG - predicados

- CUIDADO: Predicados têm a MESMA SINTAXE das estruturas, porém a SEMÂNTICA é DISTINTA
  - Estrutura: DADO
  - Predicado: AFIRMAÇÃO sobre dados
  - Predicados têm valor-verdade associado e estruturas não têm
  - Ex:
    - pai_de(joao,pedro): é verdade que João é pai de Pedro.
    - gosta_de(ana,vinho): é verdade que Ana gosta de vinho.
    - data(22,outubro,2002)
    - pares(2,4,6)

Exemplo

```
grande(urso).
grande(elefante).
pequeno(gato).
marrom(urso).
preto(gato).
cinza(elefante).
escuro(Z) :-
    preto(Z).
escuro(Z) :-
    marrom(Z).
```

Fatos

Regras

Lembrar que na LPO escrevemos
$\forall Z$ preto(Z) $\rightarrow$ escuro(Z)

Exercício:
- digitar o programa acima no SWI-Prolog
  - criar um arquivo
  - digitar o texto
  - salvar com a extensão .pl
- No prompt ?- faça as seguintes consultas:
  - preto(gato).
  - preto(urso).
  - escuro(X). (depois da resposta digite ;)
  - escuro(X), grande(X). (depois da resposta digite ;)
  - elabore e faça outras consultas

- Mas como as variáveis são instanciadas?

Unificação de termos

- Unificação (matching): é a operação mais importante entre termos
- Processo que pega como entrada dois termos e checa se eles unificam (casam)
  - Se os termos não unificam, o processo falha
  - Se eles unificam, a unificação foi bem sucedida e as variáveis em ambos os termos são instanciadas, caso existam
- Existe um operador de unificação em Prolog: ‘=‘

Unificação de termos

- Dois termos S e T unificam se:
  - Eles são idênticos ou
  - As variáveis em ambos os termos podem ser instanciadas a objetos de maneira que após a substituição das variáveis por esses objetos os termos se tornam idênticos

Unificação de termos

- Se S e T são constantes, então S e T unificam somente se são idênticos
  - a unifica com a
  - 4 unifica com 4
  - a não unifica com 4
  - 1 não unifica com 0 (em prolog não podemos ter mais de um símbolo constante representando um mesmo objeto)

Unificação de termos

- Se S for uma variável livre e T for qualquer termo, então S e T unificam e S é instanciado para T
  - X unifica com joão
  - Y unifica com 8
  - A unifica com ponto(1,2)
  - Z unifica com triangulo(ponto(1,1), A, ponto(2,3))

Unificação de termos

- Se S e T são estruturas, elas unificam se
  - S e T têm o mesmo funtor e aridade e
  - Cada um dos termos correspondentes (argumentos) unificam (a ordem dos termos na estrutura deve ser preservada)
    - pai(joao,maria) unifica com pai(joao,maria)
    - pai(X,Y) unifica com pai(joao,maria) e instancia: X/joao e Y/maria
    - pai(maria,joao) não unifica com pai(joao,maria)
    - dia(10,maio,04) não unifica com data(10,maio,04)

Unificação de termos

- Mais exemplos

Termo 1	Termo 2	Resultado da Unificação
henrique	henrique	unificam
eduardo	henrique	não unificam
X	par(a,b)	unificam (X = par(a,b))
2.35	Var	unificam (Var=2.35)
data(2,outubro,1999)	date(2,outubro,1999)	não unificam
pai_de(X,eduardo)	pai_de(henrique,Y)	unificam (X= henrique e Y= eduardo)
pai_de(X,maria)	pai_de(eduardo,X)	não unificam
pai_de(X,maria)	pai_de(X,filhos(eduardo , maria))	não unificam
familia(P,M,F)	familia(joao,pedro)	não unificam

Exercício

- Para as tentativas de unificação a seguir:
  - Indique se os termos unificam
  - Digite-os no prompt do SWI-Prolog, anote a resposta, e explique o porquê do que foi obtido

1) pedro = Ana.
2) X = teste(Z,W,J).
3) 2 = Variavel.
4) g(f(X,g(Z))) = Y.
5) data(X,X,2004) = data(12,04,2004).
6) g(f(X)) = f(g(X)).
7) familia(joao,maria,pedro) = familia(A,B).
8) gosta(ana, Julio) = gosta(x, Ana).
9) X = estuda(ana, Y), Y = livro(prolog), X = Y.