

INTELIGÊNCIA ARTIFICIAL

6. LÓGICAS DA IA

Adriana Postal
André Luiz Brun

Antes de iniciarmos....

- Ler material sobre “Lógica Proposicional”:
 - É uma revisão.
 - Em Material de Aula/Material de Apoio/Lógicas para IA e Prolog:
 - /Revisão Lógica Proposicional: 3 textos com o assunto
 - Livro de IA baseada em lógica
 - Livro do Luger: formalismo para o cálculo de predicados
 - Outros materiais utilizados durante as aulas

6.1. Introdução

- A lógica está relacionada com raciocínio e com a validade de argumentos.
- Geralmente, não estamos preocupados com a veracidade das sentenças, mas sim com a validade.
- Exemplo:

Todos os limões são azuis

Mary é um limão

Então, Mary é azul

PREMISSAS

CONCLUSÃO

6.1. Introdução

- Por que validade e veracidade podem ser separadas desta forma?
 - Um raciocínio é considerado como válido se sua conclusão for verdadeira, nos casos em que suas premissas também sejam verdadeiras.

“

Um raciocínio é válido se ele conduzir a uma conclusão verdadeira em todas as situações nas quais as premissas sejam verdadeiras”

(COPPIN, 2013)

Algumas deficiências (AMADOR; PIMENTÃO, 1988)

- Não tem conceitos.
- Apresenta grande dificuldade na legibilidade e expressão do conhecimento por pessoas não familiarizadas com ela.
- Incapaz de lidar com incertezas: só funciona sobre V e F, não sobre possibilidades.

Então, por que usar?

- Ao recorrer à dedução matemática, a lógica é capaz de derivar novos conhecimentos a partir de outros já existentes.
- Ao contrário das outras representações vistas, permite raciocinar facilmente sobre negativas e disjunções (OU).

Então, por que usar?

- A lógica pode ser utilizada como um método representacional para comunicar conceitos e teorias.
- É utilizada para representar linguagem em sistemas que sejam capazes de compreender e analisar a linguagem humana.

6.2. Lógica proposicional

- Material disponível para leitura!

Sistema lógico

- É definido em termos:
 - Sintaxe: o alfabeto de símbolos e como eles podem ser combinados.
 - Semântica: o que o símbolo significa.
 - Regras de dedução: permite derivar uma expressão a partir de um conjunto de outras expressões e, desse modo, fazer argumentações e provas.

Sistema lógico

- Um sistema lógico tem um conjunto de verdades fundamentais → axiomas.
- Axiomas: regras básicas que são conhecidas como verdadeiras e a partir das quais todos os outros teoremas do sistema podem ser provados.
- Características de um sistema lógico (COPPIN, 2013):

Características

- a) Correção: um sistema lógico é descrito como sendo correto se todo teorema for logicamente válido ou uma tautologia.
- b) Completude: um sistema lógico é completo se toda tautologia for um teorema.

Características

- c) Decidibilidade: um sistema lógico é decidível se for possível produzir um algoritmo que determinará se uma “fórmula bem formada” (fbf) é um teorema.
- d) Monotonicidade: um sistema lógico é descrito como sendo monotônico se uma prova válida no sistema não pode ser tornada inválida pelo acréscimo de premissas adicionais ou hipóteses.

Resumo características de um Sistema Lógico

Sistema Lógico	Lóg. Proposicional	Lóg. de Predicados
Correto	✓	✓
Completo	✓	✓
Decidível	✓	✗
Monotônico	✓	✓

As linguagens da lógica

- A linguagem que expressa a lógica proposicional é chamada de cálculo proposicional.
- Para a lógica de predicados, é o cálculo de predicados.

6.3. Lógica dos predicados

- Lógica de 1^a ordem; cálculo dos predicados.
- Predicados: são declarações a respeito de objetos em si ou sobre as relações dos objetos entre si.
- Apresenta muitas limitações, mas permite fazer raciocínios com facilidade.

6.3. Lógica dos predicados

- Principal diferença entre as lógicas proposicional e de predicados: compromisso ontológico:
 - Lógica proposicional: pressupõe que existem fatos que são válidos ou não-válidos no mundo.
 - Lógica de predicados: pressupõe que o mundo consiste em objetos, com certas relações entre eles que são válidas ou não-válidas.
- Extensão da lógica proposicional, introduzindo novos conceitos:

a) Termos

- Constituem expressões que descrevem ou nomeiam entidades ou objetos.
- Tipos de termos:
 - Variáveis: em “ x é poeta romântico”, “ x ” é uma variável e pode ser substituída por qualquer constante durante a avaliação de um predicado.

a) Termos

- Constantes: em “Colombo descobriu a América”, “Colombo” é uma constante.
- Funções: em “idade(x)”, “idade” é uma função que retorna um valor inteiro resultante de um cálculo específico.

b) Predicados

- Constituem nomes de propriedades e relações.
- A articulação de um predicado com um termo formará a unidade significativa mínima.
- Exemplo: “z é senador”. “senador” é um predicado o qual, quando avaliado, resulta em um valor-verdade.
- Pode-se escrever: senador(z)

b) Predicados

- Aridade de predicados: número de termos que o predicado tem:
 - chove → aridade 0
 - senador(x) → aridade 1
 - pai(x, Alberto) → aridade 2

b) Predicados

- Instanciação: é o processo de avaliação de um predicado, que consiste em substituir variáveis por constantes para a obtenção de enunciados:
 - $\text{pai}(x, y)$
 - Se $x = \text{João}$ e $y = \text{José}$
 $\text{pai}(\text{João}, \text{José})$

c) Quantificadores

- São 2 quantificadores usados nos enunciados do cálculo de predicados:
 - Universal (\forall): “para todo...”
 - Existencial (\exists): “existe um...”
- Quantificação: aplicar quantificadores ao processo de geração de enunciados a partir de funções enunciativas.

Um exemplo

- Para exemplificar como seria a representação de informação utilizando a lógica dos predicados, considere as declarações (RABUSKE, 1995):

Exemplo

1. Calabar foi enforcado
2. Getúlio foi presidente
3. Todo traidor é enforcado
4. Todos os índios eram selvagens
5. Tiradentes não era índio
6. Tiradentes foi considerado traidor

Exemplo

- Sendo:
 - \forall : quantificador universal
 - \exists : quantificador existencial
 - \neg : negação
- As declarações seriam assim representadas na lógica dos predicados:

EXEMPLO

- | | | |
|---------------------------------------|---|---|
| 1. Calabar foi enforcado | → | 1. enforcado(Calabar) |
| 2. Getúlio foi presidente | → | 2. presidente(Getúlio) |
| 3. Todo traidor é enforcado | → | 3. $\forall x \text{ traidor}(x) \rightarrow \text{enforcado}(x)$ |
| 4. Todos os índios eram selvagens | → | 4. $\forall x \text{ índio}(x) \rightarrow \text{selvagem}(x)$ |
| 5. Tiradentes não era índio | → | 5. $\neg \text{índio}(\text{Tiradentes})$ |
| 6. Tiradentes foi considerado traidor | → | 6. $\text{traidor}(\text{Tiradentes})$ |

Um exemplo

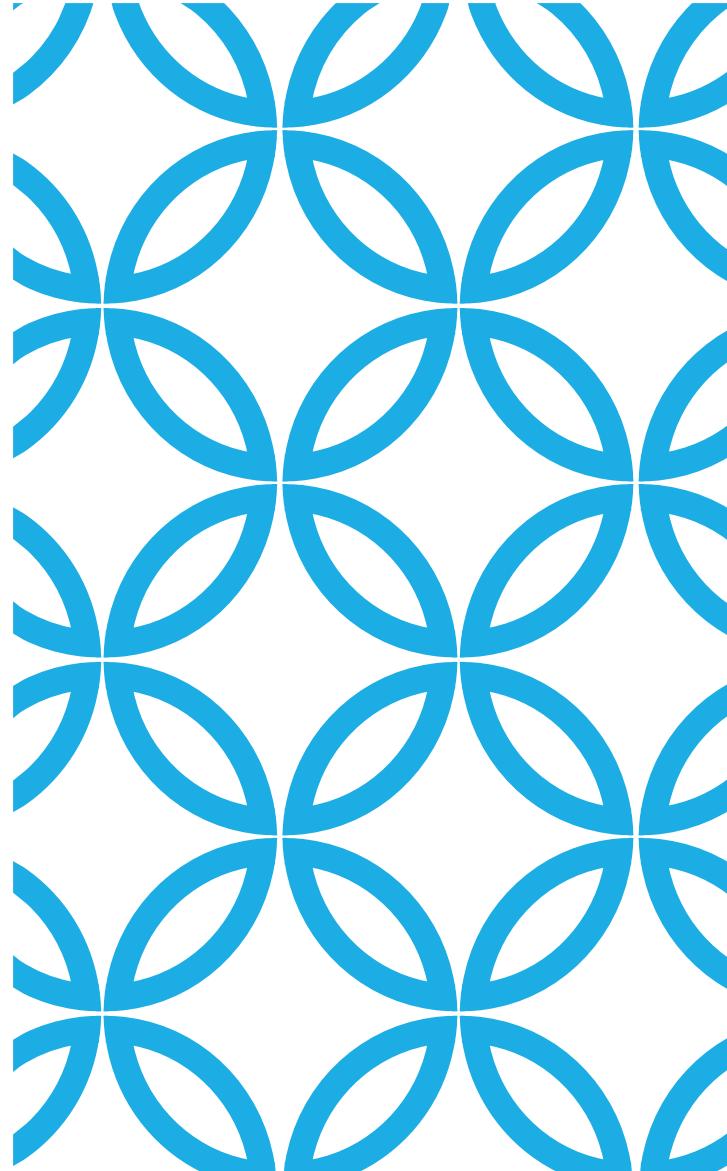
- A representação com predicados ocasionou perda de informação: tempos de ocorrência dos fatos.
- Podemos ver que as declarações não são facilmente trabalháveis.
- A matemática trabalha com expressões lógicas através da resolução, mas elas devem estar em forma de cláusulas.

Um exemplo

- A cláusula deve estar na Forma Normal Conjuntiva (FNC).
- Passos para transformar expressões em FNC (RABUSKE, 1995):
 - Arquivo com os 7 passos no Teams.
- Para nosso exemplo, após executar os 7 passos, teremos um conjunto de cláusulas:

RESUMO EXEMPLO

Declarações originais	Declarações em Predicados	Declarações em FNC
Calabar foi enforcado	enforcado(Calabar)	enforcado(Cabalar)
Getúlio foi presidente	presidente(Getúlio)	presidente(Getúlio)
Todo traidor é enforcado	$\forall x \text{ traidor}(x) \rightarrow \text{enforcado}(x)$	$\neg \text{ traidor}(x) \vee \text{enforcado}(x)$
Todos os índios eram selvagens	$\forall x \text{ índio}(x) \rightarrow \text{selvagem}(x)$	$\neg \text{ índio}(y) \vee \text{selvagem}(y)$
Tiradentes não era índio	$\neg \text{ índio}(Tiradentes)$	$\neg \text{ índio}(Tiradentes)$
Tiradentes foi considerado traidor	traidor (Tiradentes)	traidor(Tiradentes)



6.4. VANTAGENS E DESVANTAGENS

Vantagens

- Oferece um ferramental exaustivamente testado, capaz de fazer inferências interessantes sobre o conhecimento
- Natural para quem entende
- Flexibilidade no que se refere aos campos de aplicação

Vantagens

- Precisão e modularidade
- PROLOG representa muito bem a lógica de predicados: linguagem declarativa, onde é realçado “o que” e não “como” fazer.

Desvantagem

- A lógica separa a representação e o processamento, tornando difícil incluir aspectos heurísticos.

6.5. PROLOG

- PROLOG é uma linguagem declarativa, em que cada linha corresponde a uma afirmação.
- A variável na afirmação deve ser entendida como universalmente quantificada. Exemplo:

$\text{pai_de}(X, Y) \rightarrow \forall X \forall Y \text{ pai_de}(X, Y)$

6.5. PROLOG

- A negação em PROLOG é por falha finita:
“da consulta à base de conhecimento não é possível afirmar que ...”
- PROLOG é programação em lógica (PALAZZO, 1997):
 - Tem raízes no cálculo de predicado
 - 1ª implementação: Alain Colmerauer e sua equipe, na Universidade de Aix-Marseille, em 1972

6.5. PROLOG

- A formalização semântica da programação com cláusulas de Horn é devida a Kowalsky, em 1974
- A especificação do 1º *standard* (PROLOG de Edimburgo) foi realizada por Warren e Pereira, em 1977.
- O PROLOG trabalha com estruturas de dados de alto nível:
 - Listas
 - Árvores
 - Grafos

PRINCIPAIS DIFERENÇAS

Programas Convencionais

- Processamento numérico
- Soluções algorítmicas
- Estruturas de controle e conhecimento integradas
- Difícil modificação
- Somente respostas totalmente corretas
- Somente a melhor solução possível

Programas em Lógica

- Processamento simbólico
- Soluções heurísticas
- Estruturas de controle e conhecimento separadas
- Fácil modificação
- Incluem respostas parcialmente corretas
- Incluem todas as soluções possíveis

Principais propriedades

- Funciona simultaneamente como linguagem de programação e de especificação
- Possui capacidade dedutiva
- Opera de forma não-determinística
- Permite a representação de relações reversíveis
- Permite interpretação declarativa, procedural e operacional
- São naturalmente recursivos.

Principais aplicações

- Sistemas Baseados em Conhecimento (SBCs)
- Sistemas de Base de Dados (BDs)
- Sistemas Especialistas (SEs)
- Processamento da Linguagem Natural (PLN)
- Educação
- Modelagem de Arquiteturas não-convencionais.

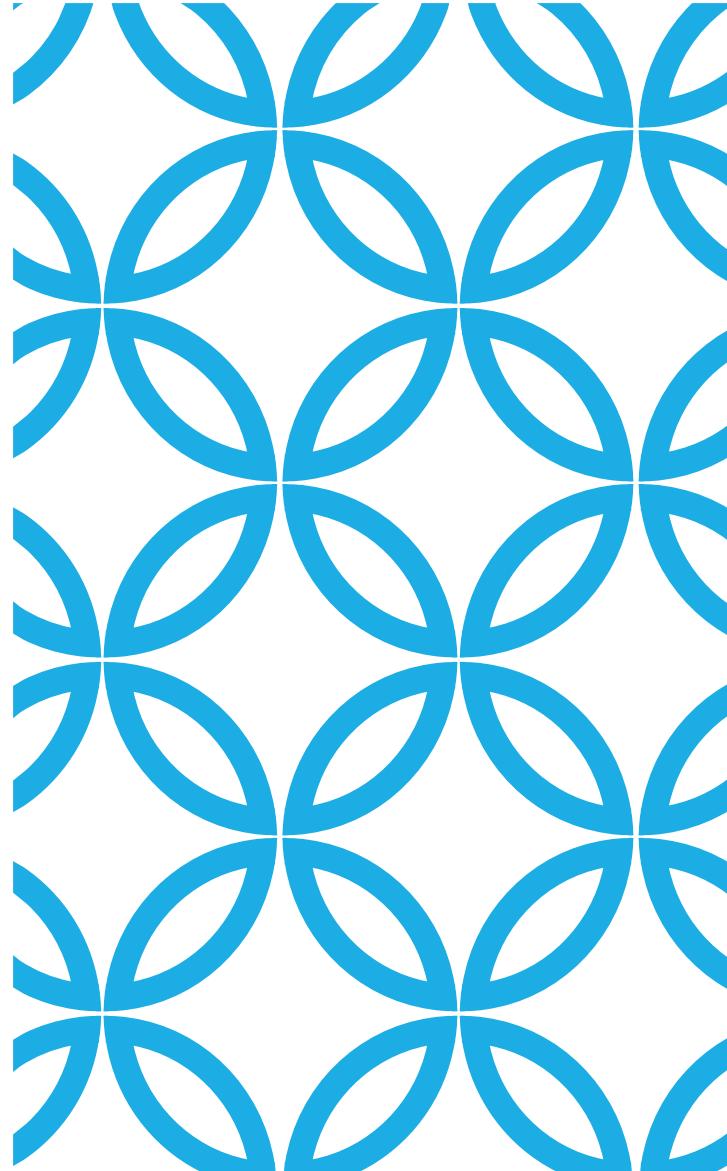
6.5. PROLOG

- A programação em PROLOG consiste em:
 - Estabelecer relações entre objetos
 - Formular consultas sobre tais relações
- Um programa PROLOG é formado por cláusulas, que podem ser de 3 tipos:
 - Fatos ou assertivas
 - Regras ou procedimentos
 - Consultas

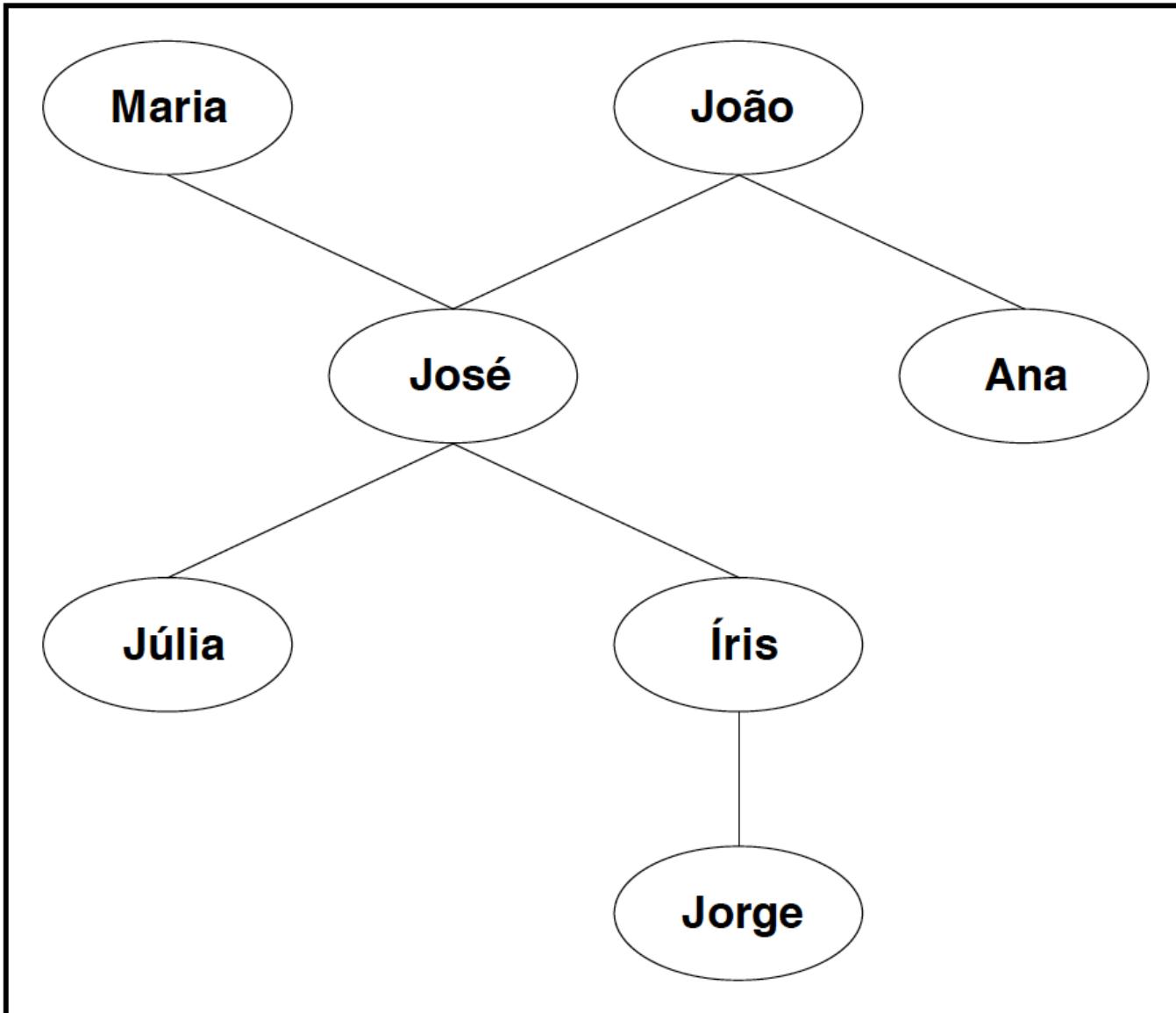
Software para PROLOG

Web: <https://swish.swi-prolog.org>

Vocês também podem instalar na máquina.



EXEMPLO: ÁRVORE GENEALÓGICA



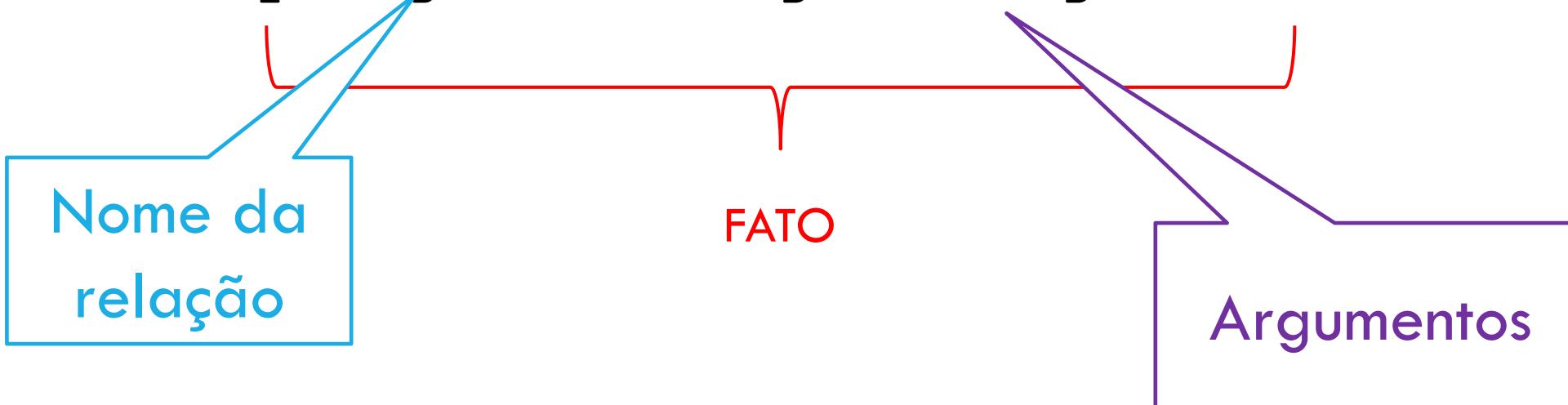
Exemplo

- João é um dos progenitores de José:
progenitor(joão, josé) .

Exemplo

- João é um dos progenitores de José:

progenitor (joão, José).



Exemplo

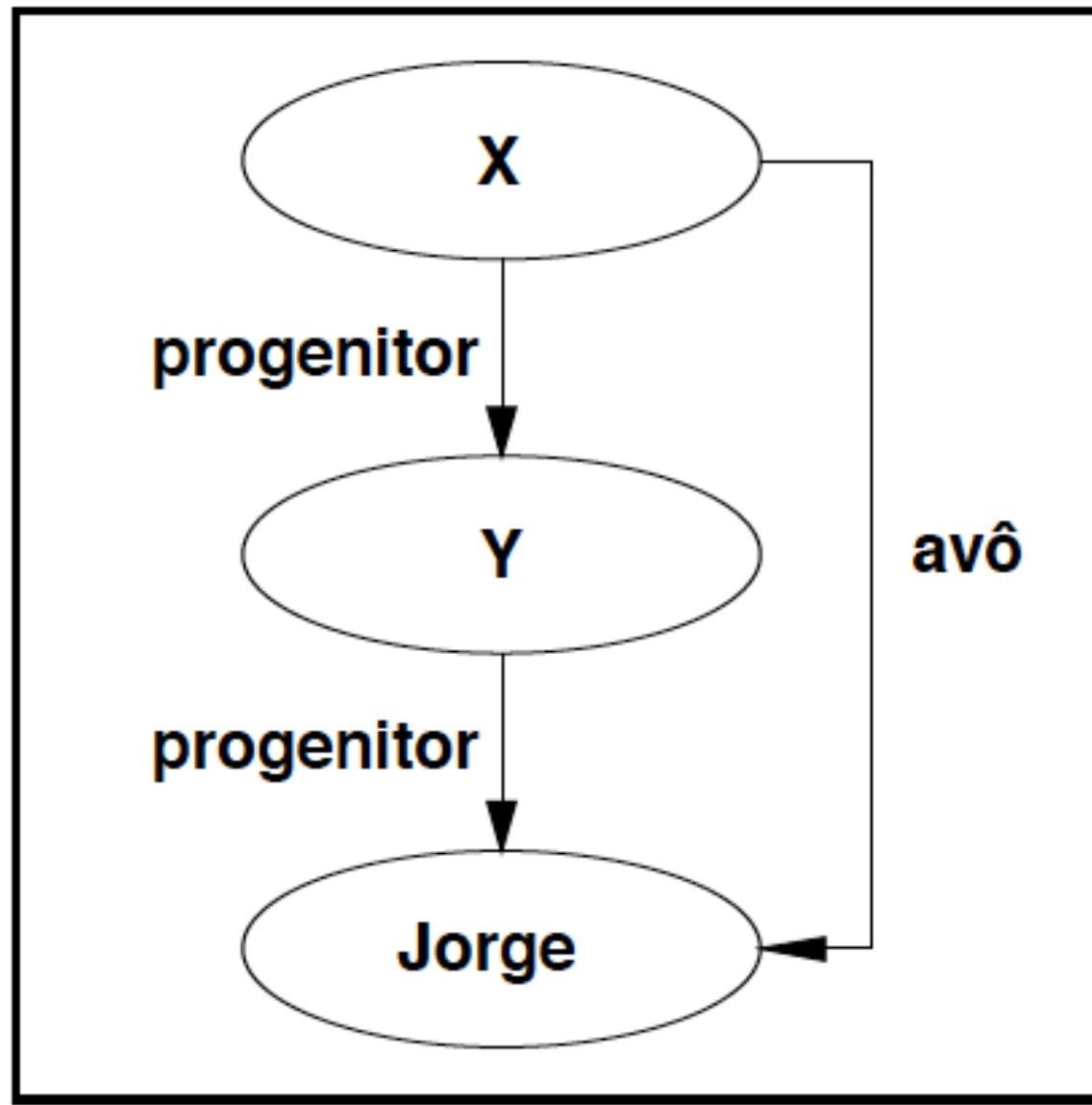
- **Fatos que representam a árvore do exemplo:**
progenitor(joão, josé).
progenitor(maria, josé).
progenitor(joão, ana).
progenitor(josé, júlia).
progenitor(josé, íris).
progenitor(íris, jorge).

Exemplo

- Consultas: interrogar um programa acerca de suas relações
- Resposta do PROLOG: conjunto de objetos que satisfazem as condições originalmente estabelecidas pela consulta

Exemplo

- Perguntas possíveis com nossa base:
 - José é o progenitor de Íris?
 - Ana é um dos progenitores de Jorge?
 - Quem é o progenitor de Íris?
 - Quem são os filhos de José?
 - Quem é progenitor de quem?
 - Quem são os avós de Jorge?



Exemplo

1. Quem é progenitor de Jorge? (por ex.: Y)
2. Quem é progenitor de Y? (por ex.: X)

progenitor(X, Y) , progenitor(Y, jorge) .



AMPLIANDO A BASE



Exemplo

- Adicionar fatos novos: o sexo das pessoas.

masculino(joão) .

masculino(josé) .

masculino(jorge) .

feminino(maria) .

feminino(júlia) .

feminino(ana) .

feminino(íris) .

Exemplo

- Adicionar novas relações:

- Filho?

filho(josé, maria) .

filho(josé, joão) .

...

- Algo mais elegante?

Exemplo

- Declaração lógica para a relação filho:

Para todo X e Y

Y é filho de X se

X é progenitor de Y

- Em PROLOG:

filho(X, Y) :- progenitor(X, Y).

SE

REGRA

6.5. PROLOG

- Regras estabelecem condições para a satisfação das relações:
 - Pode ser verdadeiro se alguma condições forem satisfeitas.
- As regras têm 2 partes:

conclusão :- condição

se

cabeça

corpo

Exemplo – novas relações

- **Mãe:**

Para todo X e Y

X é mãe de Y se

X é progenitor de Y e

X é do sexo feminino

Exemplo – novas relações

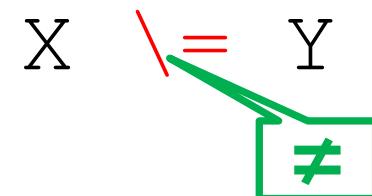
- **Irmã:**

Para todo X e Y

X é irmã de Y se

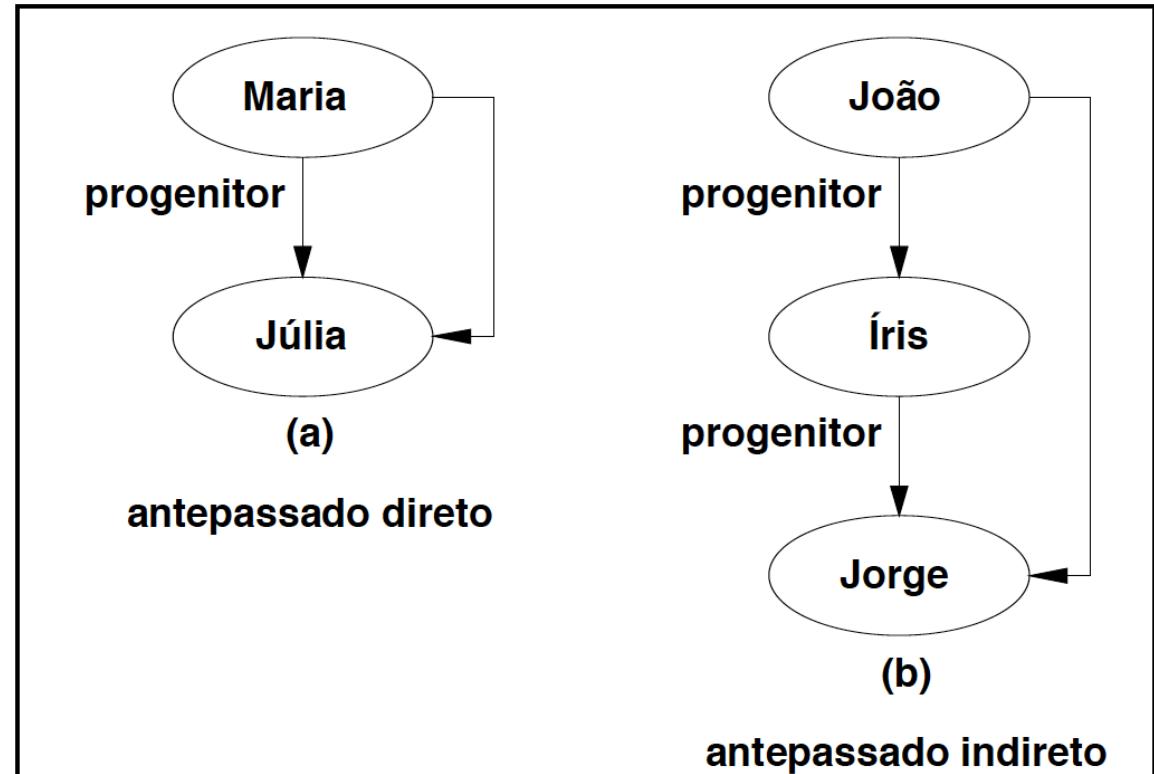
X e Y possuem um progenitor
comum e

X é do sexo feminino



Exemplo

- É possível utilizar construções recursivas
- Suponha a relação antepassado:



Exemplo

- Para antepassado direto, a regra é simples:

antepassado(X, Z) :- progenitor(X, Z).

- E o antepassado indireto?

```
antepassado(X, Z) :-  
    progenitor(X, Y),  
    progenitor(Y, Z).  
antepassado(X, Z) :-  
    progenitor(X, Y1),  
    progenitor(Y1, Y2),  
    progenitor(Y2, Z).  
antepassado(X, Z) :-  
    progenitor(X, Y1),  
    progenitor(Y1, Y2),  
    progenitor(Y2, Y3),  
    progenitor(Y3, Z). . . . etc.
```

Exemplo

- **Forma elegante: usar recursão**

Para todo X e Z

X é antepassado de Z se

existe um Y tal que

X é progenitor de Y e

Y é antepassado de Z

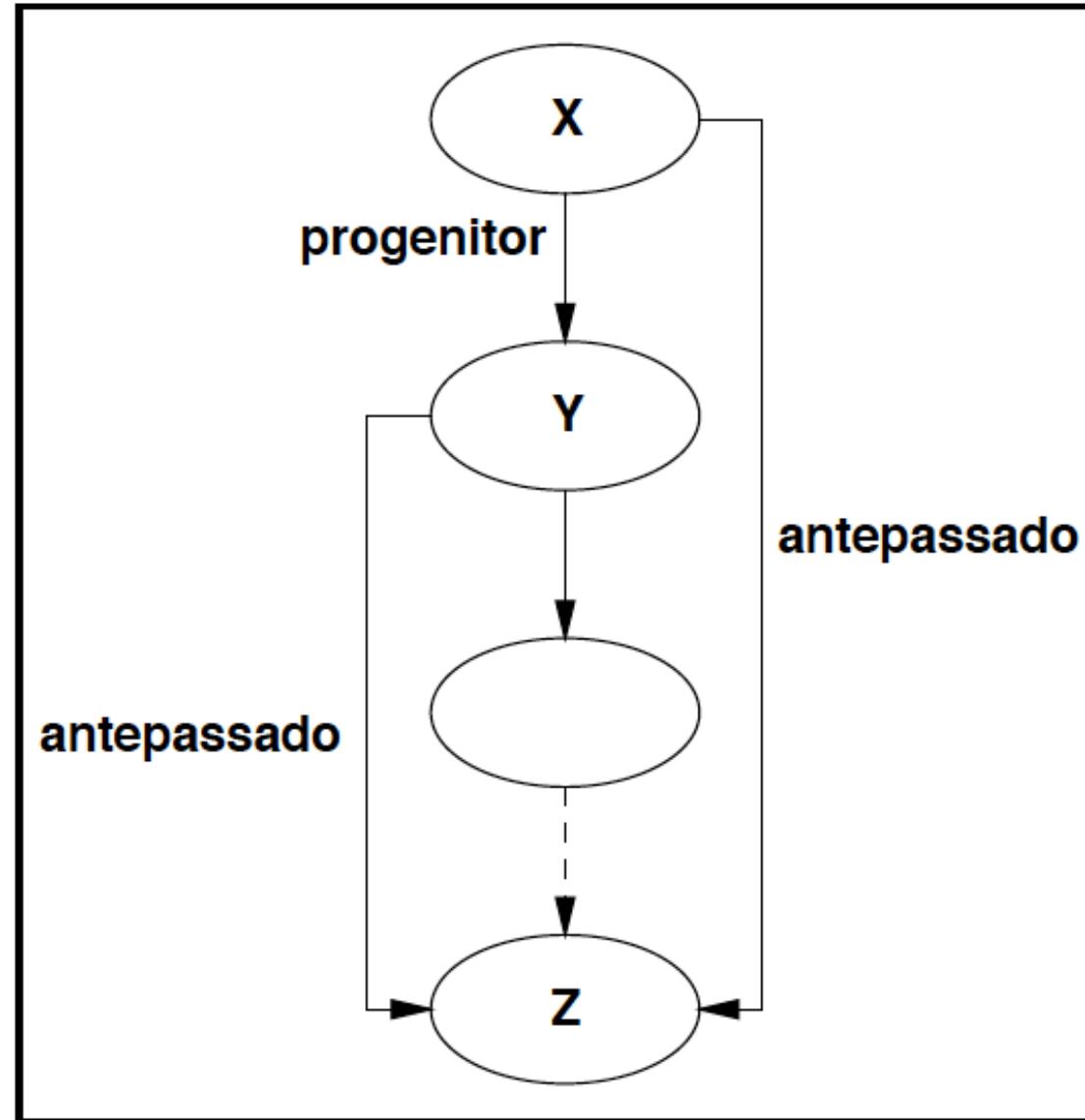
Exemplo

- Em PROLOG:

```
antepassado(X, Z) :-  
    progenitor(X, Y),  
    antepassado(Y, Z).
```

- Então, temos 2 regras:

- A 1^a, para antepassado direto
- A 2^a, para antepassado indireto



6.5. PROLOG

- Estabelecer se um objeto satisfaz uma consulta requer vários passos, entre eles:
 - Inferência lógica
 - Exploração de caminhos em uma árvore de busca ou pesquisa:
 - *Backtracking*.

Noções básicas de PROLOG

- Abrir documento no Teams (NocoesProlog.pdf).
- Vamos treinar!

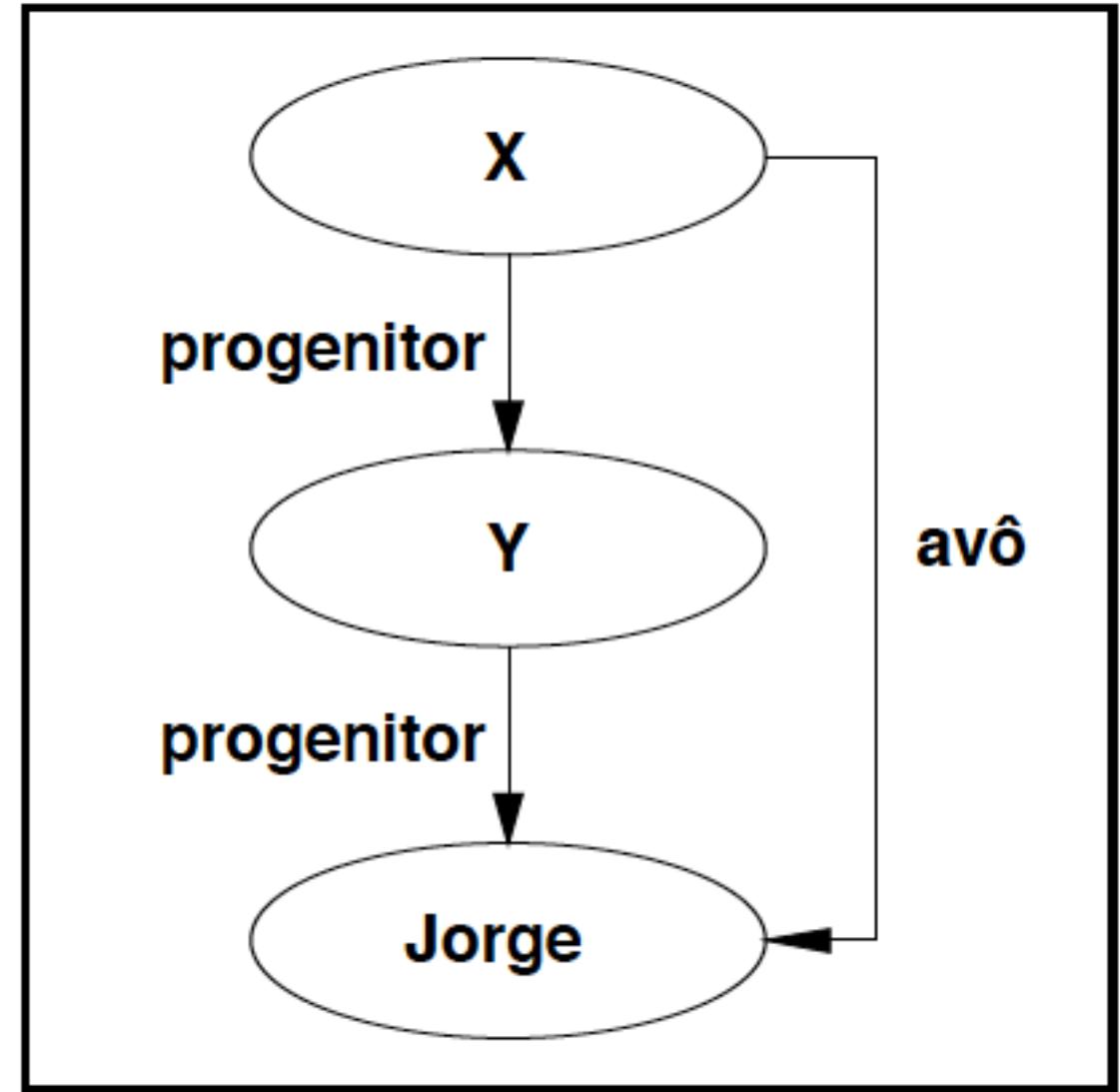
EXERCÍCIOS

PROLOG



Complementação da árvore genealógica

- Seja a base de conhecimento já implementada.
- Seja a relação avô em função de progenitor:



1. Mostre as consultas em PROLOG que respondam as seguintes perguntas:

a) Quem é neto de João?

b) José e Ana possuem algum progenitor em comum?

2. Crie uma relação avô, considerando a função progenitor já definida na base de conhecimento.

Função para cálculo do fatorial

- Calculado pela função:

$$n! = n * (n-1) !$$

- Em C:

Em PROLOG?

```
//Função recursiva que calcula o fatorial
//de um numero inteiro n
double factorial(int n)
{
    double vfat;

    if ( n <= 1 )
        //Caso base: fatorial de n <= 1 retorna 1
        return (1);
    else
    {
        //Chamada recursiva
        vfat = n * factorial(n - 1);
        return (vfat);
    }
}
```

```
fat(0, 1) :- !.  
fat(A, B) :- (A1 is A-1),  
            fat(A1, B1),  
            (B is A*B1).
```

Testar `fat(4, A)`.

3. Implementar uma função que calcule a série de Fibonacci:

$$a_1 = 1;$$

$$a_2 = 1;$$

$$a_n = a_{(n-1)} + a_{(n-2)}.$$

Os 7 primeiros valores são: 1, 1, 2, 3, 5, 8, 13

Definição de operadores

- Ver material para maiores detalhes.
- Formato:

`:- op(Prec, Tipo, Functor)`

onde:

- Prec: prioridade do operador (p. ex., entre 1 e 1200);
- Tipo: tipos dos operadores
- Functor: nome do operador

Definição de operadores

- Operador para a função factorial:

```
: - op(500, xfx, f) .
```

```
N f F :- fat(N, F) .
```

- Testar: 4 f X.

4. Implementar um operador para a série de Fibonacci.

6.6. Lógicas não-clássicas

- Até aqui, as lógicas vistas não consideram a possibilidade de que coisas sofram alterações ou que coisas não sejam sempre como agora
- Há muito tempo, estudiosos questionavam se a lógica clássica era suficiente para formalizar toda a realidade
- Por exemplo:

fecha_janela(joaquim)

Joaquim irá fechar
a janela

Joaquim fechou
a janela

6.6. Lógicas não-clássicas

- Objetivo das lógicas não-clássicas:
 - Formalizar argumentos que não são possíveis de ser formalizados pelos métodos clássicos.
- Características destas lógicas:
 - São baseadas em linguagens mais ricas em poder de expressão
 - São baseadas em princípios distintos
 - Admitem semânticas distintas.

6.6. Lógicas não-clássicas

- Segundo Haack (1978), podemos dividir as lógicas não-clássicas em dois principais tipos:
 - Complementares: não negam os axiomas/regras da lógica clássica, apenas ampliam com novos operadores/quantificadores e criam regras/axiomas para estes novos operadores/quantificadores.
 - Exemplos: lógicas temporal e modal.

6.6. Lógicas não-clássicas

- Alternativas: negam algum ou todos os axiomas da lógica clássica e pretendem substituí-la em muitos (ou todos) domínios da clássica.
 - Exemplos: lógicas que rejeitam o princípio de bivalência (V e F), como as lógicas de 3 valores e a lógica intuicionista.

a) Lógica temporal

- O tempo dá o tom
- O sistema precisa ter a capacidade de aceitar a mudança de estado lógico
- Inclui operadores para representar o tempo:

Operadores temporais

- **F**: algo será verdadeiro em algum tempo futuro
- **P**: algo foi verdadeiro em algum tempo passado
- **G**: algo será verdadeiro durante todo o futuro
- **H**: algo foi verdadeiro durante todo o passado

a) Lógica temporal

- Aspecto central: definição de fluxo de tempo, pontos do tempo e precedência temporal.

b) Lógica modal

- Trata de conceitos como necessidade e possibilidade
- 3 tipos de proposições:

Proposições obrigatoriamente verdadeiras	“Proposições necessariamente verdadeiras” OU “Proposições necessárias”
Proposições falsas	“Proposições impossíveis”
Proposições ora falsas ora verdadeiras	“Contigentes”

b) Lógica modal

- Se uma proposição é não-impossível, então ela é dita possível:
 - Proposições possíveis incluem todas, menos as impossíveis.

Operadores Modais

- L (ou \Box): é necessário que
- M (ou \Diamond) é possível que
- Exemplo: $Lp \vee q$
 - ou p é necessário ou q é verdade.
- Mais informações podem ser vistas no artigo de Moore (1984).

c) Lógica 3-Valores (L3V)

- Surgiram para preencher diversas lacunas que a clássica deixou para trás:
 - Solução de paradoxos semânticos (Lógica de Bochvar)
 - Afirmação com contingência futura (Lógica de Lukasiewicz)
 - Declarações matemáticas não decididas (Lógica de Kleene)

c) Lógica 3-Valores (L3V)

- Como o nome diz, além de V e F, admite um 3º valor.
- O 3º valor possui algumas interpretações e difere nas diversas lógicas, o que o torna uma fonte de diferentes semânticas:
 - Pode indicar um estado parcial de ignorância
 - Indica uma impossibilidade de se atribuir V ou F
 - Indica falta de sentido de se atribuir V ou F

i) Lógica de Kleene

- O 3º valor, u de *undecided* (não decidido), indica que “não se sabe se é V ou F” e não admite a interpretação de que “não é V nem F”.

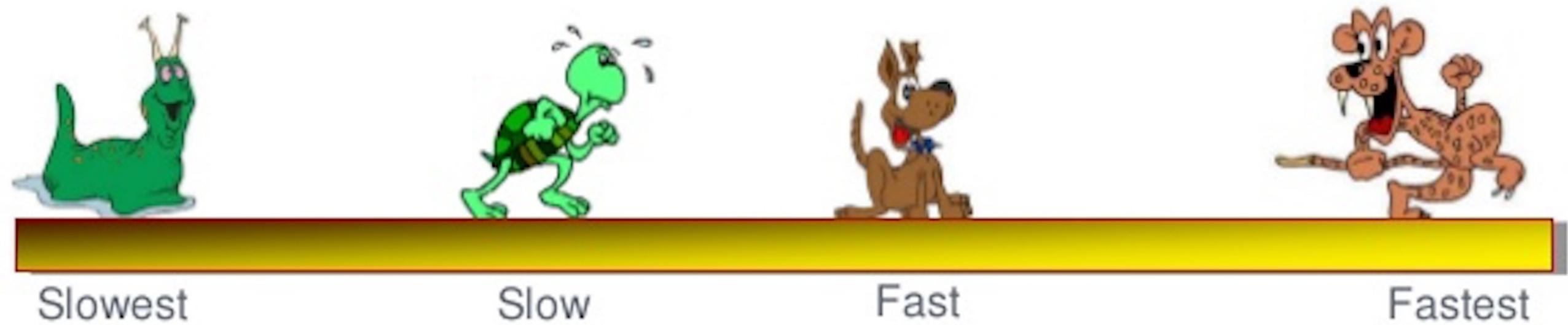
ii) Lógica de Lukasiewicz

- O 3º valor, i de *indeterminate*, deve ser entendido como “a declaração não é exatamente nem V nem F, mas indeterminada em algum sentido metafísico.”
- Diferença entre u (de Kleene) e o i: o i não é resultante de falta de informação e sim de impedimento de se poder fazer uma avaliação conclusiva sobre V ou F.

iii) Lógica de Bochvar

- O 3º valor, m de *meaningless* (paradoxal; sem significado) indica que a sentença não é nem verdadeira nem falsa.
- Exemplo:
 - “esta sentença é falsa”

d) Lógica Nebulosa (*Fuzzy*)





RESOLUÇÃO DE PROBLEMAS EM PROLOG

Problema dos missionários e canibais

- No Teams, abrir o documento MisCan.pl
- Copiar seu conteúdo para o PROLOG web.
- Testar a resolução: miss_can .

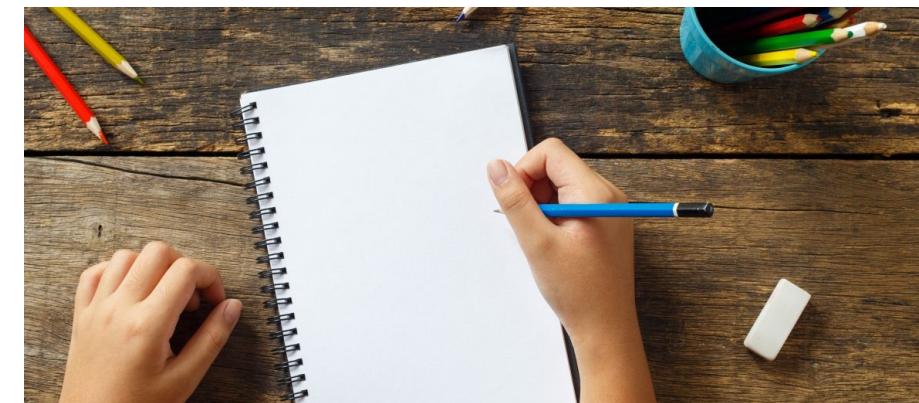
Problema dos músicos



- Determine a nacionalidade de cada músico, seu instrumento e a ordem de apresentação, considerando as seguintes informações/restricções:
 - Três músicos de uma banda multinacional revezam-se tocando solos em uma peça de música
 - Cada um toca um instrumento diferente
 - Cada um toca somente uma vez
 - O pianista toca primeiro

Problema dos músicos

- John toca saxofone e toca antes do australiano
- Mark é dos Estados Unidos e toca antes do violinista
- Um solista vem do Japão e um chama-se Sam



Problema dos músicos

- Abrir o documento `Musicos.pl`
- Copiar seu conteúdo para o PROLOG web.
- Testar a resolução: solução (S) .

Problema dos 3 vizinhos

- Numa pequena vila moram 3 vizinhos, todos de nacionalidades diferentes, um alemão, um brasileiro e um espanhol.
- Cada um cria animais diferentes, borboletas, cachorros e cavalos, e praticam esportes diferentes, futebol, sinuca e tênis.
- Cada um mora em uma casa com cores diferentes, azul, preta e branca.

Problema dos 3 vizinhos

- O brasileiro não mora na segunda casa.
- Quem cria cachorros gosta de jogar futebol.
- Tem uma casa entre o jogador de tênis e a casa preta, que fica a direita.
- O homem que cria cavalos mora exatamente do lado esquerdo do homem que cria borboletas.
- O homem que cria cachorros mora exatamente do lado direito da casa branca.
- O espanhol mora na terceira casa.



Problema dos 3 vizinhos

- Esse tem 2 soluções diferentes.
- Estarão na pasta de material de apoio, dentro de Lógicas para IA e Prolog, na pasta Códigos Prolog.

Fatos ou predicados

- A unidade básica do PROLOG é o predicado:
 - São fatos postulados e, portanto, são verdadeiros por definição
 - Representam o conhecimento tácito que se tem sobre o domínio.

- Exemplo:

gato(tom) .

rato(jerry) .

cao(spike) .

Interpretação dos fatos

- É livre por parte do programador.
- Nos exemplos anteriores:

gato (tom) . %tom é um gato

rato (jerry) . %jerry é um rato

cao (spike) . %spike é um cão

Interpretação dos fatos

- Mas o mesmo conhecimento pode ser expresso de diferentes maneiras:

tom (gato) .

jerry (rato) .

spike (cao) .

Interpretação dos fatos

- As diferenças de interpretação, no entanto, não são equivalentes:
 - As consultas são feitas sempre com relação a um predicado
 - No 2º exemplo, não conseguimos fazer uma consulta genérica para saber “quem é gato?” → gato(X) .

Interpretação dos fatos

- É interessante utilizar convenções sobre a interpretação.
- Exemplo:
pai (ana, beto).
pai (beto, ana).

Interpretação dos fatos

- É interessante utilizar convenções sobre a interpretação.
- Exemplo:
pai (ana, beto).
pai (beto, ana).



“beto é pai de ana”

Interpretação dos fatos

- É interessante utilizar convenções sobre a interpretação.
- Exemplo:

pai (ana, beto) .

pai (beto, ana) .

verbo (sujeito, objeto)

Interpretação dos fatos

- É interessante utilizar convenções sobre a interpretação.
- Exemplo:

pai (ana, beto) .

pai (beto, ana) .

verbo (objeto, sujeito)

Processamento de listas

- Listas: estruturas simples de dados → é uma sequência de itens
- Exemplo: uma lista com os países Brasil, Uruguai, Argentina e Paraguai, em PROLOG:

[brasil, uruguai, argentina, paraguai]

Processamento de listas

- Em PROLOG uma lista é representada como uma árvore
- Dois casos de listas:
 - Lista vazia: é considerada um átomo e é representada por []
 - Lista não-vazia: deve ser pensada como dois componentes, cabeça e corpo/cauda.

Processamento de listas

- No exemplo dos países:

[brasil, uruguai, argentina, paraguai]

Cabeça da lista

Corpo/cauda da lista

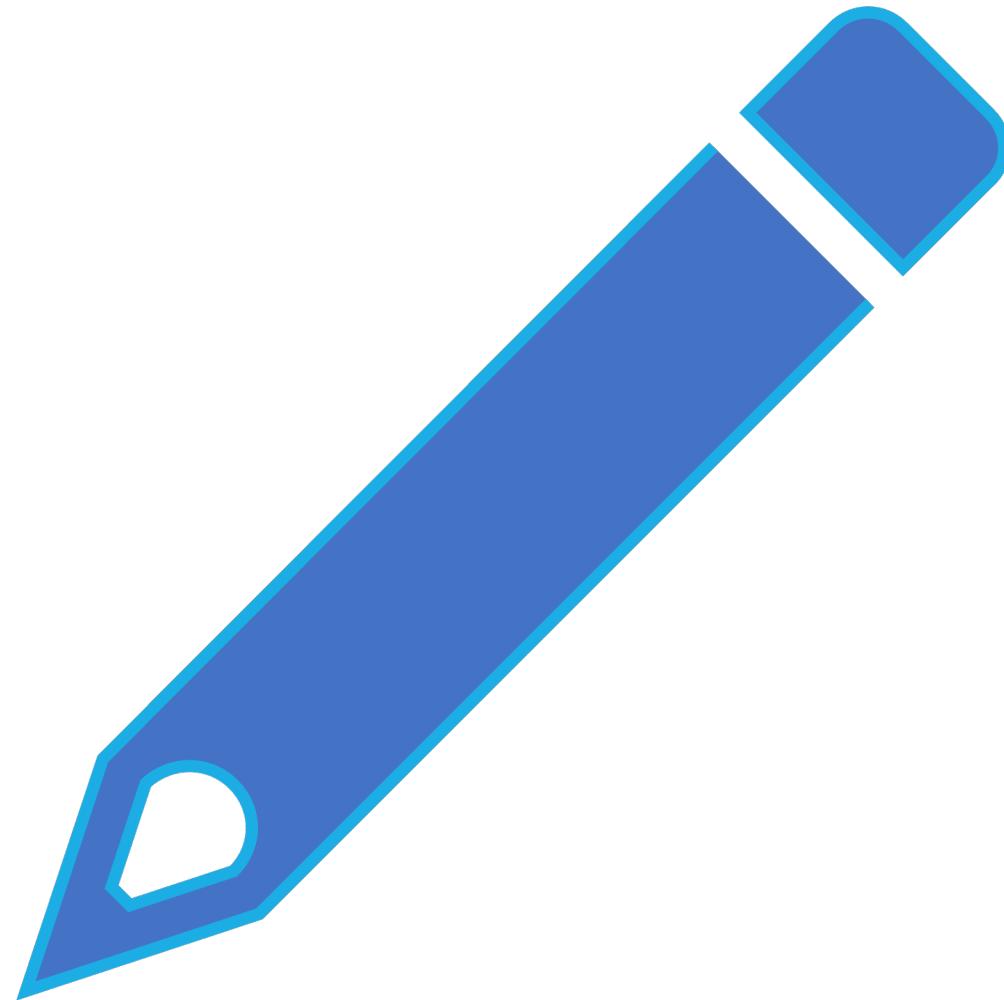
Processamento de listas

- A notação $[H | T]$ é tratada como uma lista de **cabeça** H e **corpo** T :
 - O corpo de uma lista é sempre outra lista, mesmo que seja vazia.

Operações com listas

- Documento no Teams (NocoesProlog.pdf).
- Abrir o documento e o software web para PROLOG.

EXERCÍCIOS



Exercícios

- Criar um programa PROLOG e colocar os seguintes predicados (disponíveis no documento Teams - NocoesProlog.pdf):
 1. Verificar se um elemento é membro de uma lista
 2. Concatenação de duas listas
 3. Inserir um elemento no início da lista
 4. Inserir um elemento no final da lista
 5. Encontrar o menor elemento da lista

Exercícios

6. Encontrar o maior elemento da lista
 7. Comprimento de uma lista
 8. Inverte uma lista
 9. Remover um elemento da lista
 10. Remover todas as ocorrências de um elemento da lista
- Testar os predicados. Exemplos de testes:

membro(a, [a, b, c, d, e]) .

membro(x, [a, b, c, d, e]) .

concat([a, b, c, d, e], [x, m, o, p], L) .

listIni(c, [a, x, d, m], L1) .

listFim(c, [a, x, d, m], L1) .

min([a, b, z, c, d, e], X) .

max([a, b, z, c, d, e], X) .

tam(criem uma lista qualquer, X).

inverte(criem uma lista qualquer, L1).

remove(c, [a, b, c, d, e, f], L).

remove(x, [a, b, c, d, e, f], L).

removeTodas(c, [a, b, c, d, e, f], L).

removeTodas(a, [a, b, c, d, e, f], L).



AGORA, MÃO NA MASSA...

1. No programa para listas, implementado em sala, inclua novos predicados conforme lista abaixo:

- a) Construa um predicado em Prolog que receba uma lista de números e retorne o somatório dos elementos da lista.
- b) Construa um predicado em Prolog que receba uma lista e retorne o último elemento da mesma.
- c) Construa um predicado em Prolog que receba uma lista e retorne o n-ésimo elemento da mesma.
- d) Construa um predicado em Prolog que substitua todas as ocorrências do elemento X por um elemento Y.



Referências Bibliográficas

AMADOR, R.M.S.T.; PIMENTÃO, J.P.B. **Representação do conhecimento através de “frames”**. Departamento de Informática da Universidade Nova de Lisboa - Portugal, 1988.

COPPIN, Ben. **Inteligência Artificial**. Rio de Janeiro: Ltc, 2013. 636 p.

ERTEL, Wolfgang. **Introduction to Artificial Intelligence**. 2. ed. Germany: Springer, 2017. Translated by Nathanael Black with illustrations by Florian Mast.

HAACK, Susan. **Philosophy of Logics**. Cambridge: Cambridge University Press, 1978.

LUGER, George F.. **Inteligência Artificial**. 6. ed. São Paulo: Pearson, 2013. 614 p.

MINKER, Jack (ed.). **Logic-Based Artificial Intelligence**. Boston: Kluwer Academic Publishers, 2000.

MONARD, M. C.; NICOLETTI, M. C. Programas Prolog para processamento de listas e aplicações. **Notas Didáticas do ICMSC**, n. 7, p. 71, 1993.

MOORE, Robert C.. A Formal Theory of Knowledge and Action. In: HOBBS, Jerry B.; MOORE, Robert C. (Org.). **Formal Theories of the Commonsense World**. New Jersey: Intellect Ltd, 1985. p. 1-84. (Ablex Series in Artificial Intelligence).

NILSSON, Ulf; MALUSZYNSKI, Jan. **Logic, Programming and Prolog**. 2. ed. Sweden: John Wiley & Sons Ltd., 2000.

PALAZZO, Luiz A. M.. Introdução à Programação Prolog. Pelotas: Editora da Universidade Católica de Pelotas, 1997. Disponível em: <<http://jeiks.net/wp-content/uploads/2013/06/prolog-palazzo.pdf>>. Acesso em: 07 abril 2021.

RABUSKE, Renato Antonio. **Inteligência Artificial**. Florianópolis: UFSC, 1995. 240 p. (Série Didática).

RUSSELL, Stuart; NORVIG, Peter. **Inteligência Artificial**. 3. ed. Rio de Janeiro: Elsevier, 2013. Tradução Regina Célia Simille.

TAKEUTI, Gaisi. First Order Predicate Calculus. In: TAKEUTI, Gaisi. **Proof Theory**. 81. ed. Illinois: Elsevier, 1975. Cap. 1. p. 5-67. (Studies in Logic and the Foundations of Mathematics).