Linguagens Lógicas

Paradigmas de Programação

Centro Universitário Senac

Prof. Celso Crivelaro

celso.vcrivelaro@sp.senac.br

Problemas lógicos

Alguns problemas são fáceis de descrever em linguagem humana e matemática. Exemplo: Quebra cabeças como Sudoku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Problemas lógicos

Provas de conceito e testes lógicos

Os corpos celeste dignos de nota são as estrelas, os planetas e os cometas.
Vênus é um corpo celeste, mas não é uma estrela.
Os cometas possuem cauda quando estão perto do sol.
Vênus está perto do sol, mas não possui cauda.

A última afirmação é verdadeira?

Problemas lógicos

Problemas de satisfação de restrições: Como alocar espaços baseado em regras e base de conhecimento

Para comemorar o aniversário de Cíntia, ela e mais quatro amigas – Alice, Bia, Dirce e Eunice – foram almoçar juntas no RU. As mesas são redondas e acomodam exatamente 5 pessoas. Cíntia e Dirce sentam-se uma ao lado da outra. Alice e Bia não sentam-se uma ao lado da outra.

As duas amigas sentadas ao lado de Eunice são:

1. Cintia e Alice
2. Cintia e Dirce
3. Alice e Bia
4. Dirce e Bia
5. Alice e Dirce

Linguagem Lógicas

Linguagens Lógicas

São Linguagens em que podemos nos expressar logicamente

Geralmente são declarativas: não se explica o que deve fazer (como na imperativas) e nem funções matemáticas (como funcionais). Expressam o mundo como é e consultas

Se define uma base de conhecimento, regras e consultas

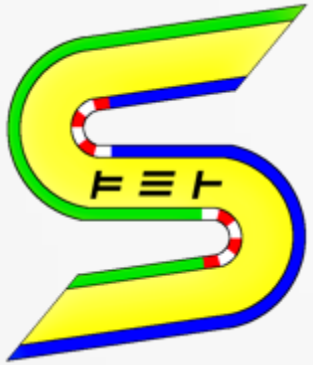
Exemplos

PICAT

http://picat-lang.org/

MERCURY

https://mercurylang.org/



http://www.gprolog.org/

SWI Prolog

https://www.swi-prolog.org/

Tau Prolog

http://tau-prolog.org/

Base teórica: Lógica

Literais

Literal: fórmula atômica ou sua negação

Literal Positivo: G => Gato é preto

Literal Negativo: NOT(G) => Gato não é preto

Cláusula

Expressão formada por uma coleção finita de literais.

Exemplo: C1 => G (Gato é preto) ^ W (Dorme em cima da Geladeira)

Cláusula Conjuntiva (E) => Verdadeiro se todos verdadeiros

Cláusula Disjuntiva (OU) => Verdadeiro se um for verdadeiro

Predicado

Função booleana: P: X -> {verdadeiro, falso}, X valor de entrada

É uma afirmação que pode ser V/F a depender dos valores de entrada

Exemplo: P(X) = { x | x >= 4}, V para 5 e F para 2

Lógica de Predicados ou Proposicional

Sistema lógico com predicados para inferência

Exemplo: Todo carro é dirigido por uma motorista

(Todos C) DirigeCarro(C) => Motorista (C)

Assim, podemos fazer inferência computacional sobre C

Só que a lógica proposicional é ruim de se representar computacionalmente

Cláusula de Horn

Clásula de Horn é composta de Cabeçalho ← Corpo

pato(a) ← !voa(a), nada(a), faz_quack(a), se “a” não voa, nada e faz quack, então é pato

Uma clásula de Horn precisa ter no máximo um predicado verdadeiro

Resolução. Podemos ter combinações: h ← t1, t2;
t ← t3, h, t4;

Então: t ← t3, t1, t2, t4

Unificação e Base de conhecimento

```
% Clásulas
falaLingua(Marie, francês)
falaLingua(Giovanni, italiano)
falaLingua(Pièrre, francês)
falaLingua(Noel, francês)
falaLingua(Lana, português)

% Predicado
podemConversar(X, Y) ←
falaLingua(X,L), falaLingua(Y,L),
X ≠ Y
% Inferência
podemConversar(Marie, Y)
```

Unificação é o processo de pattern Matching para verificar quais instância deixam a clásula verdadeira

No caso: falaLingua(Maria, francês), falalingua(Y, francês) e Marie ≠ Y

É verdadeiro quando: Y = Pièrre ou Y = Noel

Prolog

Prolog

Alain Colmeraeur e Robert Kowalski, criadores do Prolog

Criada em 1972 na França

Criada originalmente para Processamento de Linguagem Natural

Hoje temos diversas implementações com SWI Prolog e Gnu Prolog

Bases do Prolog

Fatos: Afirmações e representação do mundo

Regras: Inferência sobre os fatos

Busca: Pergunta sobre o mundo

Fatos

Afirmações sobre o mundo verdadeira, literais da lógica

gosta_de(maria, joao) => Se lê maria gosta de joao. Não é inverso.

maria => Átomo, parecido como uma string ou símbolo em outras linguagens. Deve iniciar por minúsculo

Pode-se usar quantos átomos quiser

Regras

Afirmações sobre o mundo verdadeira, literais da lógica

amizade(X,Y) :- gosta_de(X, Y), gosta_de(Y, X).

X é uma variável do Predicado.

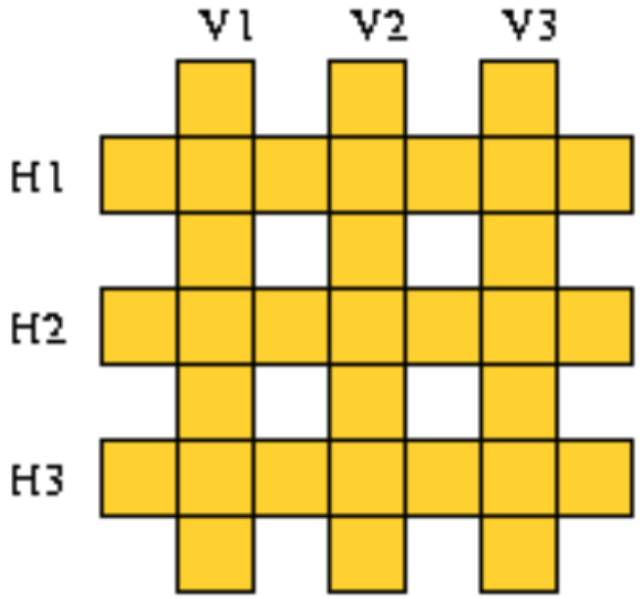
A regra se define: amizade com X e Y é quando gosta_de(X,Y) é verdadeiro e gosta_de(Y,X) também é verdadeiro

Exemplo

Exercise 2.4 *Here are six English words:*

abalone, abandon, anagram, connect, elegant, enhance.

They are to be arranged in a crossword puzzle like fashion in the grid given below



The following knowledge base represents a lexicon containing these words.

```
word(abalone,a,b,a,l,o,n,e).
word(abandon,a,b,a,n,d,o,n).
word(enhance,e,n,h,a,n,c,e).
word(anagram,a,n,a,g,r,a,m).
word(connect,c,o,n,n,e,c,t).
word(elegant,e,l,e,g,a,n,t).
```

Unificação

Unificação

A unificação pode é igual igualdade. Porém, é usada para associação de variáveis

pilotos(A, londres) = pilotos(sao_paulo, londres).

O valor de A será sao_paulo, porém, só por causa do match com londres

Unificação

A unificação poderá ser feita por por estruturas de dados

[A, 2, 5] = [1, 2, B]. => A = 1 e B = 5

A associação é feita pelos dois lados. Afina, igualdade é dos dois lados.

Unificação com Listas

Podemos obter dados de uma lista:

[4, 2, 5] = [Cabecalho|Corpo]. => Cabecalho = 4, Corpo = [2,5]

Podemos fazer uma regra recursive que colete um corpo:

[4, 2, 5] = [Cab1|[Cab2|Corpo]]. => Cab1 = 4, Cab2 = 2, Corpo = [5]

Regras Recursivas

Regras Recursivas

Podemos fazer regras recursivas. Toda Recursão precisa de um critério de parada.

ancestral(X, Y) :- mae(X, Y); pai(X, Y).=> critério de parada

ancestral(X, Y) :- (mae(X, Z); pai(X, Z)), ancestral(Z, Y). => recursão

São regras construídas a partir de **propriedades do problema**

Regras Recursivas

Podemos ter critérios de parada que são fatos. Exemplo: Checar de um elemento pertence a uma lista

pertence(X, [X|_]). => (Fato) Critério de parada

pertence(X, [_ | Y]) :- pertence(X, Y). => Regra Recursiva

Exercício

Exercício

Fazer um predicado ultimo(L,X) que é satisfeito quando o termo X é o último elemento da lista L

Fazer um predicado apaga(L1, X, L2) que é satisfeito quando L2 é a lista obtida pela remoção de todas as ocorrências de X em L1

Muito Obrigado!

Centro Universitário Senac

Prof. **Celso Crivelaro**
celso.vcrivelaro@sp.senac.br