CLÁUSULAS DE HORN

Estudante: _____

Estudante: _____

Estudante: _____

Estudante: _____

Visão Geral

Na **lógica matemática** e na **programação lógica,** uma **cláusula de Horn** é uma fórmula lógica de formato próprio, semelhante a uma **regra** utilizada em programação lógica

As cláusulas de Horn são nomeadas em homenagem ao matemático **Alfred Horn**[1], que primeiro destacou seu significado em 1951.

Programação Lógica

- Paradigma de programação baseado na lógica formal.
- Um programa escrito em uma linguagem de programação lógica é um conjunto de sentenças em forma lógica, que expressam **fatos** e **regras** sobre algum domínio de problema.

PROLOG

- Uma das principais linguagens de programação lógica.
- **Regras** em **Prolog** são escritas na forma de **cláusulas** no formato:

  H: - **B1**,…, **Bn**.

  H é chamado de cabeça da regra e **B1**, ..., **Bn** é chamado de corpo

  Que são lidas como implicações lógicas:

  H se **B1** e ... e **Bn**.

- **Fatos** em **Prolog** são regras que não têm corpo e são escritos no formato:

  H.

Definição

Na **Lógica dos Predicados**, uma fórmula atômica (indivisível) tem o seguinte formato:

$$P(t_1, t_2, \ldots, t_k)$$

Onde P é o símbolo de predicado e cada t_k é um termo.

[1] ***Alfred Horn*** (1918 –2001) foi um matemático americano conhecido pelo seu trabalho na Teoria dos Reticulados e na Álgebra Universal. Seu artigo de 1951 "**On sentences which are true of direct unions of algebras**" descreve as cláusulas de Horn e as sentenças de Horn as quais, mais tarde, seriam os fundamentos da **programação lógica**. (Wikipedia)

Exemplo:

$M(x)$ – é uma fórmula atômica (Ex. de Interpretação: *"x é matemático"*)

$P(x) \rightarrow Q(x)$ – não é uma fórmula atômica, pois tem o conectivo binário → (implicação)

Um **literal** pode ser:

- uma fórmula atômica (**literal positivo**)
- uma fórmula atômica negada (**literal negativo**).

Exemplo:

$\neg M(x)$ – é uma fórmula atômica negada (Ex. de Interpretação: *"x **não** é matemático"*)

Uma **cláusula** é uma disjunção de literais:

Exemplo:

$L_1 \vee L_2 \vee \ldots \vee L_n$

Onde cada L_n é um literal.

Uma **cláusula de Horn** é uma cláusula com no máximo um literal positivo, ou seja, não negado.
As Cláusulas de Horn são os blocos de construção básicos do Prolog.

Tipos de **Cláusulas de Horn** adequadas ao **Prolog**:

No Prolog	LITERAL POSITIVO	LITERAL NEGATIVO	Fórmula	Interpretação
Regra / *Rule* (clausula definitiva)	1	n	Disjuntiva $H \vee \neg G_1 \vee \neg G_2 \vee \ldots \vee \neg G_n$ Equivalente $H \vee \neg(G_1 \wedge G_2 \wedge \ldots \wedge G_n)$ Implicação $H \leftarrow (G_1 \wedge G_2 \wedge \ldots \wedge G_n)$	H é o ***head*** $(G_1 \wedge G_2 \wedge \ldots \wedge G_n)$ é o ***body*** Assume: ▪ Se $(G_1 \wedge G_2 \wedge \ldots \wedge G_n)$ Então H ▪ H se $(G_1 \wedge G_2 \wedge \ldots \wedge G_n)$ ▪ H é válido se $(G_1 \wedge G_2 \wedge \ldots \wedge G_n)$ é válido
Fato / *Fact* (cláusula incondicional)	1	0	$H \leftarrow$	Assume: ▪ H é válido incondicionalmente
Consulta / *Query* (cláusula meta)	0	n	$\leftarrow (G_1 \wedge G_2 \wedge \ldots \wedge G_n)$	Demonstre que: ▪ $(G_1 \wedge G_2 \wedge \ldots \wedge G_n)$ é válido

Símbolo	Linguagem Prolog
←	:-

∧	,
∨	;

No Prolog	Fórmula	Sintaxe Prolog	
Regra / *Rule* (clausula definitiva)	$H \leftarrow (G_1 \wedge G_2 \wedge \ldots \wedge G_n)$	***head* :- *body*.**	$H :- G_1, G_2, \ldots, G_n.$
Fato / *Fact* (cláusula incondicional)	$H \leftarrow$	***head* :-.**	$H.$
Consulta / *Query* (cláusula meta)	$\leftarrow (G_1 \wedge G_2 \wedge \ldots \wedge G_n)$	**:- *body*.**	$:- G_1, G_2, \ldots, G_n.$ $?- G_1, G_2, \ldots, G_n.$

Exercícios

QUESTÃO 1

Complete as diferentes representações das fórmulas a seguir, identificando o único literal não negativo para realizar representar em Prolog.

#	Fórmula Condicional	Fórmula Condicional	Prolog	Cláusulas de Horn
1.	(p ∧ r) → s	s ← (p ∧ r)	s :- p, r.	(¬ p ∨ ¬ r ∨ s)
2.				(¬ p ∨ ¬ q ∨ ¬ r ∨ ¬ s ∨ t)
3.			u :- p, r, t.	
4.	¬ (p ∨ ¬ s) → ¬ t			
5.			q :- r, w, y, z.	
6.				(¬ p ∨ ¬ q ∨ r ∨ ¬ s ∨ ¬ t ∨ ¬ u)
7.		u ← (r ∧ t ∧ s)		

QUESTÃO 2 - EXEMPLO (**Exercício**: Faça a resolução no Prolog: https://swish.swi-prolog.org/)
Considere o problema descrito a seguir.

Se Arthur gosta de lógica, lógica é fácil. Por outro lado, se Carla gosta de física, Arthur gosta de lógica. Da mesma forma, Carla gosta de física desde que Paula goste de lógica. O que podemos concluir se sabemos que Paula gosta de lógica?

a) Arthur não gosta de lógica e Carla gosta de física.
b) Carla não gosta de física e lógica é fácil.
c) Lógica é fácil e Arthur não gosta de lógica.
d) Carla gosta de física e lógica é fácil. (correta)

Predicados:

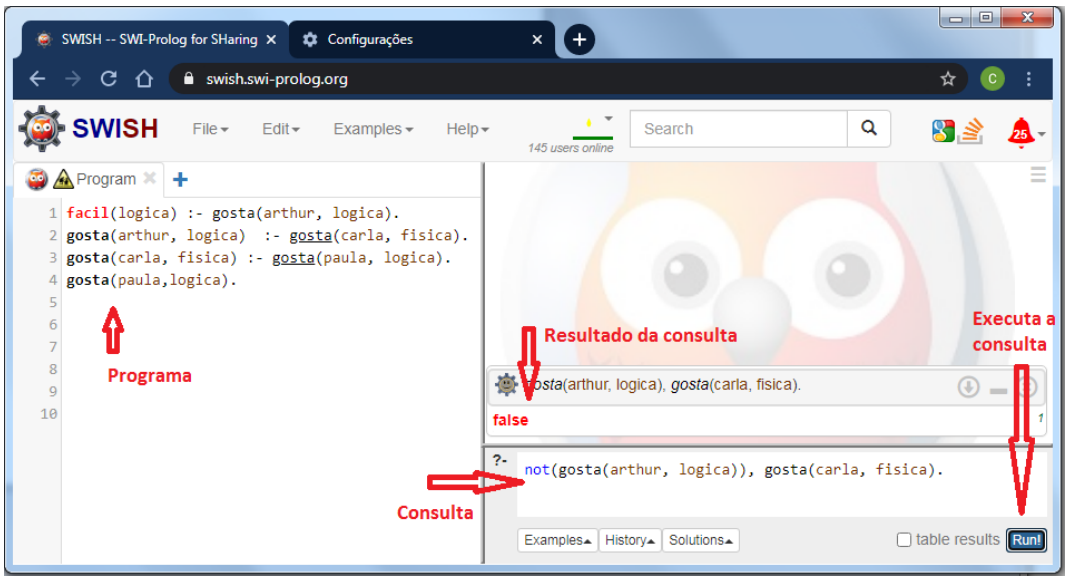
1. gosta (X, Y) = “X gosta de Y”.
2. fácil (X) = “X é fácil”.

Observação - em Prolog:
- *um **predicado** começa com letra **minúscula**;*
- *uma **variável** começa com letra **maiúscula**;*
- *uma **constante** começa com letra **minúscula**.*

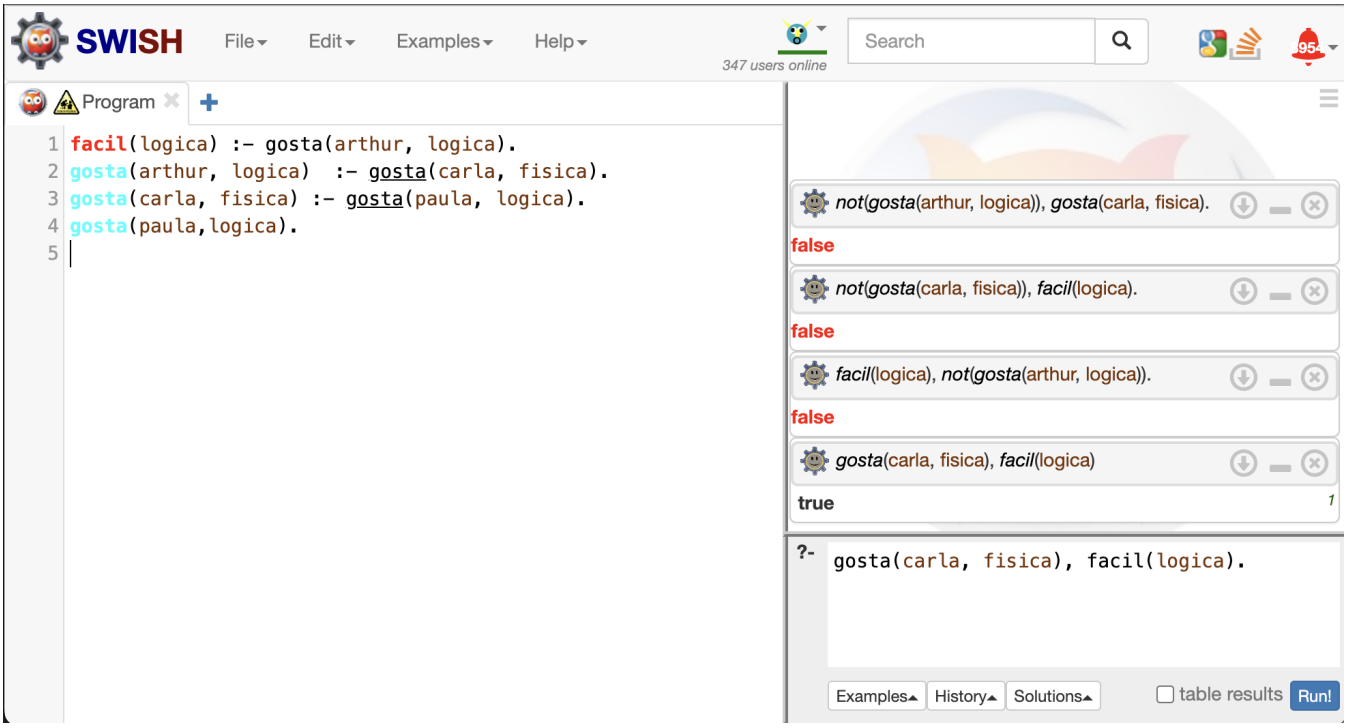
Tradução do problema para Prolog:

	Tipo	Lógica	Prolog
Programa em Prolog	Regra	facil(logica) ← gosta(arthur, logica)	`facil(logica) :- gosta(arthur, logica).`
	Regra	gosta(arthur, logica) ← gosta(carla, fisica)	`gosta(arthur, logica) :- gosta(carla, fisica).`
	Regra	gosta(carla, fisica) ← gosta(paula, logica)	`gosta(carla, fisica) :- gosta(paula, logica).`
	Fato	gosta(paula,logica) ←	`gosta(paula,logica).`
Consultas	a) F	← ¬ gosta(arthur, logica) ∧ gosta(carla, fisica)	`?- not(gosta(arthur, logica)), gosta(carla, fisica).`
	b) F	← ¬ gosta(carla, fisica) ∧ facil(logica)	`?- not(gosta(carla, fisica)), facil(logica).`
	c) F	← facil(logica) ∧ ¬ gosta(arthur, logica)	`?- facil(logica), not(gosta(arthur, logica)).`
	d) V	← gosta(carla, fisica) ∧ facil(logica)	`?- gosta(carla, fisica), facil(logica)`

Exemplo: executando **1 consulta** no Prolog:



Apresente o resultado das **suas 4 consultas** em uma única imagem: (substitua a imagem exemplo)



QUESTÃO 3

Considere o problema descrito a seguir.

João pesca quando está de férias. Desse modo, se João pesca, é verão. Já Pedro passeia de barco quando o tempo está limpo. Ou seja, se Pedro passeia de barco, mar está calmo. O que podemos concluir se sabemos que João está de férias e o tempo está limpo?

a) Não é verão e Pedro passeia de barco.
b) João não pesca e o mar não está calmo.
c) É verão, mas Pedro não passeia de barco.
d) Pedro passeia de barco e João pesca.

Predicados:

1. passeia (X, Y) = “X passeia de Y”.
2. esta (X,Y) = “X está Y”.
3. pesca (X) = “X pesca”.
4. eh (X) = “É X”.

Observação - em Prolog:

- *um **predicado** começa com letra **minúscula**;*
- *uma **variável** começa com letra **maiúscula**;*
- *uma **constante** começa com letra **minúscula**.*

1) Represente o problema na lógica dos predicados:

pesca(joao) ← esta(joao, ferias)

eh(verao) ← pesca(joao)

passeia(pedro, barco) ← esta(tempo, limpo)

esta(mar, calmo) ← passeia(pedro, barco)

esta(joao, ferias) ←

esta(tempo, limpo) ←

2) Traduza o problema para Prolog:

	Tipo	Lógica	Prolog
Programa em Prolog	Regra	pesca(joao) ← esta(joao, ferias)	`pesca(joao) :- esta(joao, ferias).`
	Regra	eh(verao) ← pesca(joao)	`eh(verao) :- pesca(joao).`
	Regra	passeia(pedro, barco) ← esta(tempo, limpo)	`passeia(pedro, barco) :- esta(tempo, limpo).`
	Regra	esta(mar, calmo) ← passeia(pedro, barco)	`esta(mar, calmo) :- passeia(pedro, barco).`
	Fato	esta(joao, ferias) ←	`esta(joao, ferias).`
	Fato	esta(tempo, limpo) ←	`esta(tempo, limpo).`
Consultas	a)	¬ eh(verao) ⋀ passeia(pedro, barco)	`?- not(eh(verao)), passeia(pedro, barco).`
	b)	¬ pesca(joao) ⋀ ¬ esta(mar, calmo)	`?- not(pesca(joao)), not(esta(mar, calmo)).`
	c)	eh(verao) ⋀ ¬ passeia(pedro, barco)	`?- eh(verao), not(passeia(pedro, barco))`
	d)	passeia(pedro, barco) ⋀ pesca(joao)	`?- passeia(pedro, barco), pesca(joao).`

3) Apresente o resultado das suas **4 consultas** em **Prolog** em uma única imagem: