**Atividade – Aula 19**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Disciplina** | | Data Analytics | | **Sala** | 14/15 | **Sprint** | 1 |
| **Docente** | | Arthur Gustavo de Araujo Ferreira | | **Data** | 28/08/2025 | **Hora** | 18:10 - 23:10 |
| **Aula** | 19 | **Assunto da Aula** | Prática em Prolog | | | | |
| **Aluno:** | Alan Diek da Silva Guimaraes | | | | | | |

|  |
| --- |
| **Lista de Exercícios de Prática em Prolog** |

|  |
| --- |
| Esta folha de exercícios contém práticas para aprender os conceitos básicos de Prolog. Use o  SWISH Online (https://swish.swi-prolog.org/) ou uma instalação local do SWI-Prolog para  rodar os exemplos e consultas. |

|  |
| --- |
| **1. Fatos e Regras**  1) Defina alguns fatos de família:  parent(maria, joao).  parent(joao, ana).  parent(joao, pedro).  2) Escreva uma regra para avô/avó:  grandparent(X,Z) :- parent(X,Y), parent(Y,Z).  A) Consultas para testar:  ?- parent(maria, X).  ?- grandparent(maria, Who). |
| **R:**  **parent(maria, X).**  **X = joao**  **grandparent(maria, Who).**  **Who = ana**  **Who = pedro** |

|  |
| --- |
| **2. Unificação e Retrocesso (Backtracking)**  1) Adicione fatos:  likes(alice, pizza).  likes(bob, salad).  likes(charlie, pizza).  a) Consulta:  ?- likes(X, pizza).  Observe como o Prolog encontra múltiplas respostas usando retrocesso. |
| **R:**  **likes(X, pizza).**  **X = alice**  **X = charlie** |

|  |
| --- |
| **3. Listas**  1) Defina o predicado member/2:  member(X, [X|\_]).  member(X, [\_|T]) :- member(X, T).  A) Consultas:  ?- member(3, [1,2,3,4]).  ?- member(X, [apple,banana,cherry]). |
| **R:**  **member(3, [1,2,3,4]).**  **true** |

|  |
| --- |
| **4. Chatbot Simples**  1) Defina respostas:  response(hello, 'Oi!').  response(hungry, 'Coma uma pizza!').  response(\_, 'Não entendi.').  A) Consultas:  ?- response(hello, R).  ?- response(hungry, R).  ?- response(weather, R). |
| **R:**  **response(hello, R).**  **R = 'Oi!'**  **R = 'Não entendi.'**  **response(hungry, R).**  **R = 'Coma uma pizza!'**  **R = 'Não entendi.'**  **response(weather, R).**  **R = 'Não entendi.'** |

|  |
| --- |
| **5. Desafio de Lógica**  Codifique o seguinte quebra-cabeça em Prolog:  - Alice, Bob e Carol têm cada um um animal de estimação diferente (gato, cachorro, peixe). - Alice não tem o cachorro.  - Bob não tem o peixe.  - Carol tem o gato.  Escreva regras em Prolog para determinar qual animal cada pessoa possui. |
| **R:**  **pessoa(alice).**  **pessoa(bob).**  **pessoa(carol).**  **animal(gato).**  **animal(cachorro).**  **animal(peixe).**  **dono(carol, gato).**  **dono(alice, Animal) :-**  **animal(Animal),**  **Animal \= cachorro,**  **Animal \= gato.**  **dono(bob, Animal) :-**  **animal(Animal),**  **Animal \= peixe,**  **Animal \= gato.**  **solucao(AnimalAlice, AnimalBob, AnimalCarol) :-**  **dono(alice, AnimalAlice),**  **dono(bob, AnimalBob),**  **dono(carol, AnimalCarol),**  **AnimalAlice \= AnimalBob,**  **AnimalAlice \= AnimalCarol,**  **AnimalBob \= AnimalCarol.**  **solucao(AnimalAlice, AnimalBob, AnimalCarol).**  **AnimalAlice = peixe, AnimalBob = cachorro, AnimalCarol = gato** |

|  |
| --- |
| **6. Desafio de Lógica**  Amanda, Bárbara, Clara, Davi e Emília moram em um prédio de cinco andares. Algumas  vezes os porteiros tem problemas em chamar moradores nos andares específicos em que  eles moram. Apesar disso eles podem deduzir quem mora em cada andar baseado em dicas  com informações que eles conseguem lembrar.  Descubra quem vive em cada andar do prédio.  1- Existem 5 andares. 1 (Térreo), 2,3,4 e 5 (cobertura).  2- Cada morador mora em um andar diferente.  3- Amanda não mora na cobertura.  4- Bárbara não mora no térreo.  5- Clara não mora tanto na cobertura ou no térreo.  6- Davi mora em um andar mais alto que Bárbara.  7- Emília não mora em um andar adjacente (vizinho) a Cora.  8- Clara não mora em um andar adjacente (vizinho) a Bárbara. |
| **R:**  **nao\_adjacente(Andar1, Andar2) :-**  **abs(Andar1 - Andar2) > 1.**  **solucao(Andares\_Corretos) :-**  **Andares\_Corretos = [**  **mora(amanda, Andar\_da\_Amanda),**  **mora(barbara, Andar\_da\_Barbara),**  **mora(clara, Andar\_da\_Clara),**  **mora(davi, Andar\_do\_Davi),**  **mora(emilia, Andar\_da\_Emilia)**  **],**  **permutation([1, 2, 3, 4, 5], [Andar\_da\_Amanda, Andar\_da\_Barbara, Andar\_da\_Clara, Andar\_do\_Davi, Andar\_da\_Emilia]),**  **Andar\_da\_Amanda \= 5,**  **Andar\_da\_Barbara \= 1,**  **Andar\_da\_Clara \= 5,**  **Andar\_da\_Clara \= 1,**  **Andar\_do\_Davi > Andar\_da\_Barbara,**  **nao\_adjacente(Andar\_da\_Emilia, Andar\_da\_Clara),**  **nao\_adjacente(Andar\_da\_Clara, Andar\_da\_Barbara).**  **solucao(Andares\_Corretos). Andares\_Corretos = [*mora(*amanda,3), *mora(*barbara,2), *mora(*clara,4), *mora(*davi,5), *mora(*emilia,1)]** |

|  |
| --- |
| **7. Desafio de Lógica**  Monte um esquema em Prolog que faça para todos estados brasileiros um esquema de 4 cores em que nenhum estado possua a mesma cor que seu vizinho. |
| **color(red).**  **color(green).**  **color(blue).**  **color(yellow).**  **neighbor(X,Y) :- adj(X,Y) ; adj(Y,X).**  **adj(acre, amazonas).**  **adj(acre, rondonia).**  **adj(alagoas, bahia).**  **adj(alagoas, pernambuco).**  **adj(alagoas, sergipe).**  **adj(amapa, para).**  **adj(amazonas, mato\_grosso).**  **adj(amazonas, para).**  **adj(amazonas, rondonia).**  **adj(amazonas, roraima).**  **adj(bahia, espirito\_santo).**  **adj(bahia, goias).**  **adj(bahia, minas\_gerais).**  **adj(bahia, pernambuco).**  **adj(bahia, piaui).**  **adj(bahia, sergipe).**  **adj(bahia, tocantins).**  **adj(ceara, paraiba).**  **adj(ceara, pernambuco).**  **adj(ceara, piaui).**  **adj(ceara, rio\_grande\_do\_norte).**  **adj(distrito\_federal, goias).**  **adj(distrito\_federal, minas\_gerais).**  **adj(espirito\_santo, minas\_gerais).**  **adj(espirito\_santo, rio\_de\_janeiro).**  **adj(goias, mato\_grosso).**  **adj(goias, mato\_grosso\_do\_sul).**  **adj(goias, minas\_gerais).**  **adj(goias, tocantins).**  **adj(maranhao, para).**  **adj(maranhao, piaui).**  **adj(maranhao, tocantins).**  **adj(mato\_grosso, para).**  **adj(mato\_grosso, rondonia).**  **adj(mato\_grosso, tocantins).**  **adj(mato\_grosso\_do\_sul, minas\_gerais).**  **adj(mato\_grosso\_do\_sul, parana).**  **adj(mato\_grosso\_do\_sul, sao\_paulo).**  **adj(minas\_gerais, rio\_de\_janeiro).**  **adj(minas\_gerais, sao\_paulo).**  **adj(para, roraima).**  **adj(para, tocantins).**  **adj(paraiba, pernambuco).**  **adj(paraiba, rio\_grande\_do\_norte).**  **adj(parana, santa\_catarina).**  **adj(parana, sao\_paulo).**  **adj(pernambuco, piaui).**  **adj(piaui, tocantins).**  **adj(rio\_grande\_do\_sul, santa\_catarina).**  **states([**  **acre, alagoas, amapa, amazonas, bahia, ceara, distrito\_federal, espirito\_santo,**  **goias, maranhao, mato\_grosso, mato\_grosso\_do\_sul, minas\_gerais, para, paraiba,**  **parana, pernambuco, piaui, rio\_de\_janeiro, rio\_grande\_do\_norte, rio\_grande\_do\_sul,**  **rondonia, roraima, santa\_catarina, sao\_paulo, sergipe, tocantins**  **]).**  **color\_brazil\_3(Colors) :-**  **states(Ss),**  **color\_states(Ss, [], Colors).**  **color\_states([], Acc, Acc).**  **color\_states([S|Ss], AccIn, AccOut) :-**  **color(C),**  **\+ ( member(S2-C2, AccIn),**  **neighbor(S, S2),**  **C = C2 ),**  **color\_states(Ss, [S-C|AccIn], AccOut).**  **color\_brazil\_3\_one(Colors) :-**  **once(color\_brazil\_3(Colors)).**  **Resultado:  Colors = [tocantins*-*yellow, sergipe*-*blue, sao\_paulo*-*blue, santa\_catarina*-*blue, roraima*-*red, rondonia*-*blue, rio\_grande\_do\_sul*-*red, rio\_grande\_do\_norte*-*blue, rio\_de\_janeiro*-*green, piaui*-*blue, pernambuco*-*yellow, parana*-*green, paraiba*-*green, para*-*blue, minas\_gerais*-*yellow, mato\_grosso\_do\_sul*-*red, mato\_grosso*-*red, maranhao*-*red, goias*-*blue, espirito\_santo*-*red, distrito\_federal*-*red, ceara*-*red, bahia*-*green, amazonas*-*green, amapa*-*red, alagoas*-*red, acre*-*red]** |