UNIVERSIDADE FEDERAL DE SANTA CATARINA – UFSC- CTC
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
INTELIGÊNCIA ARTIFICIAL SIMBÓLICA
Aluno: Osmar de Oliveira Braz Junior(osmar.braz@udesc.br),
Maciel Hogenn(maciel.hog@gmail.com)

Lista de Exercícios – Lógica, Método da Resolução e Método de Tableaux

1. Dados os seguintes conhecimentos:

(a) Se a seleção brasileira jogar bem, ganha a copa do mundo.
(b) Se a seleção brasileira não jogar bem, a culpa é do técnico da seleção.
(c) Se a seleção brasileira jogar bem, os torcedores fazem festa.
(d) Os torcedores não fazem festa.

• Represente-os em lógica proposicional. Converta as fórmulas obtidas para uma forma normal conjuntiva. Demonstre se possível, utilizando o método da resolução, que o técnico é culpado?

P-> "seleção brasileira joga bem"
Q-> "ganha à copa do mundo"
R-> "o técnico da seleção é culpado"
S-> "os torcedores fazem festa"

(a) P → Q // ¬P V Q
(b) ¬P → R // ¬¬P V R // P V R
(c) P → S // ¬P V S
(d) ¬S // ¬S

Forma Normal Conjuntiva
(a) (¬P V Q)
(b) (P V R)
(c) (¬P V S)
(d) (¬S)
(¬P V Q) Λ (P V R) Λ(¬P V S) Λ(¬S)

O técnico é culpado?
((¬P V Q) Λ (P V R) Λ(¬P V S) Λ(¬S)) →R

1. ¬ (((¬P V Q) Λ (P V R) Λ(¬P V S) Λ(¬S)) →R) //Negar o teorema	
2. (¬P V Q) Λ (P V R) Λ(¬P V S) Λ(¬S) // de 1	
3. ¬R // de 1	
4. (¬P V Q) // de 2	
5. (P V R) // de 2	
6. (¬P V S) // de 2	
7. ¬S // de 2	
8. ¬P // de 4	8.1 Q // de 4
9. P // de 5	9.1 R // de 5
10. ¬P // de 6	10.1 S // de 6

10.1 e 7 S e ¬S
10 e 9 P e ¬P
9.1 e 3 R e ¬R
9 e 8 ¬P e P

O argumento não é valido, pois Q não pode ser negado.

2. Represente os seguintes conhecimentos em lógica de predicados:

(a) Toda a criança gosta de chocolate. Adultos não gostam de chocolate. Pedro é criança. Raquel é adulta. Carla gosta de chocolate.

P-> Toda criança gosta de chocolate
Q->Adulto não gostam de chocolate.
R->Pedro é criança
S->Raquel é adulta
T-> Carla gosta de chocolate

(a) crianca(Pedro) //crianca(x3)
(b) ¬crianca(Raquel) //¬crianca(x4)
(c) ∀x.crianca(x) → gosta(x, chocolate) //¬crianca(x1) V gosta(x1, chocolate)
(d) ∀x.¬crianca(x) → ¬gosta(x, chocolate) //crianca(x2) V ¬gosta(x2, chocolate)
(e) gosta(Carla,chocolate) // gosta(x5,chocolate)

Forma Normal Conjuntiva

crianca(x3) Λ ¬crianca(x4) Λ (¬crianca(x1) V gosta(x1, chocolate)) Λ(crianca(x2) V ¬gosta(x2, chocolate)) Λ gosta(x5,chocolate)

Quem gosta de chocolate? gosta(x, chocolate)

¬(crianca(x3) Λ ¬crianca(x4) (¬crianca(x1) V gosta(x1, chocolate)) Λ(crianca(x2) V ¬gosta(x2, chocolate)) Λ gosta(x5,chocolate) → gosta(x, chocolate)) //negar o teorema

¬(¬(crianca(x3) Λ ¬crianca(x4) Λ (¬crianca(x1) V gosta(x1, chocolate)) Λ(crianca(x2) V ¬gosta(x2, chocolate)) Λ gosta(x5,chocolate) V gosta(x, chocolate)) // Elimina implicação

¬(¬(crianca(x3) Λ ¬crianca(x4) Λ (¬crianca(x1) V gosta(x1, chocolate)) Λ(crianca(x2) V ¬gosta(x2, chocolate)) Λ gosta(x5,chocolate)) Λ ¬(gosta(x, chocolate))//Distribuir a negação

(crianca(x3) Λ ¬crianca(x4) Λ (¬crianca(x1) V gosta(x1, chocolate)) Λ(crianca(x2) V ¬gosta(x2, chocolate)) Λ gosta(x5,chocolate) Λ ¬(gosta(x, chocolate))//Eliminando a dupla negação
((¬crianca(x1) V gosta(x1, chocolate)) Λ(crianca(x2) V ¬gosta(x2, chocolate)) Λ gosta(x5,chocolate) Λ ¬(gosta(x, chocolate) //Elimina //crianca(x3) e ¬crianca(x4)

((¬crianca(x1) V gosta(x1, chocolate)) Λ(crianca(x2) V ¬gosta(x2, chocolate)) //Elimina gosta(x5,chocolate) e ¬(gosta(x, chocolate)

(gosta(x1, chocolate) Λ ¬gosta(x2, chocolate) //Elimina ¬crianca(x1) e crianca(x2)
□ // Elimina gosta(x1, chocolate) e ¬gosta(x2, chocolate)

Carla é adulto? ¬crianca(Carla)

crianca(x3) Λ ¬crianca(x4) (¬crianca(x1) V gosta(x1, chocolate)) Λ(crianca(x2) V ¬gosta(x2, chocolate)) Λ gosta(x5,chocolate) → ¬crianca(Carla)

¬(crianca(x3) Λ ¬crianca(x4) (¬crianca(x1) V gosta(x1, chocolate)) Λ(crianca(x2) V ¬gosta(x2, chocolate)) Λ gosta(x5,chocolate) → ¬crianca(Carla)) //negar o teorema

¬(¬(crianca(x3) Λ ¬crianca(x4) Λ (¬crianca(x1) V gosta(x1, chocolate)) Λ(crianca(x2) V ¬gosta(x2, chocolate)) Λ gosta(x5,chocolate) V ¬crianca(Carla)// Elimina implicação

¬(¬(crianca(x3) Λ ¬crianca(x4) Λ (¬crianca(x1) V gosta(x1, chocolate)) Λ(crianca(x2) V ¬gosta(x2, chocolate)) Λ gosta(x5,chocolate)) Λ ¬(¬crianca(Carla))//Distribuir a negação

(crianca(x3) Λ ¬crianca(x4) Λ (¬crianca(x1) V gosta(x1, chocolate)) Λ(crianca(x2) V ¬gosta(x2, chocolate)) Λ gosta(x5,chocolate) Λ crianca(Carla)//Eliminando a dupla negação

((¬crianca(x1) V gosta(x1, chocolate)) Λ(crianca(x2) V ¬gosta(x2, chocolate)) Λ gosta(x5,chocolate) Λ crianca(Carla) //Elimina //crianca(x3) e ¬crianca(x4)

(gosta(x1, chocolate) Λ(¬gosta(x2, chocolate) Λ gosta(x5,chocolate) Λ crianca(Carla) // Elimina ¬crianca(x1) e crianca(x2)

gosta(x5,chocolate) Λ crianca(Carla) // Elimina gosta(x1, chocolate) e ¬gosta(x2, chocolate)

/Não foi possível negar a proposição.

(b) Todas as pessoas gostam de alguma coisa. Homens e mulheres são pessoas. Todos os homens gostam de futebol ou de corridas de automóveis. Mulheres gostam de filmes românticos e seriados. Carla é mulher. João gosta de futebol.

mulher(carla) // Carla é mulher
gostafutebol(joao) // João gosta de futebol.
pessoa(homem) // Homens e mulheres são pessoas
pessoa(mulher) // Homens e mulheres são pessoas

// Todas as pessoas gostam de alguma coisa.
∀x.pessoa(x) → gostafutebol(x) Λ gostacorrida(x) Λ gostafilme(x) Λ gostaseriados(x) //¬pessoa(x1) V gostafutebol(x1) Λ gostacorrida(x1) Λ gostafilme(x1) Λ gostaseriados(x1)
ou
∀x.pessoa(x) → gosta(x,futebol) Λ gosta (x, corrida) Λ gosta (x, filme) Λ gosta (x,seriado) //¬pessoa(x1) V gosta(x,futebol) Λ gosta (x, corrida) Λ gosta (x, filme) Λ gosta (x,seriado)

// Todos os homens gostam de futebol ou de corridas de automóveis.
∀x.pessoa(x) Λ ¬mulher(x) Λ gostafutebol(x) V gostacorrida(x) // pessoa(x2) Λ ¬mulher(x2) Λ gostafutebol(x2) V gostacorrida(x2)
ou
∀x.pessoa(x) Λ ¬mulher(x) Λ gostafutebol(x) V gostacorrida(x) // pessoa(x2) Λ ¬mulher(x2) Λ gosta(x2,futebol) V gosta(x2, corrida)

// Mulheres gostam de filmes românticos e seriados.
∀x.pessoa(x) Λ mulher(x) Λ gostafilme(x) Λ gostaseriados(x) // pessoa(x3) Λ mulher(x3) Λ gostafilme(x3) Λ gostaseriados(x3)
ou
∀x.pessoa(x) Λ mulher(x) Λ gosta(x, filme) Λ gosta (x, seriado) // pessoa(x3) Λ mulher(x3) Λ gosta(x3,filme) Λ gosta(x3,seriados)

Forma Normal Conjuntiva

mulher(carla) Λ gostafutebol(joao) Λ pessoa(homem) Λ pessoa(mulher) Λ (¬pessoa(x1) V gostafutebol(x1) Λ gostacorrida(x1) Λ gostafilme(x1) Λ gostaseriados(x1)) Λ (pessoa(x2) Λ ¬mulher(x2) Λ gostafutebol(x2) V gostacorrida(x2)) Λ (pessoa(x3) Λ mulher(x3) Λ gostafilme(x3) Λ gostaseriados(x3))

ou

mulher(carla) Λ gosta(joao, futebol) Λ pessoa(homem) Λ pessoa(mulher) Λ (¬pessoa(x1) V gosta(x1,futebol) Λ gosta(x1,corrida) Λ gosta(x1, filme) Λ gosta(x1, seriados)) Λ (pessoa(x2) Λ ¬mulher(x2) Λ gosta(x2, futebol) V gosta(x2, corrida)) Λ (pessoa(x3) Λ mulher(x3) Λ gosta(x3, filme) Λ gosta(x3,seriado))

João é uma pessoa? pessoa(joao)

¬ (mulher(carla) Λ gostafutebol(joao) Λ pessoa(homem) Λ pessoa(mulher) Λ (¬pessoa(x1) V gostafutebol(x) Λ gostacorrida(x1) Λ gostafilme(x1) Λ gostaseriados(x1)) Λ (pessoa(x2) Λ ¬mulher(x2) Λ gostafutebol(x2) V gostacorrida(x2)) Λ (pessoa(x3) Λ mulher(x3) Λ gostafilme(x3) Λ gostaseriados(x3)) → pessoa(joao))//negar o teorema

¬¬(mulher(carla) Λ gostafutebol(joao) Λ pessoa(homem) Λ pessoa(mulher) Λ (¬pessoa(x1) V gostafutebol(x1) Λ gostacorrida(x1) Λ gostafilme(x1) Λ gostaseriados(x1)) Λ (pessoa(x2) Λ ¬mulher(x2) Λ gostafutebol(x2) V gostacorrida(x2)) Λ (pessoa(x3) Λ mulher(x3) Λ gostafilme(x3) Λ gostaseriados(x3))) V pessoa(joao))//remover a implicação

(mulher(carla) Λ gostafutebol(joao) Λ pessoa(homem) Λ pessoa(mulher) Λ (¬pessoa(x1) V gostafutebol(x1) Λ gostacorrida(x1) Λ gostafilme(x1) Λ gostaseriados(x1)) Λ (pessoa(x2) Λ ¬mulher(x2) Λ gostafutebol(x2) V gostacorrida(x2)) Λ (pessoa(x3) Λ mulher(x3) Λ gostafilme(x3) Λ gostaseriados(x3))) V pessoa(joao))//remover a dupla negação

Do que Carla gosta? gosta(carla,x)

¬(mulher(x) Λ gostafutebol(y) Λ pessoa(homem) Λ pessoa(mulher) Λ (¬pessoa(x1) V gosta(x1,futebol) Λ gosta(x1,corrida) Λ gosta(x1, filme) Λ gosta(x1, seriado)) Λ (pessoa(x2) Λ ¬mulher(x2) Λ gosta(x2, futebol) V gosta(x2, corrida)) Λ (pessoa(x3) Λ mulher(x3) Λ gosta(x3, filme) Λ gosta(x3,seriado)) → gosta(carla,x)) //negar o teorema

¬(¬(mulher(carla) Λ gostafutebol(joao) Λ pessoa(homem) Λ pessoa(mulher) Λ (¬pessoa(x1) V gosta(x1,futebol) Λ gosta(x1,corrida) Λ gosta(x1, filme) Λ gosta(x1, seriado)) Λ (pessoa(x2) Λ ¬mulher(x2) Λ gosta(x2, futebol) V gosta(x2, corrida)) Λ (pessoa(x3) Λ mulher(x3) Λ gosta(x3, filme) Λ gosta(x3,seriado))) V gosta(carla,x)) //remover a implicação

(mulher(carla) Λ gostafutebol(joao) Λ pessoa(homem) Λ pessoa(mulher) Λ (¬pessoa(x1) V gosta(x1,futebol) Λ gosta(x1,corrida) Λ gosta(x1, filme) Λ gosta(x1, seriado)) Λ (pessoa(x2) Λ ¬mulher(x2) Λ gosta(x2, futebol) V gosta(x2, corrida)) Λ (pessoa(x3) Λ mulher(x3) Λ gosta(x3, filme) Λ gosta(x3,seriado))) V gosta(carla,x)) //remover a dupla negação

Caso não seja possível responder alguma das perguntas acima, descreva o que precisa ser modificado na base para extrair as respostas.

3. Prove os seguintes teoremas utilizando o método da Resolução e o método de Tableaux (lembre-se de negar o teorema):

→ implica
∃ Quantificador existencial (Existe, Para algum, nem todos, somente alguns)
Λ and e
V or ou
¬ not neg negação
∀ Quantificador universal, (Qualquer que seja, Para todo, para cada, qualquer um, todos eles)
□

d(a) Resolução

(PΛ¬Q) → ¬(P → Q)

¬ ((PΛ¬Q) → ¬(P → Q)) //Negar o teorema
¬ (¬(PΛ¬Q) V ¬(P → Q)) //Eliminação da primeira implicação
¬ (¬(PΛ¬Q) V ¬(¬P V Q)) //Eliminação da segunda implicação
¬ (¬(PΛ¬Q)) Λ ¬(¬(¬P V Q)) //Distribuindo a negação
(PΛ¬Q) Λ (¬P V Q) //Eliminação da dupla negação
(PΛ ¬P) //Eliminação do Q pois ¬Q e Q
□ //Eliminação do P pois P e ¬P

(a) Tableaux

(PΛ¬Q) → ¬(P → Q)

1. ¬ ((PΛ¬Q) → ¬(P → Q)) //Negar o teorema	
2. (PΛ¬Q) // de 1	
3. ¬(¬(P → Q)) // de 1	
4. (P → Q) // de 3	
5. P // de 2	
6. ¬Q // de 2	
7. ¬P // de 4	7.1 Q // de 4

(b) Resolução

$\neg(\neg P \land \neg Q) \rightarrow (\neg P \rightarrow Q)$

$\neg(\neg(\neg P \land \neg Q) \rightarrow (\neg P \rightarrow Q))$ //Negar o teorema

$\neg(\neg(\neg(\neg P \land \neg Q)) \lor (\neg P \rightarrow Q)$ //Eliminação da primeira implicação

$\neg((\neg P \land \neg Q) \lor (\neg P \rightarrow Q))$ // Eliminação da dupla negação

$\neg((\neg P \land \neg Q) \lor (\neg(\neg P) \lor Q))$ //Eliminação da segunda implicação

$\neg((\neg P \land \neg Q) \lor (P \lor Q))$ // Eliminação da dupla negação

$\neg(\neg P \land \neg Q) \land \neg(P \lor Q)$ //Distribuindo a negação

$(\neg(\neg P) \lor \neg(\neg Q)) \land \neg(P \lor Q)$ //Distribuindo a negação

$(\neg(\neg P) \lor \neg(\neg Q)) \land (\neg P \land \neg Q)$ //Distribuindo a negação

$(P \lor Q) \land (\neg P \land \neg Q)$ // Eliminação da dupla negação

$(P \land \neg P)$ //Eliminação do Q pois Q e ¬Q

□ //Eliminação do P pois P e ¬P

(b) Tableau

$\neg(\neg P \land \neg Q) \rightarrow (\neg P \rightarrow Q)$

1. $\neg(\neg(\neg P \land \neg Q) \rightarrow (\neg P \rightarrow Q))$ //Negar o teorema	
2. $\neg(\neg P \land \neg Q)$ // de 1	
3. $\neg(\neg P \rightarrow Q)$ // de 1	
4. $\neg P$ // de 3	
5. $\neg Q$ // de 3	
6. $\neg(\neg P)$ // de 2	6.1 $\neg(\neg Q)$ // de 2
7. P // de 6	7.1 Q // de 6.1

(c) Resolução

(P Λ (¬Q V R)) → ((P Λ ¬Q) V (P Λ R))
¬((P Λ (¬Q V R)) → ((P Λ ¬Q) V (P Λ R))) //Negar o teorema
¬(¬(P Λ (¬Q V R)) V ((PΛ¬Q) V (P Λ R)))//Eliminação da implicação
¬(¬(P Λ (¬Q V R))) Λ ¬((PΛ¬Q) V (P Λ R)) //Distribuindo a negação
(P Λ (¬Q V R)) Λ ¬((PΛ¬Q) V (P Λ R)) //Eliminação da dupla negação
(P Λ (¬Q V R)) Λ (¬(PΛ¬Q) Λ ¬(P Λ R)) //Distribuindo a negação
(P Λ (¬Q V R)) Λ ((¬PV¬(¬Q)) Λ (¬P V ¬R)) //Distribuindo a negação
(P Λ (¬Q V R)) Λ ((¬PVQ) Λ (¬P V ¬R)) //Eliminação da dupla negação
((PΛ¬Q)V (PΛR)) Λ ((¬PV Q) Λ (¬P V ¬R)) //Distributividade
(¬Q V (PΛR)) Λ ((¬PV Q) Λ ¬R) // Eliminação do P pois P e ¬P
(¬Q V R) Λ (Q Λ ¬R) // Eliminação do P pois P e ¬P
(R Λ ¬R) // Eliminação do Q pois ¬Q e Q
□ //Eliminação do R pois R e ¬R

(c) Tableau

(P Λ (¬Q V R)) → ((P Λ ¬Q) V (P Λ R))

1. ¬ ((P Λ (¬Q V R)) → ((P Λ ¬Q) V (P Λ R))) //Negar o teorema	
2. (P Λ (¬Q V R)) // de 1	
3. ¬((P Λ ¬Q) V (P Λ R)) // de 1	
4. P // de 2	
5. ¬Q V R // de 2	
6. ¬ (P Λ ¬Q) // de 3	
7. ¬ (P Λ R) // de 3	
8. ¬Q // de 5	8.1 R // de 5
9. ¬P // de 6	9.1 ¬(¬Q) // de 6
10.	10.1 Q // de 9.1
11.	11.1 ¬R // de 7

P 4 e 9

Q 8 e 10.1

R 8.1 e 11.1

(d) Resolução

((P→Q) ∧ (Q → R)) → ¬(¬R ∧ P)

¬(((P→Q) ∧ (Q→R)) → ¬(¬R ∧ P)) //Negar o teorema

¬(¬((P→Q) ∧ (Q→R)) V ¬(¬R ∧ P)) //Eliminação da implicação

¬(¬((¬PVQ) ∧ (Q→R)) V ¬(¬R ∧ P)) //Eliminação da implicação

¬(¬((¬PVQ) ∧ (¬QVR)) V ¬(¬R ∧ P)) //Eliminação da implicação

¬(¬((¬PVQ) ∧ (¬Q V R))) ∧ ¬(¬(¬R ∧ P)) //Distribuindo a negação

¬(¬((¬PVQ) ∧ (¬Q V R))) ∧ ¬(¬(¬R ∧ P)) //Eliminação da dupla negação

((¬PVQ) ∧ (¬Q V R)) ∧ ¬(¬(¬R ∧ P)) //Eliminação da dupla negação

((¬PVQ) ∧ (¬Q V R)) ∧ (¬R ∧ P) //Eliminação da dupla negação

(Q ∧ (¬Q V R)) ∧ ¬R // Eliminação do P pois P e ¬P

R ∧ ¬R // Eliminação do Q pois Q e ¬Q

□ //Eliminação do R pois R e ¬R

(d) Tableau

((P→Q) ∧ (Q → R)) → ¬(¬R ∧ P)

1. ¬ (((P→Q) ∧ (Q → R)) → ¬(¬R ∧ P)) //Negar o teorema	
2. ((P→Q) ∧ (Q → R)) // de 1	
3. ¬(¬(¬R ∧ P)) // de 1	
4. ¬(R ∧ P) // de 3	
5. (P→Q) // de 2	
6. (Q→R) // de 2	
7. ¬R // de 4	7.1 ¬P // de 4
8. ¬P // de 5	8.1 Q // de 5
9. ¬Q // de 6	9.1 R // de 5

(e) Resolução

$\exists x.\forall y.P(x, y) \rightarrow \forall y.\exists x.P(x, y)$

1) $\neg(\exists x.\forall y.P(x, y) \rightarrow \forall y.\exists x.P(x, y))$ //Negar o teorema
2) $\neg(\neg(\exists x.\forall y.P(x, y)) \lor (\forall y. \exists x.P(x, y)))$ //Eliminação da implicação
3) $\neg(\neg(\exists x.\forall y.P(x, y))) \land \neg(\forall y. \exists x.P(x, y))$ //Distribuindo a negação
4) $\exists x.\forall y.P(x, y) \land \neg(\forall y. \exists x.P(x, y))$ //Eliminação da dupla negação
5) $\exists x.\forall y.P(x, y) \land (\exists y.\neg\exists x.P(x, y))$ //Inverter a negação
6) $\exists x.\forall y.P(x, y) \land \exists y.\forall x.\neg P(x, y)$ //Inverter a negação
7) $\exists x_1.\forall y_1.P(x_1, y_1) \land \exists y_2.\forall x_2.\neg P(x_2, y_2)$ //Numerar variáveis
8) $\exists x_1.\forall y_1.(P(x_1, y_1) \land \exists y_2.\forall x_2.\neg P(x_2, y_2))$ // Mover quantificadores início
//Existe um pai(x1) para todo filho(y1), tal que x1 é pai de y1
//Não existe um filho(y2) para todo pai(x2), tal que x2 é pai de y2
9) $\exists x_1.\forall y_1.\exists y_2.\forall x_2.(P(x_1, y_1) \land\neg P(x_2, y_2))$ // Mover quantificadores início
10) $P(a, y_1) \land \neg P(x_2, f(y1))$ //Skolemização do passo 8
//Existe um pai(a) para todo filho(y1), tal que a é pai de y1
//Não existe um filho(f(y1)) para todo pai(x2), tal que x2 é pai de f(y1)
11) $\square$ //Eliminação do P pois P $P(a, y_1)$ e $\neg P(x_2, f(y_1))$

(e) Tableau

$\exists x.\forall y.P(x, y) \rightarrow \forall y.\exists x.P(x, y)$

1. $\neg (\exists x.\forall y.P(x, y) \rightarrow \forall y.\exists x.P(x, y))$ //Negar o teorema
2. $\exists x.\forall y.P(x, y)$ // de 1
3. $\neg(\forall y.\exists x.P(x, y))$ // de 1
4. $P(a,b)$ // de 2
5. $\neg P(a,b)$ // de 3

(f) Resolução

$(\exists x.P(x) \land \forall x.Q(x)) \rightarrow \exists x.(P(x) \land Q(x))$

$\neg((\exists x.P(x) \land \forall x.Q(x)) \rightarrow \exists x.(P(x) \land Q(x)))$ //Negar o teorema

$\neg(\neg(\exists x.P(x) \land \forall x.Q(x)) \lor \exists x.(P(x) \land Q(x)))$ //Eliminação da implicação

$\neg(\neg(\exists x.P(x) \land \forall x.Q(x))) \land \neg(\exists x.(P(x) \land Q(x)))$ //Distribuindo a negação

$(\exists x.P(x) \land \forall x.Q(x)) \land \neg(\exists x.(P(x) \land Q(x)))$ //Eliminação da dupla negação

$(\exists x.P(x) \land \forall x.Q(x)) \land (\forall x.\neg(P(x) \land Q(x)))$ //Inverter a negação

$(\exists x.P(x) \land \forall x.Q(x)) \land (\forall x.(\neg P(x) \lor \neg Q(x)))$ //Distribuindo a negação

$(\exists x_1.P(x_1) \land \forall x_2.Q(x_2)) \land (\forall x_3.(\neg P(x_3) \lor \neg Q(x_3)))$ //Numerar variáveis

$\exists x_1 \forall x_2.\forall x_3.(P(x_1) \land Q(x_2)) \land (\neg P(x_3) \lor \neg Q(x_3))$ //Mover quantificadores início

$(P(x_1) \land Q(x_2)) \land (\neg P(x_3) \lor \neg Q(x_3))$ // Eliminar quantificadores

$(P(a) \land Q(b)) \land (\neg P(c) \lor \neg Q(c))$ //Skolemização

$(Q(b)) \land (\neg Q(c))$ //Eliminação do P pois P(a) e ¬P(c)

□ //Eliminação do Q pois Q(b) e ¬Q(c)

(f) Tableau

$(\exists x.P(x) \land \forall x.Q(x)) \rightarrow \exists x.(P(x) \land Q(x))$

1. $\neg ((\exists x.P(x) \land \forall x.Q(x)) \rightarrow \exists x.(P(x) \land Q(x)))$ //Negar o teorema	
2. $(\exists x.P(x) \land \forall x.Q(x))$ // de 1	
3. $\neg(\exists x.(P(x) \land Q(x)))$ // de 1	
4. $\exists x.P(x)$ // de 2	
5. $\forall x.Q(x)$ // de 2	
6. $P(a)$ // de 4	
7. $Q(a)$ // de 5	
8. $\neg\exists x.(P(x)$ // de 3	8.1 $\neg\exists x.Q(x)$ // de 3
9. $\neg(P(a)$ // de 8	9.1 $\neg Q(a)$ // de 8.1

(g) Resolução

$\forall x.(P(x) \land Q(x)) \rightarrow (\forall x.P(x) \land \forall y.Q(y))$

$\neg(\forall x.(P(x) \land Q(x)) \rightarrow (\forall x.P(x) \land \forall y.Q(y)))$ //Negar o teorema
$\neg(\neg\forall x.(P(x) \land Q(x)) \lor (\forall x.P(x) \land \forall y.Q(y)))$ //Eliminação da implicação
$\neg(\neg\forall x.(P(x) \land Q(x))) \land \neg(\forall x.P(x) \land \forall y.Q(y)))$ //Distribuindo a negação
$\forall x.(P(x) \land Q(x)) \land \neg(\forall x.P(x) \land \forall y.Q(y))$ //Eliminação da dupla negação
$\forall x.(P(x) \land Q(x)) \land (\neg\forall x.P(x) \lor \neg\forall y.Q(y))$ //Distribuindo a negação
$\forall x.P(x) \land \forall x.Q(x) \land (\neg\forall x.P(x) \lor \neg\forall y.Q(y))$ //Distribuindo o ∀
$\forall x.P(x) \land \forall x.Q(x) \land (\exists x.\neg P(x) \lor \exists y.\neg Q(y))$ //Distribuindo o ∀
$\forall x_1.P(x_1) \land \forall x_2.Q(x_2) \land (\exists x_3.\neg P(x_3) \lor \exists y_4.\neg Q(y_4))$ //Numerar variáveis
$\forall x_1.\forall x_2.\exists x_3.\exists y_4.P(x_1)\land Q(x_2)\land(\neg P(x_3)\lor\neg Q(y_4))$ //Mover quantificadores início
$P(x_1)\land Q(x_2)\land(\neg P(x_3)\lor\neg Q(y_4))$ //Eliminar quantificadores
$P(a)\land Q(b)\land(\neg P(c)\lor\neg Q(d))$ //Skolemização
$(Q(b)) \land (\neg Q(d))$ //Eliminação do P pois P(a) e ¬P(c)
□ //Eliminação do Q pois Q(b) e ¬Q(d)

(g) Tableau
$\forall x.(P(x) \land Q(x)) \rightarrow (\forall x.P(x) \land \forall y.Q(y))$

1.	¬ (∀x.(P(x) Λ Q(x)) → (∀x.P(x) Λ ∀y.Q(y))) //Negar o teorema		
2.	∀x.(P(x) Λ Q(x)) // de 1		
3.	¬(∀x.P(x) Λ ∀y.Q(y)) // de 1		
4.	∀x.(P(x) // de 2		
5.	∀x.(Q(x) // de 2		
6.	P(a) // de 4		
7.	Q(a) // de 5		
8.	¬∀x.(P(x) // de 3	8.1	¬∀y.Q(y) // de 3
9.	¬(P(a) // de 8	9.1	¬Q(a) // de 8.1

(h) $(\forall x.P(x) \rightarrow \exists x.Q(x)) \rightarrow \exists x.(P(x) \rightarrow Q(x))$

$\neg((\forall x.P(x) \rightarrow \exists x.Q(x)) \rightarrow \exists x.(P(x) \rightarrow Q(x)))$ //Negar o teorema

$\neg(\neg(\forall x.P(x) \rightarrow \exists x.Q(x)) \vee \exists x.(P(x) \rightarrow Q(x)))$ //Eliminação da implicação

$\neg(\neg(\forall x.P(x) \rightarrow \exists x.Q(x))) \wedge \neg(\exists x.(P(x) \rightarrow Q(x))$ //Distribuindo a negação

$(\forall x.P(x) \rightarrow \exists x.Q(x)) \wedge \neg(\exists x.(P(x) \rightarrow Q(x))$ //Eliminação da dupla negação

$(\forall x.P(x) \rightarrow \exists x.Q(x)) \wedge \neg(\neg\exists x.(P(x) \vee Q(x))$ //Eliminação da implicação

$(\forall x.P(x) \rightarrow \exists x.Q(x)) \wedge \neg(\neg\exists x.(P(x)) \wedge \neg Q(x)$ //Distribuindo a negação

$(\forall x.P(x) \rightarrow \exists x.Q(x)) \wedge \exists x.(P(x) \wedge \neg Q(x)$ //Eliminação da dupla negação

$(\neg\forall x.P(x) \vee \exists x.Q(x)) \wedge \exists x.(P(x) \wedge \neg Q(x)$ //Eliminação da implicação

$(\exists x.\neg P(x) \vee \exists x.Q(x)) \wedge \exists x.(P(x) \wedge \neg Q(x)$//Inverter a negação

$(\exists x_1.\neg P(x_1) \vee \exists x_2.Q(x_2)) \wedge \exists x_3.(P(x_3) \wedge \neg Q(x_3)$ //Numerar variáveis

$\exists x_1.\exists x_2.\exists x_3.\neg P(x_1) \vee Q(x_2) \wedge (P(x_3) \wedge \neg Q(x_3)$ //Mover quantificadores início

$\neg P(x_1) \vee Q(x_2) \wedge (P(x_3) \wedge \neg Q(x_3)$ //Eliminar quantificadores

$\neg P(a) \vee Q(b) \wedge (P(c) \wedge \neg Q(d)$ //Skolemização

$Q(b) \wedge \neg Q(d)$ //Eliminação do P pois $\neg P(a)$ e $P(c)$

□ //Eliminação do Q pois $Q(b)$ e $\neg Q(d)$

(h) Tableau

$(\forall x.P(x) \rightarrow \exists x.Q(x)) \rightarrow \exists x.(P(x) \rightarrow Q(x))$

1. $\neg ((\forall x.P(x) \rightarrow \exists x.Q(x)) \rightarrow \exists x.(P(x) \rightarrow Q(x)))$ //Negar o teorema	
2. $(\forall x.P(x) \rightarrow \exists x.Q(x))$ // de 1	
3. $\neg(\exists x.(P(x) \rightarrow Q(x)))$ // de 1	
4. $\exists x.(P(x)$ // de 3	
5. $\neg Q(x)$ // de 3	
6. $P(a)$ // de 4	
7. $\exists x.(P(x)$ // de 3	
8. $\neg(\forall x.P(x))$ // de 2	8.1 $\exists x.Q(x)$ // de 2
9. $\neg P(a)$ // de 8	9.1 $Q(a)$ // de 8.1

5 e 9.1

6 e 9

(i) Resolução

$\forall x.(P(x) \land Q(x)) \rightarrow (\forall x.P(x) \land \forall y.Q(y))$

$\neg(\forall x.(P(x) \land Q(x)) \rightarrow (\forall x.P(x) \land \forall y.Q(y)))$ //Negar o teorema

$\neg(\neg(\forall x.(P(x) \land Q(x))) \lor (\forall x.P(x) \land \forall y.Q(y))$ //Eliminação da implicação

$\neg(\neg(\forall x.(P(x) \land Q(x))) \land \neg(\forall x.P(x) \land \forall y.Q(y))$ //Distribuindo a negação

$\forall x.(P(x) \land Q(x)) \land \neg(\forall x.P(x) \land \forall y.Q(y))$ //Eliminação da dupla negação

$\forall x.(P(x) \land Q(x)) \land (\neg\forall x.P(x) \lor \neg\forall y.Q(y))$ //Distribuindo a negação

$\forall x.(P(x) \land Q(x)) \land (\exists x.\neg P(x) \lor \neg\forall y.Q(y))$ //Inverter a negação

$\forall x.(P(x) \land Q(x)) \land (\exists x.\neg P(x) \lor \exists y.\neg Q(y))$ //Inverter a negação

$\forall x_1.(P(x_1) \land Q(x_1)) \land (\exists x_2.\neg P(x_2) \lor \exists y_3.\neg Q(y_3))$ //Numerar variáveis

$\forall x_1.\exists x_2.\exists y_3.(P(x_1) \land Q(x_1))\land(\neg P(x_2)\lor\neg Q(y_3))$ //Mover quantificadores início

$(P(x_1) \land Q(x_1)) \land (\neg P(x_2) \lor \neg Q(y_3))$ //Eliminar quantificadores

$(P(a) \land Q(a)) \land (\neg P(b) \lor \neg Q(c))$ //Skolemização

$Q(a) \land (\neg Q(c))$ //Eliminação do P pois $P(a)$ e $\neg P(b)$

$\square$ //Eliminação do Q pois $Q(a)$ e $\neg Q(c)$

(i) Tableau

$\forall x.(P(x) \land Q(x)) \rightarrow (\forall x.P(x) \land \forall y.Q(y))$

1. $\neg (\forall x.(P(x) \land Q(x)) \rightarrow (\forall x.P(x) \land \forall y.Q(y)))$ //Negar o teorema	
2. $(\forall x.(P(x) \land Q(x))$ // de 1	
3. $\neg(\forall x.P(x) \land \forall y.Q(y))$ // de 1	
4. $(\forall x.P(x)$ // de 2	
5. $(\forall x.Q(x)$ // de 2	
6. $P(a)$ // de 4	
7. $Q(a)$ // de 5	
8. $\neg(\forall x.P(x))$ // de 3	8.1 $\neg\forall y.Q(x)$ // de 3
9. $\neg P(a)$ // de 8	9.1 $\neg Q(a)$ // de 8.1

(j) $\exists x.(P(x) \land Q(x)) \rightarrow (\exists x.P(x) \land \exists x.Q(x))$

$\neg(\exists x.(P(x) \land Q(x)) \rightarrow (\exists x.P(x) \land \exists x.Q(x)))$ //Negar o teorema
$\neg(\neg(\exists x.(P(x) \land Q(x))) \lor (\exists x.P(x) \land \exists x.Q(x)))$ //Eliminação da implicação
$\neg(\neg(\exists x.(P(x) \land Q(x)))) \land \neg(\exists x.P(x) \land \exists x.Q(x))$ //Distribuindo a negação
$\exists x.(P(x) \land Q(x)) \land \neg(\exists x.P(x) \land \exists x.Q(x))$ //Eliminação da dupla negação
$\exists x.(P(x) \land Q(x)) \land (\neg(\exists x.P(x)) \lor \neg\exists x.Q(x))$ //Distribuindo a negação $\exists$
$\exists x.(P(x) \land Q(x)) \land (\forall x.\neg P(x) \lor \forall x.\neg Q(x))$ //Distribuindo a negação
$\exists x_1.(P(x_1) \land Q(x_1)) \land (\forall x_2.\neg P(x_2) \lor \forall x_3.\neg Q(x_3))$ //Numerar variáveis
$\exists x_1.\forall x_2.\forall x_3 (P(x_1) \land Q(x_1)) \land (\neg P(x_2) \lor \neg Q(x_3))$ //Mover quantificadores início
$(P(x_1) \land Q(x_1)) \land (\neg P(x_2) \lor \neg Q(x_3))$ //Eliminar quantificadores
$(P(a) \land Q(a)) \land (\neg P(b) \lor \neg Q(c))$ //Skolemização
$(Q(a)) \land (\neg Q(c))$ //Eliminação do P pois $P(a)$ e $\neg P(b)$
□ //Eliminação do Q pois $Q(a)$ e $\neg Q(c)$

(j) Tableau

$\exists x.(P(x) \land Q(x)) \rightarrow (\exists x.P(x) \land \exists x.Q(x))$

1. $\neg(\exists x.(P(x) \land Q(x)) \rightarrow (\exists x.P(x) \land \exists x.Q(x)))$ //Negar o teorema	
2. $\exists x.(P(x) \land Q(x))$ // de 1	
3. $\neg(\exists x.(P(x) \land Q(x))$ //de 1	
4. $\exists x.P(x)$ // de 2	
5. $P(a)$ // de 4	
6. $\exists x.Q(x)$ // de 2	
7. $Q(a)$ // de 6	
8. $\neg(\forall x.P(x))$ // de 3	8.1 $\neg\forall y.Q(x)$ // de 3
9. $\neg P(a)$ // de 8	9.1 $\neg Q(a)$ // de 8.1

5 e 9

7 e 9.1

(k) Resolução

$(\exists x.P(x) \wedge \forall x.Q(x)) \rightarrow \exists x.(P(x) \wedge Q(x))$

$\neg((\exists x.P(x) \wedge \forall x.Q(x)) \rightarrow \exists x.(P(x) \wedge Q(x)))$ //Negar o teorema

$\neg(\neg(\exists x.P(x) \wedge \forall x.Q(x)) \vee \exists x.(P(x) \wedge Q(x)))$//Eliminação da implicação

$\neg(\neg(\exists x.P(x) \wedge \forall x.Q(x))) \wedge \neg\exists x.(P(x) \wedge Q(x))$ //Distribuindo a negação

$(\exists x.P(x) \wedge \forall x.Q(x)) \wedge \neg\exists x.(P(x) \wedge Q(x))$ //Eliminação da dupla negação

$(\exists x.P(x) \wedge \forall x.Q(x)) \wedge \forall x.\neg(P(x) \wedge Q(x))$ // Inverter a negação

$(\exists x.P(x) \wedge \forall x.Q(x)) \wedge \forall x.(\neg P(x) \vee \neg Q(x))$ //Distribuindo a negação

$(\exists x_1.P(x_1) \wedge \forall x_2.Q(x_2)) \wedge \forall x_3.(\neg P(x_3) \vee \neg Q(x_3))$ //Numerar variáveis

$\exists x_1.\forall x_2.\forall x_3.(P(x_1) \wedge Q(x_2)) \wedge (\neg P(x_3) \vee \neg Q(x_3))$ //Mover quantificadores início

$(P(x_1) \wedge Q(x_2)) \vee (\neg P(x_3) \wedge \neg Q(x_3))$ //Eliminar quantificadores

$(P(a) \wedge Q(b)) \wedge (\neg P(c) \vee \neg Q(c))$ //Skolemização

$(Q(b)) \wedge (\neg Q(c))$ //Eliminação do P, pois P(a) e ¬P(c)

$\square$ //Eliminação do Q pois Q(b) e ¬Q(c)

(k) Tableau

$(\exists x.P(x) \wedge \forall x.Q(x)) \rightarrow \exists x.(P(x) \wedge Q(x))$

1. $\neg((\exists x.P(x) \wedge \forall x.Q(x)) \rightarrow \exists x.(P(x) \wedge Q(x)))$ //Negar o teorema	
2. $(\exists x.P(x) \wedge \forall x.Q(x))$ // de 1	
3. $\neg(\exists x.(P(x) \wedge Q(x)))$ //de 1	
4. $\exists x.P(x)$ // de 2	
5. $P(a)$ // de 4	
6. $\forall x.Q(x)$ // de 2	
7. $Q(a)$ // de 6	
8. $\neg(\exists x.P(x))$ // de 3	8.1 $\neg Q(x)$ // de 3
9. $\neg P(a)$ // de 8	9.1 $\neg Q(a)$ // de 8.1

5 e 9

7 e 9.1

(I) Resolução

$\forall x.(P(x) \vee Q(x)) \rightarrow (\exists x.P(x) \vee \forall x.Q(x))$

$\neg(\forall x.(P(x) \vee Q(x)) \rightarrow (\exists x.P(x) \vee \forall x.Q(x)))$ //Negar o teorema

$\neg(\neg\forall x.(P(x) \vee Q(x)) \vee (\exists x.P(x) \vee \forall x.Q(x)))$ //Eliminação da implicação

$\neg(\neg\forall x.(P(x) \vee Q(x))) \wedge \neg(\exists x.P(x) \vee \forall x.Q(x))$ //Distribuindo a negação

$\forall x.(P(x) \vee Q(x)) \wedge \neg(\exists x.P(x) \vee \forall x.Q(x))$ //Eliminação da dupla negação

$\forall x.(P(x) \vee Q(x)) \wedge \neg\exists x.P(x) \wedge \neg\forall x.Q(x)$ //Distribuindo a negação

$\forall x.(P(x) \vee Q(x)) \wedge \forall x.\neg P(x) \wedge \neg\forall x.Q(x)$ //Inverter a negação

$\forall x.(P(x) \vee Q(x)) \wedge \forall x.\neg P(x) \wedge \exists x.\neg Q(x)$ //Inverter a negação

$\forall x_1.(P(x_1) \vee Q(x_1)) \wedge \forall x_2.\neg P(x_2) \wedge \exists x_3.\neg Q(x_3)$ //Numerar variáveis

$\forall x_1.\forall x_2.\exists x_3.(P(x_1) \vee Q(x_1)) \wedge \neg P(x_2) \wedge \neg Q(x_3)$ //Mover quantificadores início

$(P(x_1) \vee Q(x_1)) \wedge \neg P(x_2) \wedge \neg Q(x_3)$ //Eliminar quantificadores

$(P(a) \vee Q(a)) \wedge \neg P(b) \wedge \neg Q(c)$ //Skolemização

$(Q(a)) \wedge (\neg Q(c))$ //Eliminação do P pois $P(a)$ e $\neg P(b)$

$\square$ //Eliminação do Q pois $Q(a)$ e $\neg Q(c)$

(I) Tableau

$\forall x.(P(x) \vee Q(x)) \rightarrow (\exists x.P(x) \vee \forall x.Q(x))$

1. $\neg(\forall x.(P(x) \vee Q(x)) \rightarrow (\exists x.P(x) \vee \forall x.Q(x)))$ //Negar o teorema	
2. $\forall x.(P(x) \vee Q(x))$ // de 1	
3. $\neg(\exists x.P(x) \vee \forall x.Q(x))$ //de 1	
4. $\neg\exists x.P(x)$ // de 3	
5. $\neg P(a)$ // de 4	
6. $\neg\forall x.Q(x)$ // de 3	
7. $\neg Q(a)$ // de 6	
8. $\forall x.P(x)$ // de 2	8.1 $Q(x)$ // de 2
9. $P(a)$ // de 8	9.1 $Q(a)$ // de 8.1

5 e 9

7 e 9.1

4. Teste de Einstein: Considerando as sentenças a seguir, modele o conhecimento em lógica, aplique o método da resolução e responda "quem tem um peixe como animal de estimação?"(Se quiser, use o Prolog)

(a) Há 5 casas de diferentes cores;

(b) Em cada casa mora uma pessoa de nacionalidade diferente;

(c) Nenhuma delas tem o mesmo animal, fuma o mesmo cigarro ou bebe

a mesma bebida;

(d) O inglês vive na casa vermelha;

(e) O sueco tem cachorros;

(f) O dinamarquês bebe chá;

(g) A casa verde fica do lado esquerdo da casa branca;

(h) O homem que vive na casa verde bebe café;

(i) O homem que fuma Malboro cria pássaros;

(j) O homem que vive na casa amarela fuma Hollywood;

(k) O homem que vive na casa do meio bebe leite;

(l) O norueguês vive na primeira casa;

(m) O homem que fuma Charme vive ao lado do que tem gatos;

(n) O homem que cria cavalos vive ao lado do que fuma Hollywood;

(o) O homem que fuma Free bebe cerveja;

(p) O alemão fuma Camel;

(q) O norueguês vive ao lado da casa azul;

(r) O homem que fuma Charme é vizinho do que bebe água

/*

Teste de Einstein: Considerando as sentenças a seguir, modele o conhecimento em lógica, aplique o método da resolução e responda "quem tem um peixe como animal de estimação?"(Se quiser, use o Prolog)

(a) Há 5 casas de diferentes cores;

(b) Em cada casa mora uma pessoa de nacionalidade diferente;

(c) Nenhuma delas tem o mesmo animal, fuma o mesmo cigarro ou bebe

a mesma bebida;

(d) O inglês vive na casa vermelha;

(e) O sueco tem cachorros;

(f) O dinamarquês bebe chá;

(g) A casa verde fica do lado esquerdo da casa branca;

(h) O homem que vive na casa verde bebe café;

(i) O homem que fuma Malboro cria pássaros;

(j) O homem que vive na casa amarela fuma Hollywood;

(k) O homem que vive na casa do meio bebe leite;

(l) O norueguês vive na primeira casa;

(m) O homem que fuma Charme vive ao lado do que tem gatos;

(n) O homem que cria cavalos vive ao lado do que fuma Hollywood;

(o) O homem que fuma Free bebe cerveja;

(p) O alemão fuma Camel;

(q) O norueguês vive ao lado da casa azul;

(r) O homem que fuma Charme é vizinho do que bebe água

Pergunta-se: ?

?- solucao(X).

Em Prolog o predicado select(X, L, R) seleciona o elemento X na lista L, e retira X de L formando o resto R. Ou seja, R tem todos os elementos de L, menos o elemento selecionado X. Isso pode ser usado para, dada a lista de possibilidades para cada atributo (por exemplo, cores de casa), retirar uma possibilidade da lista para gerar uma casa, depois usar o resto da lista para gerar as cores para as outras casas. Assim, duas casas nunca serão geradas com a mesma cor.

Gera as cores, é um wrapper par ao select passando a lista de cores disponíveis e recebendo o resto das cores disponíveis na última posição:

*/

```
gera_cor(casa(C, _, _, _, _), [C], []) :- !.
gera_cor(casa(C, _, _, _, _), Cores, Resto) :- select(C, Cores, Resto).
```

/*

Gera as nacionalidades, é um wrapper para o select passando a lista de nacionalidades disponíveis e recebendo o resto das nacionalidades disponíveis na última posição:

*/

```
gera_nacionalidade(casa(_, N, _, _, _), [N], []) :- !.
```

gera_nacionalidade(casa(_, N, _, _, _), Nacionalidades, Resto) :- select(N, Nacionalidades, Resto).

/*

Gera as bebidas, é um wrapper para o select passando a lista de bebidas disponíveis e recebendo o resto das bebidas disponíveis na última posição:

*/

gera_bebida(casa(_, _, B, _, _), [B], []) :- !.

gera_bebida(casa(_, _, B, _, _), Bebidas, Resto) :- select(B, Bebidas, Resto).

/*

Gera os cigarros, é um wrapper para o select passando a lista de cigarros disponíveis e recebendo o resto dos cigarros disponíveis na última posição:

*/

gera_cigarro(casa(_, _, _, C, _), [C], []) :- !.

gera_cigarro(casa(_, _, _, C, _), Cigarros, Resto) :- select(C, Cigarros, Resto).

/*

Gera os animais, é um wrapper para o select passando a lista de animais disponíveis e recebendo o resto dos animais disponíveis na última posição:

*/

gera_animal(casa(_, _, _, _, A), [A], []) :- !.

gera_animal(casa(_, _, _, _, A), Animais, Resto) :- select(A, Animais, Resto).

/*

Gera uma casa inteira e um conjunto de casas usando um mapeamento recursivo.

A estrutura "atr" guarda as listas com todos os atributos disponíveis, para simplificar o código.

*/

```
gera_casa(C, atr(Cs, Ns, Bs, Cigarros, As), atr(Cs2, Ns2, Bs2, Cigarros2, As2)) :-
    gera_cor(C, Cs, Cs2),
            gera_nacionalidade(C, Ns, Ns2),
    gera_bebida(C, Bs, Bs2),
            gera_cigarro(C, Cigarros, Cigarros2),
            gera_animal(C, As, As2).
```

```
gera_casas([], _) :- !.
gera_casas([C|Cs], Atribs) :-
	gera_casa(C, Atribs, Atribs2), gera_casas(Cs, Atribs2).

gera_solucao([C1, C2, C3, C4, C5]) :-
	Cores = [amarela,azul,branca,verde,vermelha],
	Nacionalidades = [alemao,dinamarques,ingles,noruegues,sueco],
	Bebidas = [agua,cafe,cerveja,cha,leite],
	Cigarros = [charme,free,hollywood,malboro,camel],
	Animais = [cachorro,cavalo,gato,passaro,peixe],
	gera_casas([C1, C2, C3, C4, C5], atr(Cores, Nacionalidades, Bebidas, Cigarros, Animais)).

/*
Para resolver o teste de Einstein é preciso estabelecer quando dois moradores são vizinhos, como é mencionado em várias dicas. Considerando a lista de soluções S, como gerada pelo predicado gera_solucao, cria-se predicados simples que testam (ou geram) moradores vizinhos na solução, inclusive separando vizinhos esquerdos de vizinhos direitos, pois uma dica especifica o lado.
*/
vizinho_esq(C1, C2, [C1,C2|_]).
vizinho_esq(C1, C2, [C3|T]) :- vizinho_esq(C1, C2, T).

vizinho_dir(C1, C2, [C2,C1|_]).
vizinho_dir(C1, C2, [C3|T]) :- vizinho_dir(C1, C2, T).

vizinho(C1, C2, S) :- vizinho_esq(C1, C2, S).
vizinho(C1, C2, S) :- vizinho_dir(C1, C2, S).

/*
Especifica das dicas
*/
solucao(S):-
		C1=casa(_,noruegues,_,_,_),
```

```
        C3=casa(_,_,leite,_,_),
        S=[C1,C2,C3,C4,C5], !,
    vizinho_esq(casa(verde,_,_,_,_),casa(branca,_,_,_,_), S),
    vizinho(casa(_,noruegues,_,_,_), casa(azul,_,_,_,_), S),
    vizinho(casa(_,_,_,charme,_),casa(_,_,_,_,gato), S),
    vizinho(casa(_,_,_,_,cavalo),casa(_,_,_,hollywood,_), S),
    vizinho(casa(_,_,_,charme,_),casa(_,_,agua,_,_), S),
        member(casa(vermelha,ingles,_,_,_), S),
        member(casa(_,sueco,_,_,cachorro), S),
        member(casa(_,dinamarques,cha,_,_), S),
        member(casa(verde,_,cafe,_,_), S),
        member(casa(_, _, _, malboro, passaro), S),
        member(casa(amarela,_,_,hollywood,_), S),
        member(casa(_,_,cerveja,free,_), S),
        member(casa(_,alemao,_,camel,_), S),
gera_solucao(S).
```

5. Escolha um dos testes de lógica postados pela Ana Carolina (moodle). Use

o prolog para desenvolver um programa que resolva o teste.

Exercício Lógica 1. Em arquivo em anexo. Exercício Lógica 4 em arquivo em anexo.

/*

Para ganhar pontos extras após tirar nota baixa na prova de Biologia, cinco alunos, se vestiram de plantas e fizeram uma apresentação oral sobre árvores e flores. A partir das dicas fornecidas, descubra o nome completo de cada aluno, bem como o tipo de árvore e flor que cada um falou.

(a) Érica falou sobre petúnias.

(b) O(A) aluno(a) de sobrenome Borges falou sobre pinheiros

(c) Jorge Costa falou sobre carvalhos.

(d) O(A) aluno(a) que falou sobre palmeiras também falou sobre cravos.

(e) O(A) aluno(a) de sobrenome Soares falou sobre dálias.

(f) O(A) aluno(a) de sobrenome Junqueira falou sobre ipês.

(g) Alex, cujo sobrenome não é Junqueira, falou sobre rosas.

(h) Lucas não falou sobre salgueiros.

Pergunta-se: ?

?- solucao(X).

Em Prolog o predicado select(X, L, R) seleciona o elemento X na lista L, e retira X de L formando o resto R. Ou seja, R tem todos os elementos de L, menos o elemento selecionado X. Isso pode ser usado para, dada a lista de possibilidades para cada atributo (por exemplo, nomes dos alunos), retirar uma possibilidade da lista para gerar um nome, depois usar o resto da lista para gerar os nomes para os outros alunos. Assim, dois alunos nunca serão geradas com o mesmo nome.

Gera as nome, é um wrapper par ao select passando a lista de nomes disponíveis e recebendo o resto dos nomes disponíveis na última posição:

*/

```
gera_nome(planta(C, _, _, _), [C], []) :- !.
gera_nome(planta(C, _, _, _), Nomes, Resto) :- select(C, Nomes, Resto).
```

/*

Gera os sobrenomes, é um wrapper para o select passando a lista de sobrenomes disponíveis e recebendo o resto das sobrenomes disponíveis na última posição:

*/

gera_sobrenome(planta(_, N, _, _), [N], []) :- !.

gera_sobrenome(planta(_, N, _, _), Sobrenomes, Resto) :- select(N, Sobrenomes, Resto).

/*

Gera as árvores, é um wrapper para o select passando a lista de árvores disponíveis e recebendo o resto das árvores disponíveis na última posição:

*/

gera_arvore(planta(_, _, B, _), [B], []) :- !.

gera_arvore(planta(_, _, B, _), Arvores, Resto) :- select(B, Arvores, Resto).

/*

Gera as flores, é um wrapper para o select passando a lista de flores disponíveis e recebendo o resto das flores disponíveis na última posição:

*/

gera_flor(planta(_, _, _, C), [C], []) :- !.

gera_flor(planta(_, _, _, C), Flores, Resto) :- select(C, Flores, Resto).

/*

Gera uma composicao inteira e um conjunto de alunos usando um mapeamento recursivo.

A estrutura "atr" guarda as listas com todos os atributos disponíveis, para simplificar o código.

*/

```
gera_planta(N, atr(Ns, Ss, As, Fs), atr(Ns2, Ss2, As2, Fs2)) :-
    gera_nome(N, Ns, Ns2),
            gera_sobrenome(N, Ss, Ss2),
            gera_arvore(N, As, As2),
            gera_flor(N, Fs, Fs2).


gera_plantas([], _) :- !.

gera_plantas([C|Cs], Atribs) :-
```

```
    gera_planta(C, Atribs, Atribs2), gera_plantas(Cs, Atribs2).

gera_sol([C1, C2, C3, C4, C5]) :-
    Nomes = [alex,erica,jorge,lucas,patricia],
    Sobrenomes = [borges,costa,junqueira,soares,vieira],
    Arvores = [carvalho,ipe,palmeira,pinheiro,salgueiro],
    Flores = [azaleia,cravo,dalia,petunia,rosa],
    gera_plantas([C1, C2, C3, C4, C5], atr(Nomes, Sobrenomes, Arvores, Flores)).

/*
Especifica das dicas
*/
solucao(S):-
      S=[C1,C2,C3,C4,C5], !,
            member(planta(erica, _, _, petunia), S),
            member(planta(_, borges, pinheiro, _), S),
            member(planta(jorge, costa, carvalho, _), S),
            member(planta(_, _, palmeira, cravo), S),
            member(planta(_, soares, _, dalia), S),
            member(planta(_, junqueira, ipe, _), S),
            member(planta(alex, A1, _, rosa), S), A1 \= junqueira,
            member(planta(lucas, _, _, B1), S), B1 \= salgueiro,
            gera_sol(S).
```