Lógica para Computação **Lista de Exercícios**

Fórmulas da Lógica Proposicional

1. Defina um pseudocódigo **recursivo** para a função `number_of_connectives`(A) que retorna a quantidade de conectivos da fórmula de entrada A. Por exemplo,

   `number_of_connectives`$(((\neg p) \rightarrow (\neg q))) = 3$.

   Em seguida, você deve usar o código disponível em

   `https://github.com/thiagoalvesifce/logicomp`

   e escrever um código para a função `number_of_connectives(formula)`.

2. Conforme a definição de fórmula da lógica proposicional, os conectivos binários devem ser escritos na forma infixa, ou seja, devem ser escritos entre duas fórmulas. Essa definição poderia ser modificada possibilitando escrever os conectivos na **notação polonesa**, conforme indicado pelas correspondências a seguir:

   - A fórmula A atômica corresponde à fórmula A na notação polonesa,
   - $(\neg A)$ corresponde à $\neg A$,
   - $(A \wedge B)$ corresponde à $\wedge AB$,
   - $(A \vee B)$ corresponde à $\vee AB$,
   - $(A \rightarrow B)$ corresponde à $\rightarrow AB$.

   Escreva as fórmulas a seguir utilizando a notação polonesa:

   (a) $\neg(p \rightarrow \neg q)$

   (b) $((\neg\neg p \vee q) \rightarrow (p \rightarrow q))$

3. Defina **recursivamente** um pseudocódigo para a função $atoms(A)$ que retorna o conjunto de todas as fórmulas atômicas que ocorrem em A. Por exemplo,

   $atoms(p \wedge \neg(p \rightarrow \neg q) \vee \neg q) = \{p, q\}$.

   Em seguida, você deve usar o repositório disponível em

   `https://github.com/thiagoalvesifce/logicomp`

   e escrever um código para definir a função `atoms(formula)`. Por exemplo,

   `atoms(Or(Not(And(Atom('p'), Atom('choveu_ontem'))), Atom('p')))`

   deve retornar um conjunto com as atômicas `Atom('p')` e `Atom('choveu_ontem')`.

4. Uma fórmula está na forma normal da negação (NNF - do inglês: negation normal form) se a negação só é aplicada diretamente nas atômicas e os outros únicos conectivos permitidos são a conjunção e a disjunção. Por exemplo, $((p \wedge (\neg(q \wedge r))) \wedge (\neg r)) \vee s$ **não está** na NNF e $((p \wedge ((\neg q) \wedge r)) \wedge (\neg r)) \vee s$ **está** na NNF. Defina um pseudocódigo para a função $is\_negation\_normal\_form(A)$ para verificar se A está na NNF. Em seguida, você deve usar o repositório disponível em

   `https://github.com/thiagoalvesifce/logicomp`

   e escrever um código para a função `is_negation_normal_form(formula)`.

5. Conforme a definição de fórmula da lógica proposicional, os conectivos binários devem ser escritos na forma infixa, ou seja, devem ser escritos entre duas fórmulas. Essa definição poderia ser modificada possibilitando escrever os conectivos na **notação polonesa**, conforme indicado pelas correspondências a seguir:

   - A fórmula A atômica corresponde à fórmula A na notação polonesa,
   - $(\neg A)$ corresponde à $\neg A$,
   - $(A \wedge B)$ corresponde à $\wedge AB$,
   - $(A \vee B)$ corresponde à $\vee AB$,
   - $(A \rightarrow B)$ corresponde à $\rightarrow AB$.

   As fórmulas a seguir estão na notação polonesa. Reescreva-as na notação convencional:

   (a) $\vee \rightarrow p\; q \rightarrow r \rightarrow \vee\; p\; q \neg\; s$

   (b) $\rightarrow \rightarrow p\; q \vee \rightarrow p\; q \rightarrow \neg\; r\; r$

6. Defina um pseudocódigo **recursivo** que substitui toda ocorrência da subfórmula B dentro da fórmula A pela fórmula C.

   Observe que $\mathit{substitution}(((p \wedge \neg q) \rightarrow r), (\neg q), (r \vee t))$ deve retornar a fórmula $((p \wedge (r \vee t)) \rightarrow r)$.

   Em seguida você deve usar repositório disponível em

   `https://github.com/thiagoalvesifce/logicomp`

   e escrever um código para a função

   `substitution(formula, old_subformula, new_subformula)`.

7. Além das convenções para omitir os parênteses das fórmulas, também temos outras formas para melhorar a legibilidade de fórmulas grandes. Por exemplo, a fórmula $p_1 \wedge p_2 \wedge p_3 \wedge p_4 \wedge p_5 \wedge p_6 \wedge p_7 \wedge p_8 \wedge p_9$ pode ser representada de maneira mais compacta como

$$\bigwedge_{i=1}^{9} p_i.$$

   Como outro exemplo, a fórmula $(\neg p_{1,1} \wedge \neg p_{1,2}) \vee (\neg p_{2,1} \wedge \neg p_{2,2}) \vee (\neg p_{3,1} \wedge \neg p_{3,2})$ pode ser apresentada de maneira mais compacta como

$$\bigvee_{i=1}^{3} \bigwedge_{j=1}^{2} \neg p_{i,j}.$$

   Dessa forma, essas notações são definidas de forma semelhante à notação de somatório. Nesta questão, a cada item a seguir você vai criar a fórmula completa via código a partir da fórmula compacta:

(a) $\bigwedge_{i=1}^{20} p\_i$.

(b) $\bigwedge_{i=1}^{n} p\_i$, em que n é entrada e um número inteiro positivo maior que 1.

(c) $\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} \neg p\_i\_j$, em que n e m são entradas e números inteiros maiores que 1.

(d) $\bigwedge_{i=1}^{n}((a_i \rightarrow (a_{i+1} \vee b_{i+1})) \wedge (b_i \rightarrow (a_{i+1} \vee b_{i+1})))$, em que n é entrada e um número inteiro positivo maior que 1.

(e) $(\bigwedge_{i=1}^{n}(p_i \vee q_i)) \rightarrow p_{n+1}$, em que n é entrada e um número inteiro positivo maior que 1.

(f) $\bigwedge_{i=1}^{n+1} \bigvee_{j=1}^{n} p_{i,j}$, em que n é entrada e número inteiro positivo maior que 1.

(g) $\bigvee_{i=1}^{n} \bigvee_{k=i+1}^{n+1} \bigvee_{j=1}^{n}(p_{i,j} \wedge p_{k,j})$, em que n é entrada e número inteiro positivo maior que 1.

Questões Extras

8. Defina **recursivamente** um pseudocódigo para a função $number\_of\_atoms(A)$ que retorna o número de ocorrências de atômicas em A. Por exemplo,

   $number\_of\_atoms((p \wedge \neg(p \rightarrow \neg q)) \vee \neg q) = 4$.

   Em seguida, você deve escrever uma definição para a função

   `number_of_atoms(formula)` no contexto do repositório disponível em

   `https://github.com/thiagoalvesifce/logicomp`

9. Responda os itens a seguir:

   (a) Um literal é uma atômica ou uma negação de uma atômica. Por exemplo, p e $(\neg q)$ **são** exemplos de literais, enquanto $(\neg(\neg p))$ e $((\neg p) \wedge q)$ **não são** literais. Defina um código para a função $is\_literal(A)$ para verificar se A é um literal. Em seguida, você deve usar o código disponível em

   `https://github.com/thiagoalvesifce/logicomp`

   e escrever uma definição da função `is_literal(formula)`.

   (b) Uma cláusula é uma disjunção de um ou mais literais. Por exemplo, $((p \vee (\neg q)) \vee r)$, q e $((\neg q) \vee (r \vee (\neg s)))$ **são** cláusulas, mas $\neg((p \vee (\neg q)) \vee r)$ e $(\neg(p \vee (\neg q)) \vee r)$ **não são** cláusulas. Defina um código para a função $is\_clause(A)$ para verificar se A é uma cláusula. Em seguida, você deve usar o código disponível em

   `https://github.com/thiagoalvesifce/logicomp` e escrever uma definição da função `is_clause(formula)`.

   (c) Uma fórmula está na formal normal conjuntiva (CNF - do inglês: conjunctive normal form) se ela é a conjunção de um ou mais cláusulas. Por exemplo, a fórmula $(p_1 \wedge (((\neg p_2) \vee p_3) \vee p_4)) \wedge ((\neg p_1) \vee ((\neg p_4) \vee p_1))$ **está** na CNF, enquanto $p \wedge ((\neg q) \vee ((\neg p) \wedge r))$ **não está** na CNF. Defina um código para a função

$is\_cnf(A)$ para verificar se A está na CNF. Em seguida, você deve usar o código disponível em

`https://github.com/thiagoalvesifce/logicomp` e escrever uma definição da função `is_cnf(formula)`.

10. Responda os itens a seguir:

    (a) Um termo é uma conjunção de um ou mais literais. Por exemplo, $((p \wedge (\neg q)) \wedge r)$ e p **são** termos, mas $\neg((p \wedge (\neg q)) \wedge r)$ e $(\neg(p \wedge (\neg q)) \wedge r)$ **não são** termos. Defina um código para a função $is\_term(A)$ para verificar se A é um termo. Em seguida, você deve usar o código disponível em

    `https://github.com/thiagoalvesifce/logicomp`

    e escrever uma definição da função `is_term(formula)`.

    (b) Uma fórmula está na formal normal disjuntiva (DNF - do inglês: disjunctive normal form) se ela é a disjunção um ou mais termos. Por exemplo, a fórmula $p_1 \vee (\neg p_2 \wedge p_3 \wedge p_4) \vee (\neg p_1 \wedge \neg p_4 \wedge p_1)$ **está** na DNF, enquanto $p \vee (\neg q \wedge (\neg p \vee r))$ **não está** na DNF. Defina um código para a função $is\_dnf(A)$ para verificar se A está na DNF. Em seguida, você deve usar o código disponível em

    `https://github.com/thiagoalvesifce/logicomp` e escrever uma definição da função `is_dnf(formula)`.

    (c) Uma fórmula A está na forma normal da negação decomposta (DNNF - do inglês: decomposable negation normal form) se está na NNF e para cada subfórmula $(A_1 \wedge A_2)$ de A temos que $atoms(A_1) \cap atoms(A_2) = \emptyset$. Por exemplo, $((a \vee \neg b) \wedge (c \vee d)) \vee ((a \vee b) \wedge (\neg c \vee \neg d))$ **está** na DNNF e $((a \vee \neg b) \wedge (\neg a \vee d)) \vee ((a \vee b) \wedge (\neg c \vee \neg d))$ **não está** na DNNF. Defina um código para a função $is\_decomposable\_negation\_normal\_form(A)$ para verificar se A está na DNNF. Em seguida, você deve usar o código disponível em

    `https://github.com/thiagoalvesifce/logicomp` e escrever uma definição da função

    `is_decomposable_negation_normal_form(formula)`.

11. Defina uma função recursiva `formula_height(formula)` que retorna a altura de `formula`, onde a altura é o maior número de conectivos entre o conectivo mais externo e as fórmulas atômicas. Por exemplo, para a fórmula $(p \rightarrow (q \wedge r)) \vee \neg s$, a altura é 3. Você deve definir a função no contexto do repositório:

    `https://github.com/thiagoalvesifce/logicomp`