Lógica para Computação **1ª Lista de Exercícios**

1. Defina **recursivamente** um código para a função $number\_of\_connectives(A)$ que retorna a quantidade de conectivos da fórmula de entrada A. Por exemplo,

   $number\_of\_connectives((\neg p) \to (\neg q)) = 3$.

   Para facilitar, você pode usar o código disponível em

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

   e escrever uma definição 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 $\vee AB$,
   - $(A \to B)$ corresponde à $\to AB$.

   Escreva as fórmulas a seguir utilizando a notação polonesa:

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

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

3. O rank $r(A)$ de uma fórmula A é definido por

$$r(A) = \begin{cases} 0, & \text{para } A \text{ atômica}, \\ max(r(A_1), r(A_2)) + 1, & \text{para } A = (A_1 \Box A_2) \text{ e } \Box \in \{\to, \wedge, \vee\}, \\ r(A_1) + 1, & \text{para } A = (\neg A_1). \end{cases}$$

   Demonstre por indução, se for verdadeira, ou dê um contra-exemplo, se for falsa, para a seguinte afirmação:

   para qualquer fórmula A temos que $r(A) \leq number\_of\_connectives(A)$.

4. Demonstre por indução, se for verdadeira, ou dê um contra-exemplo, se for falsa, para a seguinte afirmação:

   para toda fórmula A, $|subformulas(A)| = 2 \times number\_of\_connectives(A) + 1$.

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

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

   Defina um código para essa função. Para facilitar, você pode usar o código disponível em `https://github.com/thiagoalvesifce/logicomp` e escrever uma definição para a função `atoms(formula)`. Por exemplo,

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

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

6. 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, $(p \land \neg(q \land r) \land \neg r) \lor s$ **não está** na NNF e $(p \land (\neg q \land r) \land \neg r) \lor s$ **está** na NNF. Defina um código para a função $is\_negation\_normal\_form(A)$ para verificar se A está na NNF. Para facilitar, você pode usar o código disponível em

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

   e escrever uma definição para a função `is_negation_normal_form(formula)`.

Questões Extras

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

   $number\_of\_atoms((p \land \neg(p \to \neg q)) \lor \neg q) = 4$.

   Para facilitar, você pode usar o código disponível em

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

   e escrever uma definição para a função `number_of_atoms(formula)`.

8. Seja $number\_of\_binary\_connectives(A)$ uma função que retorna a quantidade de ocorrências de conectivos binários na fórmula A. Por exemplo,

   $number\_of\_binary\_connectives((p \to (\neg q))) = 1$.

   Demonstre por indução, se for verdadeira, ou dê um contra-exemplo, se for falsa, para a seguinte afirmação:

   para toda fórmula A, $number\_of\_atoms(A) = number\_of\_binary\_connectives(A) + 1$.

9. Qual a relação entre $number\_of\_atoms(A)$ e $number\_of\_connectives(A)$ para as fórmulas A **sem negação**? Justifique sua resposta com uma demonstração por indução.

10. 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 pós-fixa, conforme indicado pelas correspondências a seguir:

    - A atômica corresponde à A na notação pós-fixa
    - $(\neg A)$ corresponde à $A\neg$
    - $(A_1 \land A_2)$ corresponde à $A_1 A_2 \land$
    - $(A_1 \lor A_2)$ corresponde à $A_1 A_2 \lor$
    - $(A_1 \to A_2)$ corresponde à $A_1 A_2 \to$

Escreva as fórmulas a seguir na notação pós-fixa:

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

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

11. 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) $\rightarrow\ \rightarrow p\ q \vee \rightarrow p\ q \rightarrow \neg\ r\ r$

12. A substituição $substitution(A, B, C)$ de C no lugar de B em A é definida por

$$substitution(A) = \begin{cases} A, & \text{se } A \text{ é atômica} \\ & \text{e } A \neq B, \\ C, & \text{se } A = B, \\ (substitution(A_1, B, C) \Box substitution(A_2, B, C)), & \text{se } A = (A_1 \Box A_2) \\ & \text{com } \Box \in \{\rightarrow, \wedge, \vee\} \\ & \text{e } A \neq B, \\ (\neg substitution(A_1, B, C)), & \text{se } A = (\neg A_1) \\ & \text{e } A \neq B. \end{cases}$$

Observe que $substitution(((p \wedge \neg q) \rightarrow r), (\neg q), (r \vee t))$ é $((p \wedge (r \vee t)) \rightarrow r)$. **Defina um código** para essa função. Para facilitar, você pode usar o código disponível em `https://github.com/thiagoalvesifce/logicomp` e escrever uma definição para a função `substitution(formula, old_subformula, new_subformula)`.

13. 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. Para facilitar, você pode usar o código disponível em

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

e escrever uma definição para a função `is_literal(formula)`.

(b) Uma cláusula é uma disjunção de um ou mais literais. Por exemplo, $(p \vee \neg q \vee r)$ e q **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. Para facilitar, você pode usar o código disponível em `https://github.com/thiagoalvesifce/logicomp` e escrever uma definição para a 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. Para facilitar, você pode usar o código disponível em `https://github.com/thiagoalvesifce/logicomp` e escrever uma definição para a função `is_cnf(formula)`.

14. 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. Para facilitar, você pode usar o código disponível em

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

e escrever uma definição para a 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. Para facilitar, você pode usar o código disponível em `https://github.com/thiagoalvesifce/logicomp` e escrever uma definição para a 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 NNF. Para facilitar, você pode usar o código disponível em `https://github.com/thiagoalvesifce/logicomp` e escrever uma definição para a função

`is_decomposable_negation_normal_form(formula)`.