Fórmulas da Lógica Proposicional

Emannuel Levi de Assis Bezerra

Outubro de 2024

1 Questões Principais

Problema 1. Defina um código recursivo para a função number_of_connectives(A) que retorna a quantidade de conectivos da fórmula de entrada A.

Solução. Segue o código que implementa a função em python

```
def number_of_connectives(f: Formula) -> int:
    if isinstance(f, Not):
        return number_of_connectives(f.inner) + 1
    if isinstance(f, (Implies, And, Or)):
        return number_of_connectives(f.left) + number_of_connectives(f.right) + 1
    return 0
```

□

Problema 2(a). Escreva na notação polonesa a seguinte fórmula: $\neg(p \to \neg q)$

Solução.

$$\neg \to p \neg q$$

□

Problema 2(b). Escreva na notação polonesa a seguinte fórmula: $((\neg\neg p \vee q) \to (p \to q))$

Solução.

$$\to \vee \neg\neg pq \to pq$$

□

Problema 3. Defina recursivamente um código para a função atoms(A) que retorna o conjunto de todas as fórmulas atômicas que ocorrem em A.

Solução. Segue o código que implementa a função em python

```
def atoms(formula: Formula) -> set:
    atoms_set = set()
    if isinstance(formula, Atom):
        atoms_set.add(formula)
    if isinstance(formula, Not):
        atoms_set.update(atoms(formula.inner))
    if isinstance(formula, (Implies, And, Or)):
        atoms_set.update(atoms(formula.left))
        atoms_set.update(atoms(formula.right))
    return atoms_set
```

□

Problema 4. Defina um código para a função is_negation_normal_form(A) para verificar se A está na NNF (forma normal da negação).

Solução. Segue o código que implementa a função em python

```
def is_negation_normal_form(formula: Formula) -> bool:
    if isinstance(formula, Implies):
        return False
    if isinstance(formula, Not):
        return isinstance(formula.inner, Atom)
    if isinstance(formula, (And, Or)):
        return is_negation_normal_form(formula.left) and is_negation_normal_form(formula.right)
    return True
```

□

Problema 5(a). Reescreva na notação convencional a seguinte fórmula que está na notação polonesa: $\vee \to pq \to r \to \vee pq \neg s$

Solução.

$$(p \to q) \vee (r \to ((p \vee q) \to \neg s))$$

□

Problema 5(b). Reescreva na notação convencional a seguinte fórmula que está na notação polonesa: $\to\to pq\vee \to pq \to \neg rr$

Solução.

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

□

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

Solução. Segue o código que implementa a função em python

```
def substitution(formula: Formula, old_subformula: Formula, new_subformula: Formula) -> Formula:
    if formula == old_subformula:
        return new_subformula
    if isinstance(formula, Not):
        formula.inner = substitution(formula.inner, old_subformula, new_subformula)
    if isinstance(formula, (Implies, And, Or)):
        formula.left = substitution(formula.left, old_subformula, new_subformula)
        formula.right = substitution(formula.right, old_subformula, new_subformula)
    return formula
```

□

2 Questões Extras

Problema 7. Defina recursivamente um código para a função number_of_atoms(A) que retorna o número de ocorrências de atômicas em A.

Solução. Segue o código que implementa a função em python

```
def number_of_atoms(formula: Formula) -> int:
    if isinstance(formula, Not):
        return number_of_atoms(formula.inner)
    if isinstance(formula, (Implies, And, Or)):
        return number_of_atoms(formula.left) + number_of_atoms(formula.right)
    return 1
```

□

Problema 8(a). Defina um código para a função *is_literal(A)* para verificar se A é um literal.

Solução. Segue o código que implementa a função em python

```
def is_literal(formula: Formula) -> bool:
    return isinstance(formula, Atom) or (isinstance(formula, Not) and isinstance(formula.inner, Atom))
```

□

Problema 8(b). Defina um código para a função *is_clause(A)* para verificar se A é uma cláusula.

Solução. Segue o código que implementa a função em python

```
def is_clause(formula: Formula) -> bool:
    if isinstance(formula, Or):
        return is_clause(formula.left) and is_clause(formula.right)
    return is_literal(formula)
```

□

Problema 8(c). Defina um código para a função *is_cnf(A)* para verificar se A está na CNF (Forma Normal Conjuntiva).

Solução. Segue o código que implementa a função em python

```
def is_cnf(formula: Formula) -> bool:
    if isinstance(formula, And):
        condL = is_clause(formula.left) or is_cnf(formula.left)
        condR = is_clause(formula.right) or is_cnf(formula.right)
        return condL and condR
    return False
```

□

Problema 9(a). Defina um código para a função *is_term(A)* para verificar se A é um termo.

Solução. Segue o código que implementa a função em python

```
def is_term(formula: Formula) -> bool:
    if isinstance(formula, And):
        return is_clause(formula.left) and is_clause(formula.right)
    return is_literal(formula)
```

□

Problema 9(b). Defina um código para a função *is_dnf(A)* para verificar se A está na DNF (Forma Normal Dijuntiva).

Solução. Segue o código que implementa a função em python

```
def is_dnf(formula: Formula) -> bool:
    if isinstance(formula, Or):
        condL = is_term(formula.left) or is_dnf(formula.left)
        condR = is_term(formula.right) or is_dnf(formula.right)
        return condL and condR
    return False
```

□

Problema 9(c). Defina um código para a função *is_dnnf(A)* para verificar se A está na DNNF (Forma Normal da Negação Decomposta).

Solução. Segue o código que implementa a função em python

```
def is_dnnf(formula: Formula) -> bool:
    if is_negation_normal_form(formula):
        if isinstance(formula, And):
            if not (atoms(formula.left) & atoms(formula.right)):
                return True
            return is_dnnf(formula.left) and is_dnnf(formula.right)
        if isinstance(formula, Or):
            return is_dnnf(formula.left) and is_dnnf(formula.right)
    return False
```

□

Problema 10. Defina uma função recursiva *height(formula)* que retorna a altura de uma fórmula, onde a altura é o maior número de conectivos entre o conectivo mais externo e as fórmulas atômicas.

Solução. Segue o código que implementa a função em python

```
def height(formula: Formula) -> int:
    if isinstance(formula, Not):
        return 1 + height(formula.inner)
    if isinstance(formula, (And, Or, Implies)):
        return 1 + max(height(formula.left), height(formula.right))
    return 0
```

□