Lógica del Proposiciones

Paulo González

9 de agosto de 2019

1. Lógica proposicional

1.1. Sintáxis de lógica de proposiciones

En lógica de predicados, el alfabeto es la unión de:

- Un conjunto P de letras de proposiciones
- Un conjunto C de conectivas $\{\wedge, \vee, \neg, \supset, \equiv\}$
- Un conjunto V de de constantes que representano los valores de verdad

Definición Literal: Un literal es un átomo o un átomo negado.

1.2. Formula bien formada

- Si $p \in P$, entonces p es una fbf (átomo).
- Si φ es fbf, entonces $\neg\varphi$ es fbf.
- Si φ y ψ son fbf, entonces $(\varphi \wedge \psi), (\varphi \vee \psi), (\varphi \supset \psi)$ y $(\varphi \equiv \psi)$ son fbf.
- No hay más fbf.

Definición Teoría: Una teoría es un conjunto de clausulas o fórmulas bien formadas que deben ser verdaderas simultáneamente.

1.3. Reglas para la Satisfacibilidad

Un modelo o una evaluación σ satisface una fórmula (se escribe $\sigma \models \varphi$), de acuerdo a lo siguiente:

- $\sigma \models p$ ssi $\sigma(p) = 1$
- $\sigma \models \neg\varphi$ ssi $\sigma \not\models \varphi$
- $\sigma \models \varphi \wedge \psi$ ssi $\sigma \models \varphi \wedge \sigma \models \psi$
- $\sigma \models \varphi \vee \psi$ ssi $\sigma \models \varphi \vee \sigma \models \psi$
- $\sigma \models \varphi \supset \psi$ ssi $\sigma \not\models \varphi \vee \sigma \models \psi$
- $\sigma \models \varphi \equiv \psi$ ssi $\sigma \models (\varphi \supset \psi) \wedge \sigma \models (\psi \supset \varphi)$

1.4. Tabla de verdad

φ	ψ	$\neg\varphi$	$\neg\psi$	$\varphi \vee \psi$	$\varphi \wedge \psi$	$\varphi \supset \psi$	$\varphi \equiv \psi$
0	0	1	1	0	0	1	1
0	1	1	0	1	0	1	0
1	0	0	1	1	0	0	0
1	1	0	0	1	1	1	1

1.5. Programación del tipo booleano

En lo que sigue, se implementarán los operadores del tipo booleano de acuerdo a las reglas presentadas en la sección 1.3 para generar una tabla de verdad presentada en 1.4.

Implementación en C:

```c
#include<stdio.h>

int neg(int);
int disyuncion(int, int);
int conjuncion(int, int);
int implicancia(int, int);
int equivalencia(int, int);

#define verdadero 1
#define falso 0

void main(void){
    ...
    // instrucciones para imprimir tabla de verdad
    ...
}

int neg(int p){
  if(p == falso)
    return verdadero;
  return falso;
}

int disyuncion(int p, int q){
  if(p == falso)
    return q;
  return verdadero;
}

int conjuncion(int p, int q){
  if(p == falso)
    return falso;
  return q;
}

int implicancia(int p, int q){
  if(p == falso)
    return verdadero;
  return q;
}
```

```
int equivalencia(int p, int q){
  if(p == falso)
    return neg(q);
  return q;
}
```

Implementación Clase Booleanos en Python:

```
class Booleanos:
  def __init__(self):
    self.__verdadero = 1
    self.__falso = 0
  def neg(self,p):
    if( p == self.__falso):
      return self.__verdadero
    return self.__falso
  def disyuncion(self,p, q):
    if(p == self.__falso):
      return q
    return self.__verdadero
  def conjuncion(self,p, q):
    if(p == self.__falso):
      return self.__falso
    return q
  def implicancia(self,p, q):
    if(p == self.__falso):
      return self.__verdadero
    return q
  def equivalencia(self,p, q):
    if(p == self.__falso):
      return self.neg(q)
    return q
```

Implementación Clase Booleanos en Matlab:

```
classdef Booleanos < handle
   properties (SetAccess = private)
       verdadero;
       falso;
   end

   methods
       function  bool = Booleanos()
          bool.inicializar();
       end
       function res = neg(bool, p)
          if (p == bool.falso)
              res = bool.verdadero;
          else
              res = bool.falso;
          end
       end
       function res = disyuncion(bool, p, q)
```

```
            if (p == bool.falso)
                res = q;
            else
                res = bool.verdadero;
            end
        end
        function res = conjuncion(bool, p, q)
            if (p == bool.falso)
                res = bool.falso;
            else
                res = q;
            end
        end
        function res = implicancia(bool, p, q)
            if (p == bool.falso)
                res = bool.verdadero;
            else
                res = q;
            end
        end
        function res = equivalencia(bool, p, q)
            if (p == bool.falso)
                res = bool.neg(q);
            else
                res = q;
            end
        end
    end

    methods (Access = private)
        function inicializar(bool)
            bool.verdadero = 1;
            bool.falso = 0;
        end
    end
end
```