Fausto David Hernández Jasso
@FaustoJH
fausto.david.hernandez.jasso@ciencias.unam.mx

Lógica Computacional I
Introducción y Sintaxis de la Lógica de Proposiciones

2023-08-14

1 ¿Por qué estudiar lógica computacional?

Porque la lógica en las ciencias de la computación, en cierto sentido es el **cálculo de la computación** el cuál es el fundamento matemático para tratar la información y razonar acerca del comportamiento de programas.

2 ¿Por qué necesitamos la formalización del razonamiento correcto?

La lógica ha sido pieza clave para estructurar el pensamiento y el razonamiento:

- Dar un fundamento a las matemáticas.
- Eliminar errores del razonamiento.
- Encontrar una forma eficiente para llegar a una justificación de una conclusión, dada cierta información en forma de premisas.

3 Argumentos lógicos

3.1 Definición

Es una colección finita de afirmaciones *(proposiciones)* dividida en premisas y conclusión.

3.2 Acerca de las premisas y de la conclusión

Las premisas y la conclusión debe ser susceptibles de recibir un valor de verdad. El argumento lógico puede ser correcto o incorrecto.

3.3 Validez de un argumento

Un argumento es correcto o válido si suponiendo que sus premisas son ciertas, entonces necesariamente la conclusión también lo es.

3.4 Ejemplo

3.4.1 La isla de los caballeros y los bribones

En la isla de los caballeros y bribones sólo hay dos clases de habitantes, los caballeros que siempre dicen la verdad y los bribones que siempre mienten.
Un náufrago llega a la isla y encuentra dos habitantes: A y B. El habitante A afirma: Yo soy un bribón o B es un caballero. El acertijo consiste en averiguar cómo son A y B.

3.4.2 Lógica

$$p := \text{A es bribón}$$
$$q := \text{B es un caballero}$$
$$s := p \vee q$$

s es lo que dijo A.
Supongamos que $\mathcal{I}(p) = \top$
Así $\mathcal{I}(s) = \bot$
Entonces $\mathcal{I}(\neg s) = \top$
Como $\neg s \equiv \neg p \wedge \neg q$

Consecuentemente $\mathcal{I}(\neg p \wedge \neg q) = \top$
Lo que implica que se cumpla

$$\mathcal{I}(\neg p) = \top$$
$$\mathcal{I}(\neg q) = \top$$

Por lo tanto concluimos que A:

$$\mathcal{I}(p) = \bot$$

Por lo tanto hemos llegado a una contradicción, ya que A no puede ser bribón y no serlo al mismo tiempo.
Supongamos que $\mathcal{I}(p) = \bot$
Así $\mathcal{I}(s) = \top$
Como $s \equiv p \vee q$
Así tenemos que $\mathcal{I}(p) = \bot$ y $\mathcal{I}(q) = \top$
En éste caso A no es bribón y B es un caballero, entonces concluimos que:

- A es un caballero.
- B es un caballero.

3.5 Características de un argumento lógico

- Involucran individuos.
- Los individuos que involucran tienen propiedades.
- Una proposición es una oración que puede calificarse como verdadera o falsa y habla de las propiedades de los individuos.
- Se forman medinate proposiciones, clasificadas como premisas y conclusión del argumento.
- Las porposiciones pueden ser compuestas.
- Puede ser correcto o incorrecto.

4 Sistema lógico

Cualquier sistema lógico consta al menos de los siguientes tres componentes:

- Sintaxis.
- Semántica.
- Teoría de la prueba.

4.1 Sintaxis

Lenguaje formal que se utilizará como medio de expresión.

4.2 Semántica

Mecanismo que proporciona significado al lenguaje formal dado por la sintaxis.

4.3 Teoría de la prueba

Se encarga de decidir la correctud de un argumento lógico por medios puramente sintácticos.

4.4 Propiedades

4.4.1 Consistencia

No hay contradicciones

4.4.2 Correctud

Las reglas del sistema no pueden obtener una inferencia falsa a partir de una verdadera.

4.4.3 Completud

Todo lo verdadero es demostrable.

5 Lógica proposicional

5.1 Proposición

Es un enunciado que puede calificarse como verdadero o falso.

5.2 Lenguaje `PROP`

- Símbolos o variables proposicionales *(número infinito)*: $p_1, \ldots, p_n, \ldots$
- Constantes lógicas: $\top, \bot$
- Operadores lógicos: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
- Símbolos auxiliares: ()

El conjunto de expresiones o fórmulas atómicas, denotado como `ATOM` consta de:

- Variables proposicionales: $p_1, p_2, \ldots, p_n, \ldots$
- Constantes lógicas: $\top, \bot$

5.3 Expresiones en el lenguaje `PROP`

5.3.1 Definición recursiva

- Si $\varphi \in$ `ATOM` entonces $\varphi \in$ `PROP`.
- Si $\varphi \in$ `PROP` entonces $(\neg\varphi) \in$ `PROP`.
- Sí $\varphi, \psi \in$ `PROP` entonces $(\varphi \wedge \psi), (\varphi \vee \psi), (\varphi \rightarrow \psi), (\varphi \leftrightarrow \psi) \in$ `PROP`.
- Son todas.

5.3.2 Backus-Naur

$$\begin{aligned} VarP &::= p_1 \mid p_2 \mid \ \ldots \ \mid p_n \mid \ \ldots \\ \varphi, \psi &::= VarP \mid \top \mid \bot \mid \ (\neg\varphi) \ \mid \ (\varphi \wedge \psi) \ \mid \ (\varphi \vee \psi) \ \mid \ (\varphi \rightarrow \psi) \ \mid \ (\varphi \leftrightarrow \psi) \ \mid \end{aligned}$$

5.4 Precedencia y asociatividad de los operadores lógicos

Precedencia de mayor a menor

- $\neg$
- $\vee, \wedge$
- $\rightarrow$
- $\leftrightarrow$

Operadores asociativos:

- $\neg$
- $\vee, \wedge$
- $\leftrightarrow$

El operador $\rightarrow$ asocia hacia la derecha.
Ejemplificando lo anterior la expresión

$$\varphi_1 \rightarrow \varphi_2 \rightarrow \varphi_3$$

Queda asociada como:

$$\varphi_1 \rightarrow (\varphi_2 \rightarrow \varphi_3)$$

5.5 Eliminación de paréntesis innecesarios

Notemos que:

$$p \wedge q \rightarrow \neg q \vee s$$

es igual a

$$(p \wedge q) \rightarrow ((\neg q) \vee s)$$

Pero ¿Cómo podemos obtener la primera sí nos dan la última?
Simplemente debemos de ir quitando paréntesis guiándonos por la precedencia de operadores

$$\begin{array}{cr}
(p \wedge q) \rightarrow ((\neg q) \vee s) & \\
(p \wedge q) \rightarrow (\neg q \vee s) & \text{Eliminamos el paréntesis del operador } \neg \\
p \wedge q \rightarrow \neg q \vee s & \text{Eliminamos el paréntesis del operador } \vee \text{ y } \wedge
\end{array}$$

Veamos $(p \rightarrow q \wedge r) \leftrightarrow (s \vee (\neg t))$

$$\begin{array}{cr}
(p \rightarrow q \wedge r) \leftrightarrow (s \vee (\neg t)) & \\
(p \rightarrow q \wedge r) \leftrightarrow (s \vee \neg t) & \text{Eliminamos el paréntesis del operador } \neg \\
(p \rightarrow q \wedge r) \leftrightarrow s \vee \neg t & \text{Eliminamos el paréntesis del operador } \vee \\
p \rightarrow q \wedge r \leftrightarrow s \vee \neg t & \text{Eliminamos el paréntesis del operador } \rightarrow
\end{array}$$

Veamos $((p \vee q) \rightarrow r) \leftrightarrow ((\neg r) \rightarrow (\neg (p \vee q)))$

$$\begin{array}{cr}
((p \vee q) \rightarrow r) \leftrightarrow ((\neg r) \rightarrow (\neg (p \vee q))) & \\
((p \vee q) \rightarrow r) \leftrightarrow (\neg r \rightarrow (\neg (p \vee q))) & \text{Eliminamos el paréntesis del operador } \neg \\
((p \vee q) \rightarrow r) \leftrightarrow (\neg r \rightarrow \neg (p \vee q)) & \text{Eliminamos el paréntesis del operador } \neg \\
p \vee q \rightarrow r \leftrightarrow (\neg r \rightarrow \neg (p \vee q)) & \text{Eliminamos el paréntesis del operador } \rightarrow \\
p \vee q \rightarrow r \leftrightarrow \neg r \rightarrow \neg (p \vee q) & \text{Eliminamos el paréntesis del operador } \rightarrow
\end{array}$$

Veamos $\neg (((p \wedge (p \vee (\neg q))) \wedge q) \wedge p)$

$$\begin{array}{cr}
\neg (((p \wedge (p \vee (\neg q))) \wedge q) \wedge p) & \\
\neg (((p \wedge (p \vee \neg q)) \wedge q) \wedge p) & \text{Eliminamos el paréntesis del operador } \neg \\
\neg ((p \wedge (p \vee \neg q) \wedge q) \wedge p) & \text{Eliminamos el paréntesis del operador } \wedge \\
\neg (p \wedge (p \vee \neg q) \wedge q \wedge p) & \text{Eliminamos el paréntesis del operador } \wedge
\end{array}$$

Veamos $(\neg s) \rightarrow ((\neg t) \wedge \neg (p \vee q))$

$$\begin{array}{cr}
(\neg s) \rightarrow ((\neg t) \wedge \neg (p \vee q)) & \\
\neg s \rightarrow ((\neg t) \wedge \neg (p \vee q)) & \text{Eliminamos el paréntesis del operador } \neg \\
\neg s \rightarrow (\neg t \wedge \neg (p \vee q)) & \text{Eliminamos el paréntesis del operador } \neg \\
\neg s \rightarrow \neg t \wedge \neg (p \vee q) & \text{Eliminamos el paréntesis del operador } \wedge
\end{array}$$

5.5.1 Ejercicio

Elimina los paréntesis innecesarios de la expresión $(p \rightarrow (q \wedge (\neg q))) \rightarrow ((\neg p) \rightarrow p)$

6 Definiciones Recursivas y el Principio de Inducción

6.1 ¿En qué consiste?

En establecer las construcciones para generar elementos de una colección o de una estructura de datos.
La definición del lenguaje PROP es un ejemplo de ésta clase de definiciones, podemos identificar las construcciones básicas o atómicas y las construcciones recursivas.
Las definiciones recursivas también sirven para definir propiedades o funciones de una estructura mediante un análisis de casos, es decir de las distintas formas sintácticas que definen a los elementos de dicha estructura.

6.2 Ejemplo

Sea np una función que devuelve el número de paréntesis de una expresión lógica.
Definimos np como sigue:

$$
\begin{aligned}
np \ &: \ \mathtt{PROP} \to \mathbb{N} & \\
np(\varphi) &= 0 & \text{Si } \varphi \in \mathtt{ATOM} \\
np((\neg\varphi)) &= 2 + np(\varphi) & \text{Si } \varphi \in \mathtt{ATOM} \\
np((\varphi \star \psi)) &= 2 + np(\varphi) + np(\psi) & \text{con } \star \in \{\wedge, \vee, \to, \leftrightarrow\}
\end{aligned}
$$

6.3 Ejemplos de uso

Calculemos $np(((p \vee q) \to r) \leftrightarrow ((\neg r) \to (\neg (p \vee q))))$

$$np(((p \vee q) \to r) \leftrightarrow ((\neg r) \to (\neg (p \vee q)))) = \qquad \text{Aplicamos}$$