Lógica Computacional, 2025-2
Unidad 2: Lógica Proposicional

Semántica

Manuel Soto Romero Luis Fernando Loyola Cruz

23 de febrero de 2025
Facultad de Ciencias UNAM

En esta nota abordaremos uno de los conceptos fundamentales en el estudio de la lógica computacional: la semántica de la lógica proposicional. Hasta ahora, hemos explorado la sintaxis de este lenguaje formal, es decir, su estructura y las reglas que determinan cómo se pueden construir fórmulas bien formadas. Sin embargo, la lógica no solo se trata de manipular símbolos; también necesitamos entender su significado. Para ello, estudiaremos las interpretaciones de las fórmulas proposicionales y aprenderemos a determinar sus valores de verdad mediante tablas de verdad y funciones de interpretación. Además, analizaremos conceptos clave como la satisfacibilidad, las tautologías y las contradicciones, que nos permitirán formalizar el razonamiento lógico y evaluar la validez de argumentos. Todo esto nos servirá como base para aplicaciones más avanzadas, como la verificación formal y la resolución de problemas mediante solucionadores SAT.

1. Introducción

Ahora que conocemos la sintaxis de la lógica proposicional, nuestro siguiente objetivo es aprender a traducir enunciados expresados en lenguaje natural a este lenguaje formal. Este proceso se conoce como especificación formal. Una vez que logramos representar un enunciado de manera formal, podemos determinar su valor o valores de verdad. Esto nos permite, entre otras cosas, analizar si un argumento es correcto.

Para determinar el valor de verdad de una fórmula proposicional, es fundamental definir dos conceptos importantes:

- ⋆ Estado de una variable Se refiere al valor de verdad que puede tomar cada variable proposicional (verdadero o falso).
- ⋆ Significado de los conectivos lógicos Es necesario entender qué significa cada conectivo lógico, como la negación, disyunción, conjunción, condicional y bicondicional.

En esta nota exploraremos estos conceptos y presentaremos definiciones que serán muy útiles en los temas posteriores.

Comencemos analizando cómo se utilizan las variables proposicionales. Estas se emplean para representar una proposición expresada en lenguaje natural. Recordemos que una proposición es un enunciado que puede tener un valor de verdad, es decir, puede ser verdadero o falso.

Ejemplo 1

Dado el siguiente enunciado, analicemos cuántas proposiciones contiene y asignemos una variable proposicional distinta a cada una:

"Si hoy es viernes, entonces mañana es sábado; mañana es sábado, por lo tanto hoy es viernes."

1. **Identificación de proposiciones**
   - ⋆ **Hoy es viernes**
   - ⋆ **Mañana es sábado**

2. **Asignación de variables proposicionales**
   - ⋆ p Hoy es viernes
   - ⋆ q Mañana es sábado

Ejercicio 1

Dado el siguiente enunciado, identifica las proposiciones que lo componen y asigna una variable proposicional distinta a cada una:

"Si estudio, aprobaré el examen; además, si no apruebo el examen, entonces no podré ir de vacaciones."

Este es el primer paso hacia lo que llamamos especificación formal de una oración. Sin embargo, no basta con asignar un nombre a las proposiciones; también necesitamos representar la oración completa. Para ello, utilizamos los operadores lógicos. Además, es fundamental contar con una forma de indicar cuándo una proposición u oración es verdadera. Esta idea se formaliza mediante el concepto de interpretación, que nos permitirá definir de manera precisa la verdad de una proposición.

Para determinar si una proposición es falsa, utilizaremos el concepto de estado de una variable.

Definición 1. (Booleano)

El tipo de valores booleanos, denotado $\mathbb{B}$*, se define como el conjunto:*

$$\mathbb{B} = \{0, 1\}$$

donde 0 *representa el valor lógico* ***falso*** *y* 1 *el valor lógico* ***verdadero****.*

La definición formal de $\mathbb{B}$ establece que este tipo de dato es un conjunto con únicamente dos valores: 0 y 1, los cuales corresponden a los estados lógicos de falso y verdadero, respectivamente. Este modelo binario es la base de la lógica computacional y el álgebra booleana. En este contexto, 0 y 1 permiten representar y manipular proposiciones y decisiones lógicas de manera precisa y sistemática. Los booleanos

son fundamentales en programación, ya que son utilizados en estructuras de control (como condicionales) y en la representación de expresiones lógicas en sistemas computacionales.

Definición 2. (Estado de una Variable)

Un estado de las variables proposicionales es una función $\mathcal{I}$ definida como:

$$\mathcal{I} : P \to \mathbb{B}$$

donde P es el conjunto de todas las variables proposicionales.

La definición establece que un estado de las variables proposicionales es una función que asigna a cada variable proposicional un valor de verdad, ya sea 0 (falso) o 1 (verdadero). Este estado permite interpretar fórmulas proposicionales al asociarles valores específicos, haciendo posible evaluar su valor de verdad. Además es un concepto fundamental para definir el valor de verdad de fórmulas más complejas y para razonar formalmente sobre argumentos lógicos. Dado un conjunto de n variables proposicionales, es posible definir 2^n estados diferentes para dichas variables.

Ejemplo 2

En este ejemplo, utilizamos la notación $\mathcal{I}(x)$ para denotar el valor booleano asignado a una variable proposicional x bajo un estado $\mathcal{I}$. El resultado de $\mathcal{I}(x)$ puede ser 0 (falso) o 1 (verdadero). A continuación, se muestran las asignaciones para tres variables proposicionales:

$$\begin{aligned} \mathcal{I}(p) &= 0 \\ \mathcal{I}(q) &= 1 \\ \mathcal{I}(r) &= 0 \end{aligned}$$

Ejemplo 3

La notación anterior es común en lógica formal y sirve para representar estados específicos de las variables. Existen otras maneras de expresar lo mismo, por ejemplo:

1. Usar una lista explícita de pares ordenados que vinculan variables con valores booleanos:
$$\mathcal{I} = \{(p, 0), (q, 1), (r, 0)\}$$
2. Escribir las asignaciones como un mapeo explícito:
$$\mathcal{I} : \{p \mapsto 0, q \mapsto 1, r \mapsto 0\}$$
3. Representar las variables y sus valores en forma tabular:

Variable	Valor booleano
p	0
q	1
r	0

Cada estado define una función de interpretación, denotada como $\mathcal{I}^\star$, que actúa sobre las fórmulas proposicionales. Antes de introducir la definición formal de $\mathcal{I}^\star$, repasaremos el significado de los operadores lógicos, un tema que ya se abordó en el curso de Estructuras Discretas.

Para repasar el significado de los operadores lógicos, utilizaremos tablas de verdad. Estas tablas permiten representar, de manera estructurada, la interpretación de una fórmula proposicional en todos los estados posibles. Si una fórmula contiene más de una variable, debemos incluir todas las combinaciones posibles de sus valores. Aunque en este caso utilizaremos tablas de verdad para ejemplificar, evitaremos recurrir a ellas en el desarrollo posterior de este curso, a menos que sea estrictamente necesario.

2. Significado de los Conectivos Lógicos

Negación

La semántica de $\neg$ en lógica proposicional representa la negación, una operación unaria que invierte el valor de verdad de una proposición. Si una proposición es verdadera, su negación es falsa, y viceversa.

La negación $\neg\varphi$ se define mediante la siguiente tabla de verdad:

φ	$\neg\varphi$
0	1
1	0

Esto significa que si φ represena una afirmación como *Hoy llueve*, entonces $\neg\varphi$ representa *No es cierto que hoy llueve* o simplemente *Hoy no llueve*. En términos computacionales, la negación se usa para expresar condiciones opuestas y es fundamental en lógica booleana, programación y teoría de circuitos digitales.

Algunas formas comunes de expresar $\neg$ en español incluyen:

- ⋆ *No es cierto que* φ
- ⋆ *No ocurre que* φ
- ⋆ *Es falso que* φ
- ⋆ *No sucede que* φ
- ⋆ *No se cumple que* φ
- ⋆ *No* φ

Disyunción

La semántica de $\vee$ en lógica proposicional representa la disyunción, una operación binaria que se interpreta como *o*. Una disyunción $\varphi \vee \psi$ es verdadera si al menos una de las proposiciones involucradas es verdadera. Sólo es falsa cuando ambas son falsas simultáneamente.

La disyunción $\varphi \vee \psi$ se define mediante la siguiente tabla de verdad:

φ	ψ	$\varphi \vee \psi$
0	0	0
0	1	1
1	0	1
1	1	1

Esto significa que si φ representa *Está soleado* y ψ representa *Está nublado*, entonces la disyunción $\varphi \vee \psi$ expresa *Está soleado o está nublado*. En términos computacionales, la disyunción se usa en estructuras de decisión y en lógica booleanas para evaluar altenativas.

Algunas formas comunes de expresar $\varphi \vee \psi$ incluyen:

- φ *o* ψ
- *O bien* φ *o bien* ψ
- *Se cumple al menos una de* φ *o* ψ
- *Es suficiente con que* φ *o* ψ *sea cierto*
- *Basta con que* φ *o* ψ *suceda*
- *Al menos uno de* φ *o* ψ *es verdadero*

Conjunción

La semántica de $\wedge$ en lógica proposicional representa la conjunción, una operación binaria que se interpreta como *y*. Una conjunción $\varphi \wedge \psi$ es verdadera si y sólo si ambas proposiciones involucradas son verdaderas simultáneamente. En cualquier otro caso, es falsa.

La conjunción $\varphi \wedge \psi$ se define mediante la siguiente tabla de verdad:

φ	ψ	$\varphi \wedge \psi$
0	0	0
0	1	0
1	0	0
1	1	1

Esto significa que si φ representa *Está soleado* y ψ representa *Hace calor* entonces la conjunción $\varphi \wedge \psi$ expresa *Está soleado y hace calor*. En términos computacionales, la conjunción se usa en estructuras de decisión y en lógica booleana para evaluar condiciones simultáneas.

Algunas formas comunes de expresar $\varphi \wedge \psi$ en español incluyen:

- φ *y* ψ
- *Ambos* φ *y* ψ *son ciertos*
- *Se cumplen tanto* φ *como* ψ
- φ*, además* ψ
- *Sólo si* φ *y* ψ *ocurren juntos*
- *Es necesario que* φ *y* ψ *sean verdaderos al mismo tiempo*

Condicional

La semántica de $\rightarrow$ en lógica proposicional representa el condicional material[1], una operación binaria que se interpreta como *si... entonces...*". Una implicación $\varphi \rightarrow \psi$ es falsa únicamente cuando el antecedente φ es verdadero y el consecuente ψ es falso; en todos los demás casos, la expresión es verdadera.

El condicional $\varphi \rightarrow \psi$ se define mediante la siguiente tabla de verdad:

φ	ψ	$\varphi \rightarrow \psi$
0	0	1
0	1	1
1	0	0
1	1	1

Esto significa que si φ representa *Llueve* y ψ representa *El suelo está mojado* entonces la expresión $\varphi \rightarrow \psi$ se interpreta como *Si llueve, entonces el suelo está mojado*. Sin embargo, en lógica proposicional, esta expresión no implica causalidad, sino sólo una relación condicional. Es decir, si no llueve, la impicación sigue siendo verdader independientemente de si el suelo está mojado o no ψ.

En programación y razonamiento computacional, el condicional se usa en estructuras de decisión como `if`, donde una acción se ejecuta si una condición se cumple.

Algunas formas comunes de expresar $\varphi \rightarrow \psi$ en español incluyen:

- *Si φ, entonces ψ*
- *Siempre que φ, ocurre ψ*
- *Cuando φ, sucede ψ*
- *ψ si φ*
- *ψ en caso de que φ*
- *Es suficiente que φ para que ψ*
- *Siendo φ, se decuce ψ*
- *La verdad de φ garantiza la verdad de ψ*
- *ψ es una condición necesaria para φ*
- *Para que φ se cumpla, es necesario que ψ*

[1] El **condicional material** es la implicación lógica representada por $\varphi \rightarrow \psi$, que se considera verdadera en todos los casos excepto cuando φ es verdadera y ψ es falsa. Se llama *material* porque su verdad depende únicamente de los valores de verdad de sus componentes, sin implicar una relación causal o necesaria. Existen otros tipos de condicionales, como el **estricto** (basado en necesidad lógica), el **subjuntivo o contrafactual** (que considera escenarios hipotéticos) y el **indicativo** (basado en la información disponible en el mundo real).

Bicondicional

La semántica de $\leftrightarrow$ en lógica proposicional representa el bicondicional, una operación binaria que se interpreta como *si y sólo si*. La expresión $\varphi \leftrightarrow \psi$ es verdadera cuando ambos operandos tiene el mismo valor de verdad, es decir, cuando φ y ψ son ambos verdaderos o ambos falsos. En los casos en los que φ y ψ difieren en su valor de verdad, la expresión es falsa.

El bicondicional $\varphi \leftrightarrow \psi$ se define mediante la siguiente tabla de verdad:

φ	ψ	$\varphi \leftrightarrow \psi$
0	0	1
0	1	0
1	0	0
1	1	1

Esto significa que si φ representa *El interruptor está encendido* y ψ representa *La luz está encendida*, entonces la expresión $\varphi \leftrightarrow \psi$ se interpreta como *El interruptor está encendido si y sólo si la luz está encendida*. Es decir, ambas afirmaciones deben ser verdaderas o falsas simultáneamente para que la expresión sea verdadera.

En programación y razonamiento computacional, el bicondicional es útil en comparaciones lógicas y verificaciones de equivalencia, ya que establece que dos condiciones deben ser idénticas en términos de verdad para que la relación se mantenga.

Algunas formas comunes de expresar $\varphi \leftrightarrow \psi$ en español incluyen:

- φ *si y sólo si* ψ
- *Para que* φ*, es necesario y suficiente* ψ
- φ *es equivalente a* ψ
- φ *exactamente cuando* ψ
- *Es verdad que* φ *cuando y sólo cuando* ψ *también lo es*
- *Se cumple* φ*, si y sólo si, se cumple* ψ
- *No hay diferencia entre que* φ *y que* ψ

Observación 1

En términos del operador condicional, la diferencia entre necesidad y suficiencia se puede entender de la siguiente manera:

- Decimos que una proposición φ es **suficiente** para otra proposición ψ si cuando φ es verdadera, entonces ψ también debe ser verdadera. Esto se expresa lógicamente como

  $$\varphi \to \psi$$

  En español, esto se traduce como
  - *Si* φ*, entonces* ψ
  - φ *es suficiente para* ψ

⋆ Decimos que una proposición ψ es **necesaria** para otra proposición φ si φ no puede ser verdadera sin que ψ también lo sea. Esto significa que la ausencia de ψ garantiza la ausencia de φ. Se puede expresar en términos del condicional como:

$$\varphi \to \psi$$

(es la misma expresión, pero interpretada de otra forma).
En español, esto se traduce como:
- ∗ *Para que φ ocurra, es necesario que ψ ocurra*
- ∗ *ψ es una condición necesaria para φ*

Por otro lado cuando tenemos un bicondicional, es decir $\varphi \leftrightarrow \psi$, significa que φ y ψ son **necesarios y suficientes** la una para la otra. En otras palabras:

⋆ $\varphi \to \psi$ (suficiencia): φ es suficiente para ψ
⋆ $\psi \to \varphi$ (necesidad): ψ es necesario para φ
⋆ Juntas, forman $\varphi \leftrightarrow \psi$

Ejemplo 4

Solo me siento a comer si mi papá cocina o si hay enchiladas.
1. **Identificación de proposiciones:**

⋆ Me siento a comer.
⋆ Mi papá cocina.
⋆ Hay enchiladas.

2. **Asignación de variables:**
⋆ p: Me siento a comer.
⋆ q: Mi papá cocina.
⋆ r: Hay enchiladas.

3. **Traducción:**

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

Ejemplo 5

Solo me siento a comer si mi papá cocina o si hay enchiladas.
1. **Identificación de proposiciones:**

⋆ Me siento a comer.
⋆ Mi papá cocina.
⋆ Hay enchiladas.

2. **Asignación de variables:**
⋆ p: Me siento a comer.
⋆ q: Mi papá cocina.
⋆ r: Hay enchiladas.

3. **Traducción:**

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

Ejemplo 6

Los autos no vuelan y tienen ruedas es lo mismo que los perros no usan sombreros.

1. **Identificación de proposiciones:**

   ⋆ Los autos vuelan.
   ⋆ Los autos tienen ruedas.
   ⋆ Los perros usan sombreros.

2. **Asignación de variables:**
   ⋆ r: Los autos vuelan.
   ⋆ s: Los autos tienen ruedas.
   ⋆ t: Los perros usan sombreros.

3. **Traducción:**

$$(\neg r \wedge s) \leftrightarrow \neg t$$

Ejercicio 2

Realiza la especificación formal de los siguientes enunciados:

1. Si hay clases y está lloviendo, entonces llevo paraguas, pero si no está lloviendo, llevo una gorra.
2. Solo si termino mi tarea o me siento motivado, estudiaré para el examen. Además, si no estudio, entonces no aprobaré.
3. Los trenes son rápidos y eficientes si y solo si no hay retrasos y las vías están en buen estado.

3. Funciones de Interpretación

Las funciones de interpretación ($\mathcal{I}$) son esenciales para un análisis formal y práctico de la lógica proposicional, ya que permiten asignar valores de verdad a las variables de manera compacta y eficiente. A diferencia de las tablas de verdad, que crecen exponencialmente con el número de variables (2^n renglones), las funciones de interpretación son escalables y más adecuadas para estructuras complejas. Esto las hace indispensables en aplicaciones como verificación formal, resolución-SAT y razonamiento automatizado, donde el manejo abstracto de fórmulas lógicas es de gran importancia.

Definición 3. (Función de Interpretación para LProp)

Dado un estado de las variables $\mathcal{I} : P \to \mathbb{B}$, definimos la interpretación de las fórmulas con respecto a $\mathcal{I}$ como la función $\mathcal{I}^\star : \mathsf{LProp} \to \mathbb{B}$.

Para dar la definición completa de $\mathcal{I}^\star$, hagamos un análisis de los casos posibles comparándolos con su tabla de verdad.

Fórmulas Atómicas

Para las fórmulas atómicas, la definición se sigue naturalmente como sigue:

$$\mathcal{I}^\star(\bot) = 0$$

$$\mathcal{I}^\star(\top) = 1$$

$$\mathcal{I}^\star(p) = \mathcal{I}(p), \quad \text{para } p \in P$$

Negación

Invierte el valor de verdad.

φ	$\neg\varphi$
0	1
1	0

$$\mathcal{I}^\star(\neg\varphi) = \begin{cases} 0 & \text{si } \mathcal{I}^\star(\varphi) = 1, \\ 1 & \text{si } \mathcal{I}^\star(\varphi) = 0 \end{cases}$$

Disyunción

Es verdadera si al menos una de las fórmulas lo es.

φ	ψ	$\varphi \vee \psi$
0	0	0
0	1	1
1	0	1
1	1	1

$$\mathcal{I}^\star(\varphi \vee \psi) = \begin{cases} 1 & \text{si } \mathcal{I}^\star(\varphi) = 1 \text{ o } \mathcal{I}^\star(\psi) = 1, \\ 0 & \text{si } \mathcal{I}^\star(\varphi) = 0 \text{ y } \mathcal{I}^\star(\psi) = 0 \end{cases}$$

Conjunción

Es verdadera solo si ambas fórmulas lo son.

φ	ψ	$\varphi \wedge \psi$
0	0	0
0	1	0
1	0	0
1	1	1

$$\mathcal{I}^\star(\varphi \wedge \psi) = \begin{cases} 1 & \text{si } \mathcal{I}^\star(\varphi) = 1 \text{ y } \mathcal{I}^\star(\psi) = 1, \\ 0 & \text{si } \mathcal{I}^\star(\varphi) = 0 \text{ o } \mathcal{I}^\star(\psi) = 0 \end{cases}$$

Condicional

Es falsa únicamente si el antecedente es verdadero y el consecuente es falso.

φ	ψ	$\varphi \to \psi$
0	0	1
0	1	1
1	0	0
1	1	1

$$\mathcal{I}^\star(\varphi \to \psi) = \begin{cases} 0 & \text{si } \mathcal{I}^\star(\varphi) = 1 \text{ y } \mathcal{I}^\star(\psi) = 0, \\ 1 & \text{si } \mathcal{I}^\star(\varphi) = 0 \text{ o } \mathcal{I}^\star(\psi) = 1 \end{cases}$$

Bicondicional

Es verdadera si ambas fórmulas tienen el mismo valor de verdad.

φ	ψ	$\varphi \leftrightarrow \psi$
0	0	1
0	1	0
1	0	0
1	1	1

$$\mathcal{I}^\star(\varphi \leftrightarrow \psi) = \begin{cases} 1 & \text{si } \mathcal{I}^\star(\varphi) = \mathcal{I}^\star(\psi), \\ 0 & \text{si } \mathcal{I}^\star(\varphi) \neq \mathcal{I}^\star(\psi) \end{cases}$$

Observación 2

En la definición de las funciones de interpretación, las palabras *y* y *o* aparecen explícitamente en el metalenguaje para describir las condiciones bajo las cuales una fórmula compuesta toma un valor de verdad. Estas palabras no forman parte del lenguaje formal de la lógica proposicional, sino que son elementos del metalenguaje natural utilizado para expresar la semántica de los conectivos lógicos.

Por ejemplo, en la interpretación de la conjunción $\varphi \wedge \psi$, la función de interpretación $\mathcal{I}^\star$ la define como verdadera si $\mathcal{I}^\star(\varphi) = 1$ y $\mathcal{I}^\star(\psi) = 1$. Aquí, la palabra *y* en el metalenguaje describe la necesidad de que ambas condiciones se satisfagan simultáneamente, reflejando la naturaleza del operador $\wedge$. De manera similar, en la interpretación de la disyunción $\varphi \vee \psi$, se establece que es verdadera si $\mathcal{I}^\star(\varphi) = 1$ o $\mathcal{I}^\star(\psi) = 1$, donde la palabra *o* en el metalenguaje captura la idea de que basta con que una de las condiciones se cumpla para que toda la expresión sea verdadera.

El uso de metalenguajes es una práctica común en Ciencias de la Computación, ya que permite describir formalmente estructuras y reglas de sistemas formales sin ambigüedad. En teoría de lenguajes de programación, por ejemplo, se utilizan metalenguajes como BNF (Backus-Naur Form) o EBNF (Extended BNF) para especificar la sintaxis de los lenguajes de programación. Así, cuando se define la estructura de una expresión condicional en un lenguaje de programación, el metalenguaje permite describir de manera precisa qué elementos componen la instrucción sin depender de una implementación específica.

Otro ejemplo se encuentra en los metadatos, que en diversos campos del conocimiento proporcionan información sobre los datos en sí. En bases de datos y sistemas de información, los metadatos describen la estructura de los datos almacenados, mientras que en el desarrollo web, lenguajes como HTML incluyen etiquetas `<meta>` que contienen información sobre el contenido de una página. En cada caso, el metalenguaje se utiliza para describir y estructurar información sobre el lenguaje objeto sin ser parte directa de los datos procesados.

Dado un estado de las variables $\mathcal{I}$, la interpretación $\mathcal{I}^\star$ está determinada de manera única, por lo que a partir de ahora utilizaremos simplemente $\mathcal{I}$ en lugar de $\mathcal{I}^\star$. Tanto $\mathcal{I}$ como $\mathcal{I}^\star$ corresponden a los renglones de las tablas de verdad y permiten interpretar fórmulas más complejas.

Ejemplo 7

Para la fórmula $p \wedge q$ tenemos:

Tabla de verdad

	p	q	$p \wedge q$
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

Estados

$$\begin{aligned} \mathcal{I}_1 &= \{(p,0),(q,0)\} \\ \mathcal{I}_2 &= \{(p,0),(q,1)\} \\ \mathcal{I}_3 &= \{(p,1),(q,0)\} \\ \mathcal{I}_4 &= \{(p,1),(q,1)\} \end{aligned}$$

Interpretaciones

$$\begin{aligned} \mathcal{I}_1(p \wedge q) &= 0 \\ \mathcal{I}_2(p \wedge q) &= 0 \\ \mathcal{I}_3(p \wedge q) &= 0 \\ \mathcal{I}_4(p \wedge q) &= 1 \end{aligned}$$

4. Satisfacibilidad

Dada una fórmula φ, podemos preguntarnos:

¿Cuántas interpretaciones hacen verdadera a φ?

Las respuestas a esta pregunta nos conducen a distintas definiciones importantes.

Definición 4. (Tautología)

*Una fórmula φ es **tautología** o **fórmula válida** si $\mathcal{I}(\varphi) = 1$ para toda interpretación $\mathcal{I}$. Esto se denota como $\vDash \varphi$.*

Ejemplo 8

Decimos que $p \vee \neg p$ es una tautología pues $\mathcal{I}(p \vee \neg p) = 1$ para toda $\mathcal{I}$.

$$\vDash p \vee \neg p$$

Definición 5. (Fórmula Satisfacible)

*Una fórmula φ es **satisfacible** si existe al menos una interpretación $\mathcal{I}$ tal que $\mathcal{I}(\varphi) = 1$. En este caso, decimos que φ es verdadera bajo $\mathcal{I}$, o que $\mathcal{I}$ es un modelo de φ, y escribimos $\mathcal{I} \vDash \varphi$.*

*Si existe una interpretación $\mathcal{I}$ donde $\mathcal{I}(\varphi) = 0$, afirmamos que φ es **insatisfacible** bajo dicha interpretación o que $\mathcal{I}$ no es un modelo de φ, lo cual se denota como $\mathcal{I} \nvDash \varphi$.*

Ejemplo 9

Decimos que $p \to \neg p$ es satisfacible pues si $\mathcal{I}_1(p) = 0$, entonces $\mathcal{I}_1(p \to \neg p) = 1$.

$$\mathcal{I}_1 \vDash p \to \neg p$$

No es una tautología pues si $\mathcal{I}_2(p) = 1$, es insatisfacible.

$$\mathcal{I}_2 \nvDash p \to \neg p$$

Definición 6. (Contradicción)

Una fórmula φ es una ***contradicción*** *o* ***fórmula no satisfacible*** *si $\mathcal{I}(\varphi) = 0$ para toda interpretación $\mathcal{I}$. Esto se representa como $\nvDash \varphi$.*

Ejemplo 10

Decimos que la fórmula $p \wedge \neg p$ es una contradicción pues $\mathcal{I}(p \wedge \neg p) = 0$ para toda $\mathcal{I}$.

Ejercicio 3

Determina, usando funciones de interpretación, si las siguientes fórmulas son tautologías, satisfacibles, insatisfacibles o contradicciones. En ciertos casos, una fórmula puede cumplir con más de una de estas categorías.

1. $(p \vee \neg p) \wedge (q \vee \neg q)$
2. $p \wedge (q \to \neg p)$
3. $(p \to q) \wedge (\neg q \to \neg p)$
4. $(p \wedge q) \vee (\neg p \wedge \neg q)$
5. $(p \to q) \wedge (q \to r) \wedge (\neg p \vee \neg r)$

Si ahora consideramos un conjunto de fórmulas Γ, definimos:

Definición 7. (Conjunto de Fórmulas Satisfacible)

Un conjunto de fórmulas Γ es ***satisfacible*** *si existe al menos una interpretación $\mathcal{I}$ tal que $\mathcal{I}(\varphi) = 1$ para toda $\varphi \in \Gamma$.* A veces, esta propiedad se denota, por simplicidad, como $\mathcal{I}(\Gamma) = 1$.

Ejemplo 11

$\Gamma_1 = \{p \vee \neg p, p \to \neg p\}$ es satisfacible

Definición 8. (Conjunto de Fórmulas Insatisfacible)

Un conjunto de fórmulas Γ es ***insatisfacible*** *si no existe ninguna interpretación $\mathcal{I}$ que satisfaga a todas las fórmulas de Γ, es decir, no hay $\mathcal{I}$ tal que $\mathcal{I}(\varphi) = 1$ para cada $\varphi \in \Gamma$.*

Ejemplo 12

$\Gamma_2 = \{p \vee \neg p, p \wedge \neg p\}$ es insatisfacible.

Ejercicio 4

Determina si los siguientes conjuntos de fórmulas son satisfacibles o insatisfacibles. Justifica tu respuesta.

1. $\Gamma = \{p \vee q, \neg p, \neg q\}$
2. $\Gamma = \{p \to q, q \to r, p, \neg r\}$
3. $\Gamma = \{p \wedge q, \neg(p \vee q)\}$
4. $\Gamma = \{\neg p \to q, \neg q \to r, \neg r \to p\}$
5. $\Gamma = \{p \vee q, q \to r, \neg r\}$

5. Propiedades Relevantes

Sea Γ un conjunto de fórmulas, donde $\varphi \in \Gamma$, τ representa una tautología y χ una contradicción. Existen una serie de propiedades relacionadas con la satisfacibilidad e insatisfacibilidad de Γ.

Si Γ es satisfacible

- ⋆ El conjunto $\Gamma \setminus \{\varphi\}$ también será satisfacible.
- ⋆ Si se añade una tautología, es decir, $\Gamma \cup \{\tau\}$, el conjunto resultante permanece satisfacible.
- ⋆ Si se agrega una contradicción, $\Gamma \cup \{\chi\}$ será necesariamente insatisfacible.

Ejemplo 13

- ⋆ $\Gamma = \{p \vee \neg p, p \to \neg p\}$ es satisfacible, entonces $\Gamma \setminus \{p \to \neg p\}$ es satisfacible.
- ⋆ $\Gamma = \{p \vee \neg p, p \to \neg p\}$ es satisfacible, entonces $\Gamma \cup \{p \leftrightarrow p\}$ es satisfacible.
- ⋆ $\Gamma = \{p \vee \neg p, p \to \neg p\}$ es satisfacible, entonces $\Gamma \cup \{p \wedge \neg p\}$ es insatisfacible.

Si Γ es insatisfacible

- ⋆ Cualquier extensión de Γ con una fórmula adicional, $\Gamma \cup \{\psi\}$, donde $\psi \in \mathsf{LProp}$, continuará siendo insatisfacible.
- ⋆ Al eliminar una tautología del conjunto, es decir, $\Gamma \setminus \{\tau\}$, el resultado seguirá siendo insatisfacible.

Ejemplo 14

- ⋆ $\Gamma = \{p \vee \neg p, p \wedge \neg p\}$ es insatisfacible, entonces $\Gamma \cup \{q\}$ es insatisfacible.
- ⋆ $\Gamma = \{p \wedge \neg p, p \leftrightarrow p\}$ es insatisfacible, entonces $\Gamma \setminus \{p \leftrightarrow p\}$ es insatisfacible.

Casos Indeterminados

- ⋆ Si tanto Γ como ψ son satisfacibles, no se puede concluir si $\Gamma \cup \{\psi\}$ será satisfacible o insatisfacible.
- ⋆ De manera similar, si Γ es insatisfacible y ψ no es una tautología, no es posible determina la satisfacibilidad de $\Gamma \setminus \{\psi\}$

Ejercicio 5

Demuestra todas las propiedades anteriores.

6. Automatización

Concretamente, en Haskell podemos representar el estado de las variables como una lista de tuplas, partiendo de esto podemos definir distintas funciones semánticas como sigue:

```
type Estado = [(String,Bool)]

interpreta :: LProp -> Estado -> Bool

tautologia :: LProp -> Bool

satisfacible :: LProp -> Bool

insatisfacible :: LProp -> Bool

contradiccion :: LPRop -> Bool

modelos :: LProp -> [Estado]

satisfacibleConj :: [LProp] -> Estado -> Bool

satConj :: [LProp] -> Bool

insatisfacibleConj :: [LProp] -> Estado -> Bool

insatConj :: [Prop] -> Prop -> Bool
```

7. Conclusión

Para concluir, hemos explorado en detalle la semántica de la lógica proposicional, comprendiendo la relación entre la sintaxis y el significado de las fórmulas en este sistema formal. A través de la definición rigurosa de interpretaciones, funciones de evaluación y tablas de verdad, hemos establecido un marco sólido para analizar el valor de verdad de las proposiciones bajo distintos estados. Además, hemos introducido conceptos fundamentales como la satisfacibilidad, las tautologías y las contradicciones, los cuales desempeñan un papel importante en la validación de argumentos lógicos y en aplicaciones como la verificación formal y la resolución de problemas mediante solucionadores SAT.

Este análisis semántico no solo nos proporciona herramientas teóricas, sino que también sienta las bases para su automatización en lenguajes de programación como Haskell. La implementación de evaluadores de fórmulas y verificadores de propiedades lógicas permite no solo comprender el comportamiento de los operadores lógicos, sino también aplicar estos conocimientos en el desarrollo de software y sistemas inteligentes. En futuras notas, exploraremos cómo estos principios pueden extenderse a sistemas más expresivos, como la lógica de primer orden, y cómo su estudio contribuye al desarrollo de herramientas de razonamiento automatizado.

Referencias

[1] Miranda Perea, F. E., Reyes, A. L., González, L. del C., & Linares, S. (2018). *Lógica Computacional: Notas de clase.* Facultad de Ciencias, Universidad Nacional Autónoma de México.

[2] Loyola Cruz, L. F. (2023). *Manual de Prácticas para la Asignatura de Lógica Computacional* (Proyecto de Apoyo a la Docencia). Universidad Nacional Autónoma de México.

[3] Enderton, H. B. (2001). *A Mathematical Introduction to Logic* (2nd ed.). Elsevier.

[4] Huth, M., & Ryan, M. (2004). *Logic in Computer Science: Modelling and Reasoning about Systems* (2nd ed.). Cambridge University Press.

[5] Mendelson, E. (2015). *Introduction to Mathematical Logic* (6th ed.). CRC Press.

[6] Harrison, J. (2009). *Handbook of Practical Logic and Automated Reasoning. Cambridge University Press.*

[7] Sipser, M. (2012). *Introduction to the Theory of Computation* (3rd ed.). Cengage Learning.