Lógica Computacional, 2025-2
Unidad 2: Lógica Proposicional

Equivalencia Lógica

Manuel Soto Romero   Luis Fernando Loyola Cruz

24 de febrero de 2025
Facultad de Ciencias UNAM

En esta nota exploraremos el concepto de equivalencia lógica en lógica proposicional y los métodos para demostrarla. Comenzaremos con una introducción al concepto de equivalencia lógica y su definición formal, resaltando su importancia en diversas aplicaciones de la computación. Luego, analizaremos dos enfoques principales para verificar la equivalencia entre fórmulas: la evaluación por interpretaciones y la manipulación algebraica. A través de ejemplos concretos, ilustraremos cómo utilizar estos métodos para demostrar equivalencias fundamentales, como la transformación de una implicación en una disyunción. También presentaremos una lista de equivalencias básicas que sirven como herramientas clave en las demostraciones algebraicas. Finalmente, abordaremos la automatización de la equivalencia lógica mediante su implementación en Haskell, destacando cómo las técnicas de verificación pueden integrarse en sistemas computacionales.

1. Introducción

En muchas ramas de la lógica, especialmente en sus aplicaciones a la computación, es fundamental determinar si dos fórmulas son equivalentes. Para demostrar la equivalencia entre dos fórmulas, se pueden emplear los siguientes métodos (dos semánticos y uno sintáctico):

- ⋆ Tablas de verdad Consiste en construir las tablas de verdad de ambas expresiones y compararlas.
- ⋆ Evaluación paso a paso Se analiza la evaluación de ambas fórmulas a partir de su interpretación.
- ⋆ Manipulación algebraica Se utiliza el álgebra de proposiciones para transformar una expresión en otra mediante manipulaciones algebraicas.

Omitiremos el método de la tabla de verdad por razones evidentes, además de que ya lo estudiaste en el curso de Estructuras Discretas. Por ello, nos enfocaremos en el segundo y tercer método. Antes de continuar, revisemos la definición de equivalencia de fórmulas.

Definición 1. (Equivalencia Lógica)

Dos fórmulas proposicionales φ y ψ son ***lógicamente equivalentes*** *si, para cualquier interpretación $\mathcal{I}$ ambas producen el mismo resultado, es decir, $\mathcal{I}(\varphi) = \mathcal{I}(\psi)$. Esto significa que, sin importar los valores que se asignen a sus variables, ambas expresiones siempre tendrán el mismo significado lógico. Para denotar esta equivalencia, escribimos:*

$$\varphi \equiv \psi$$

Piensa en dos formas distintas de expresar la misma idea, como decir *no es de día* y *es de noche*. Aunque las frases son diferentes, siempre describen la misma situación, sin importar el contexto. En lógica, ocurre algo similar: dos fórmulas son equivalentes cuando, sin importar cómo interpretemos sus componentes, siempre conducen al mismo resultado. Es como usar diferentes palabras para transmitir el mismo mensaje.

2. Demostración por Interpretaciones

Podemos utilizar interpretaciones para comprobar si dos fórmulas son equivalentes. Veamos un ejemplo.

Ejemplo 1

Demostraremos que $p \to q \equiv \neg p \vee q$

Tomemos una interpretación arbitraria $\mathcal{I}$. Dado que la expresión contiene un condicional, el caso más sencillo para analizar es cuando su valor es 0, y a partir de ahí verificaremos que la otra expresión también toma el mismo valor.

1. $\mathcal{I}(p \to q) = 0$
2. Esto sólo ocurre en un caso: Cuando $\mathcal{I} = 1$ y $\mathcal{I}(q) = 0$.
3. Al negar p, obtenemos $\mathcal{I}(\neg p) = 0$, y como $\mathcal{I}(q) = 0$.
4. Por la definición semántica de la disyunción, se concluye que $\mathcal{I}(\neg p \vee q) = 0$

Dado que este razonamiento es válido en ambos sentidos, se tiene que ambas expresiones son falsas en el mismo único caso y verdaderas en cualquier otro. Por lo tanto, su valor es el mismo en todas las interpretaciones.

Concluimos que $p \to q \equiv \neg p \vee q$. A esta equivalencia se le conoce como **definición de la implicación**, ya que muestra que la implicación puede interpretarse como una disyunción.

Este método resulta especialmente útil cuando el número de variables en las expresiones aumenta, ya que el tamaño de la tabla de verdad crece exponencialmente. Por ejemplo, una expresión con sólo cinco variables requeriría una tabla con 32 filas.

3. Equivalencias Básicas

A continuación, presentamos una lista de equivalencias básicas que utilizaremos en las demostraciones algebraicas. Su demostración mediante interpretaciones queda como ejercicio.

- ⋆ *Asociatividad*

$$\begin{aligned} (P \wedge Q) \wedge R &\equiv P \wedge (Q \wedge R) \\ (P \vee Q) \vee R &\equiv P \vee (Q \vee R) \end{aligned}$$

- ⋆ *Identidad*

$$\begin{aligned} P \vee \bot &\equiv P \\ P \wedge \top &\equiv P \end{aligned}$$

- ⋆ *Idempotencia*

$$\begin{aligned} P \vee P &\equiv P \\ P \wedge P &\equiv P \end{aligned}$$

- ⋆ *Dominación (elemento neutro)*

$$\begin{aligned} P \vee \top &\equiv \top \\ P \wedge \bot &\equiv \bot \end{aligned}$$

- ⋆ *Conmutatividad*

$$\begin{aligned} P \vee Q &\equiv Q \vee P \\ P \wedge Q &\equiv Q \wedge P \end{aligned}$$

- ⋆ *Tercero excluido*

$$P \vee \neg P \equiv \top$$

- ⋆ *Contradicción*

$$P \wedge \neg P \equiv \bot$$

- ⋆ *Doble negación*

$$\neg\neg P \equiv P$$

- ⋆ *Distribuitividad*

$$\begin{aligned} P \vee (Q \wedge R) &\equiv (P \vee Q) \wedge (P \vee R) \\ P \wedge (Q \vee R) &\equiv (P \wedge Q) \vee (P \wedge R) \end{aligned}$$

- ⋆ *De Morgan*

$$\begin{aligned} \neg (P \wedge Q) &\equiv \neg P \vee \neg Q \\ \neg (P \vee Q) &\equiv \neg P \wedge \neg Q \end{aligned}$$

⋆ *Eliminación de operadores*

$$
\begin{array}{rcl}
P \to Q & \equiv & \neg P \vee Q \\
P \leftrightarrow Q & \equiv & (\neg P \vee Q) \wedge (\neg Q \vee P) \\
P \leftrightarrow Q & \equiv & (P \wedge Q) \vee (\neg P \wedge \neg Q) \\
P \leftrightarrow Q & \equiv & (P \to Q) \wedge (Q \to P)
\end{array}
$$

4. Demostración por Manipulación Algebraica

La idea principal de este método es que, cuando dos expresiones son lógicamente equivalentes, una puede reemplazarse por la otra sin alterar el resultado. Veamos cómo se aplica este principio en la demostración algebraica. Para manipular las expresiones, utilizaremos las equivalencias presentadas en la sección anterior.

Ejemplo 2

Demostraremos que $r \equiv (p \to p) \to (\neg (s \to s) \vee r)$.

$$
\begin{array}{rcll}
r & \equiv & (p \to p) \to (\neg (s \to s) \vee r) & \\
 & \equiv & (\neg p \vee p) \to (\neg (s \to s) \vee r) & \text{Elim Op. } (\to) \\
 & \equiv & (p \vee \neg p) \to (\neg (s \to s) \vee r) & \text{Conmutatividad} \\
 & \equiv & \top \to (\neg (s \to s) \vee r) & \text{Tercero Excluido} \\
 & \equiv & \top \to (\neg (\neg s \vee s) \vee r) & \text{Elim Op. } (\to) \\
 & \equiv & \top \to (\neg (s \vee \neg s) \vee r) & \text{Conmutatividad} \\
 & \equiv & \top \to (\neg \top \vee r) & \text{Tercero Excluido} \\
 & \equiv & \top \to (\bot \vee r) & \text{Negación} \\
 & \equiv & \top \to (r \vee \bot) & \text{Conmutatividad} \\
 & \equiv & \top \to r & \text{Identidad} \\
 & \equiv & \neg \top \vee r & \text{Elim Op. } (\to) \\
 & \equiv & \bot \vee r & \text{Negación} \\
 & \equiv & r \vee \bot & \text{Conmutatividad} \\
 & \equiv & r & \text{Identidad}
\end{array}
$$

Esto puede hacerse en ambos sentidos, es decir, transformando la fórmula de la izquierda para que coincida con la de la derecha o viceversa. El objetivo es obtener la misma expresión en ambos lados.

5. Automatización

Se puede definir la equivalencia lógica en Haskell, a través de la definición que usa el concepto de interpretación, mediante la función:

```
equiv :: LProp -> LProp -> Bool
```

6. Conclusión

En esta nota hemos explorado la equivalencia lógica en lógica proposicional y los métodos para demostrarla. Analizamos cómo las interpretaciones permiten verificar si dos expresiones siempre producen el mismo resultado y cómo la manipulación algebraica facilita la transformación de una fórmula en otra utilizando reglas predefinidas. Estos métodos son fundamentales para la simplificación de expresiones lógicas y la verificación formal en diversos ámbitos de la computación.

Además, mostramos una lista de equivalencias básicas que pueden emplearse en demostraciones algebraicas y discutimos la posibilidad de automatizar la verificación de equivalencias mediante programación en Haskell. Comprender estos conceptos no solo es útil en la teoría, sino que también tiene aplicaciones prácticas en la optimización de código, el diseño de circuitos digitales y la validación de sistemas lógicos en inteligencia artificial y verificación formal.

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

[4] Ben-Ari, M. (2012). *Mathematical Logic for Computer Science* (3rd ed.). Springer.

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

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

[7] Thompson, S. (2011). *Haskell: The Craft of Functional Programming* (3rd ed.). Addison-Wesley.