Lógica Computacional, 2025-2
Unidad 3: Lógica de Predicados

Equivalencia Lógica

Manuel Soto Romero Luis Fernando Loyola Cruz

10 de mayo de 2025
Facultad de Ciencias UNAM

En esta nota abordaremos las principales equivalencias lógicas en el contexto de la lógica de predicados, extendiendo las nociones vistas previamente en lógica proposicional. Nuestro objetivo es comprender cómo se comportan los cuantificadores frente a operadores lógicos como la negación, la conjunción y la disyunción, y cómo estas propiedades nos permiten transformar fórmulas de manera estructurada. Además, mostraremos cómo estas transformaciones pueden formalizarse y automatizarse en un lenguaje funcional como Haskell, lo cual sienta las bases para construir herramientas de verificación lógica y sistemas inteligentes. Esta nota combina teoría, ejemplos trabajados paso a paso y una aproximación computacional para fortalecer la intuición y la precisión formal.

1. Equivalencias con Cuantificadores

Negación de cuantificadores

Estas equivalencias muestran cómo se comporta la negación frente a los cuantificadores universales y existenciales:

$$\begin{aligned} \neg\forall x\, \varphi &\equiv \exists x\, \neg\varphi \\ \neg\exists x\, \varphi &\equiv \forall x\, \neg\varphi \end{aligned}$$

Es decir, negar que una propiedad se cumpla para todos equivale a afirmar que hay al menos un caso donde no se cumple, y viceversa.

Ejemplo 1

No es cierto que todas las personas aprobaron

$$\neg\forall x Aprobó(x) \equiv \exists x \neg Aprobó(x)$$

Es decir: *Existe al menos un estudiante que no aprobó.*

Distribuitividad

En algunos casos, podemos distribuir el cuantificador sobre conectivos lógicos como el $\wedge$ (conjunción) y $\vee$ (disyunción):

$$\begin{aligned} \forall x\,(\varphi \wedge \psi) &\equiv \forall x\,\varphi \wedge \forall x\,\psi \\ \exists x\,(\varphi \vee \psi) &\equiv \exists x\,\varphi \vee \exists x\,\psi \end{aligned}$$

Estas equivalencias se cumplen siempre y cuando el cuantificador afecte por igual a todas las partes de la fórmula.

Ejemplo 2

Todo número natural es positivo y entero.

$$\forall x(Positivo(x) \wedge Entero(x)) \equiv \forall x Positivo(x) \wedge \forall x Entero(x)$$

Cuantificación vacua

Cuando la variable x no aparece libre en la fórmula φ, es decir, no tiene efecto en ella, la cuantificación es irrelevante:

$$\begin{aligned} \forall x\,\varphi &\equiv \varphi \quad (\text{si } x \text{ no es libre en } \varphi) \\ \exists x\,\varphi &\equiv \varphi \quad (\text{si } x \text{ no es libre en } \varphi) \end{aligned}$$

Esto se conoce como cuantificación vacua.

Ejemplo 3

$$\forall x(2+2=4) \equiv 2+2=4$$

Como la expresión no depende de x, cuantificar sobre x no cambia su valor lógico. Lo mismo pasa en:

$$\forall x(\exists x P(x)) \equiv \exists x P(x)$$

Como la variable x está ligada al cuantificador existencial, añadir el cuantificador universal no tiene ningún efeto sobre la fórmula.

Prenexación (reordenamiento)

Las siguientes equivalencias son útiles para llevar cuantificadores al principio de una fórmula (forma prenexa), siempre que x no sea libre en φ:

$$\begin{aligned} \varphi \wedge \forall x\,\psi &\equiv \forall x\,(\varphi \wedge \psi) \quad (\text{si } x \text{ no es libre en } \varphi) \\ \varphi \vee \forall x\,\psi &\equiv \forall x\,(\varphi \vee \psi) \quad (\text{si } x \text{ no es libre en } \varphi) \\ \varphi \wedge \exists x\,\psi &\equiv \exists x\,(\varphi \wedge \psi) \quad (\text{si } x \text{ no es libre en } \varphi) \\ \varphi \vee \exists x\,\psi &\equiv \exists x\,(\varphi \vee \psi) \quad (\text{si } x \text{ no es libre en } \varphi) \\ \varphi \to \forall x\,\psi &\equiv \forall x\,(\varphi \to \psi) \quad (\text{si } x \text{ no es libre en } \varphi) \\ \varphi \to \exists x\,\psi &\equiv \exists x\,(\varphi \to \psi) \quad (\text{si } x \text{ no es libre en } \varphi) \\ \forall x\,\varphi \to \psi &\equiv \exists x\,(\varphi \to \psi) \quad (\text{si } x \text{ no es libre en } \varphi) \\ \exists x\,\varphi \to \psi &\equiv \forall x\,(\varphi \to \psi) \quad (\text{si } x \text{ no es libre en } \varphi) \end{aligned}$$

Estas transformaciones son útiles para la normalización de fórmulas que veremos más adelante y para facilitar la evaluación semántica o razonamiento automatizado.

Ejemplo 4

Hoy es lunes y existe una persona que está feliz.

$$Lunes(y) \land \exists x Feliz(x) \equiv \exists x(Lunes(y) \land Feliz(x))$$

Aquí, el predicado *Lunes* no se aplica a x, así que el cuantificador puede moverse.

Ejemplo 5

En esta sección, aplicaremos una serie de equivalencias lógicas para transformar fórmulas del lenguaje de primer orden. Trabajaremos sobre un dominio que incluye personas habitantes de la Ciudad de México, lapsos de tiempo y exámenes académicos.

Utilizaremos los siguientes predicados:

- $S(x)$: x es estudiante de la Facultad de Ciencias
- $A(x)$: x es alumno
- $E(x, y)$: x estudia en el tiempo y
- $C(x)$: el examen x fue calificado
- $I(x)$: x es inteligente
- $T(x)$: x representa un tiempo
- $R(x)$: x reprueba
- $P(x)$: x es un examen

A continuación, presentamos dos ejemplos representativos:

Enunciado: No todos los estudiantes de la Facultad de Ciencias son inteligentes.

$$\begin{aligned} \neg\forall x\,(S(x) \to I(x)) &\equiv \exists x\,\neg(S(x) \to I(x)) && \text{(Negación de cuantificador)} \\ &\equiv \exists x\,\neg(\neg S(x) \lor I(x)) && \text{(Eliminación del } \to\text{)} \\ &\equiv \exists x\,(S(x) \land \neg I(x)) && \text{(Ley de De Morgan)} \end{aligned}$$

Interpretación final: Existe al menos un estudiante de la Facultad de Ciencias que no es inteligente.

Enunciado: No existe ningún alumno que estudie en todo momento del tiempo.

$$\neg\exists x\,(A(x) \land \forall y\,(T(y) \to E(x, y))) \equiv$$

$$\begin{aligned} &\equiv \forall x\,\neg\,(A(x) \land \forall y\,(T(y) \to E(x, y))) && \text{(Negación de cuantificador)} \\ &\equiv \forall x\,(\neg A(x) \lor \neg\forall y\,(T(y) \to E(x, y))) && \text{(Ley de De Morgan)} \\ &\equiv \forall x\,(\neg A(x) \lor \exists y\,\neg(T(y) \to E(x, y))) && \text{(Negación de cuantificador)} \\ &\equiv \forall x\,(\neg A(x) \lor \exists y\,(T(y) \land \neg E(x, y))) && \text{(Ley de De Morgan)} \\ &\equiv \forall x\,(A(x) \to \exists y\,(T(y) \land \neg E(x, y))) && \text{(Eliminación del } \to\text{)} \end{aligned}$$

Interpretación final: Todo alumno tiene al menos un momento de tiempo en el que no estudia.

2. Automatización

Las equivalencias lógicas de la lógica de predicados pueden representarse de forma estructurada y ejecutable en un lenguaje funcional como Haskell. Esto permite automatizar transformaciones como la negación de cuantificadores, la distribución de conectivos, la eliminación de cuantificadores vacuos o la prenexación de fórmulas.

A continuación se describen las firmas de funciones que implementan dichas equivalencias:

```
negacionCuantificador :: Formula -> Formula

distribuyeCuantificadores :: Formula -> Formula

cuantificacionVacua :: Formula -> Formula

prenex :: Formula -> Formula
```

Estas funciones pueden componerse para construir herramientas de transformación automática de fórmulas lógicas. Este enfoque permite no solo verificar propiedades como equivalencia o validez, sino también preparar fórmulas para procesos de resolución, skolemización o prueba automática.

3. Conclusión

A lo largo de esta nota, exploramos las principales equivalencias lógicas que enriquecen el razonamiento en lógica de predicados. Al extender las reglas conocidas de la lógica proposicional, aprendimos a transformar fórmulas que involucran cuantificadores de forma estructurada, precisa y justificada. Estas transformaciones no solo permiten simplificar o normalizar expresiones, sino también prepararlas para procesos de demostración automatizada y verificación formal.

Asimismo, vimos cómo estas ideas pueden implementarse en un lenguaje funcional como Haskell, lo que permite construir herramientas que operen sobre fórmulas como objetos computables. Esta perspectiva computacional complementa la teoría lógica con aplicaciones prácticas, y abre la puerta al diseño de sistemas capaces de razonar de forma automática. Comprender estas equivalencias es un paso fundamental para seguir avanzando en el estudio de métodos formales, resolución y lógica computacional.

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] Reyes, A. L., & Reyes, R. (2021). *Matemáticas Discretas: Notas de clase.* Colegio de Ciencia y Tecnología, UACM San Lorenzo.

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

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

[**6**] Gries, D., & Schneider, F. B. (1993). *A Logical Approach to Discrete Math*. Springer.

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

[**8**] Nipkow, T., Paulson, L. C., & Wenzel, M. (2002). *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer.