Notas de Lógica

Luis Eduardo Gamboa Guzmán

Universidad Michoacana de San Nicolás de Hidalgo
Facultad de Ingeniería Eléctrica

17 de junio de 2020

Índice general

Sobre este documento

Este documento es una recopilación de conceptos y ejercicios obtenidos mayormente de [Alagar 1989], [Doerr 1985], [Enderton 2000], [Hopcroft 1979], [Knuth 1989], [Rosen 1999] y [Tourlakis 1984]. Debido a la cantidad de traducciones de estas fuentes, las referencias a cada concepto en específico han sido removidas. Los cambios realizados incluyen traducción, notación, reordenamiento, expansión y corrección de errores (aunque pocos) del material.

Muchos ejercicios han sido desarrollados enteramente por el autor de esta recopilación. Otros tantos han sido modificados de los expuestos en la bibliografía para hacerlos más claros y utilizando los conceptos que abarca el curso.

Este documento ha sido desarrollado utilizando LaTeX.

Capítulo 1

Lógica proposicional

La resolución de problemas, diseño de algoritmos y programación requieren un razonamiento lógico completo. La *lógica* trata los métodos y el arte del razonamiento sistemático.

Una proposición es una sentencia declarativa que es verdadera o falsa pero no ambas. Por ejemplo, "la mañana es fría".

1.1. Proposiciones lógicas

Una proposición que es indivisible se conoce como proposición primitiva. Las sentencias derivadas de las primitivas y de varios conectores lógicos como *no*, *y*, *o*, *si...entonces* y *sí y sólo sí* se conocen como proposiciones compuestas.

Ejemplo

- Un girasol es amarillo.
- El Sahara es un desierto.
- 17 es un número primo y 25 no es un cuadrado perfecto.
- Existe una infinidad de números perfectos.
- ¿Estás durmiendo?

Debemos ser muy cuidadosos con sentencias como la última. A pesar de que preguntas como *¿Estás durmiendo?, ¿el cielo es azul?* tienen como respuestas "sí" o "no", no se trata de proposiciones. Una proposición debe declarativa, y en el caso de las preguntas no se está declarando algo.

1.2. Tablas de Verdad de Operadores Lógicos

Las tablas de verdad son una forma conveniente de mostrar los valores de una proposición compuesta. En su construcción, usamos *1* para verdadero y *0* para falso, aunque también es común utilizar T y F.

NO

Una sentencia que es modificada con el conectivo *no* es llamada la negación de la sentencia original. Simbólicamente, sí P es una proposición entonces $\neg P$ (no P), denota la negación de P. En el cuadro 1.1 se muestra la tabla de verdad de NO.

P	$\neg P$
1	0
0	1

Cuadro 1.1: Tabla de verdad de NO

Y

La conjunción de P,Q es denotada por $P \wedge Q$. La conjunción es verdadera sólo si P y Q son verdaderos. En el cuadro 1.2 se muestra la tabla de verdad de Y.

P	Q	$P \wedge Q$
0	0	0
0	1	0
1	0	0
1	1	1

Cuadro 1.2: Tabla de verdad de Y

O

La disyunción de P,Q es denotada por $P \vee Q$. La disyunción es verdadera si al menos uno de sus elementos es verdad P, Q es verdadero. En el cuadro 1.3 se muestra la tabla de verdad de O.

P	Q	$P \vee Q$
0	0	0
0	1	1
1	0	1
1	1	1

Cuadro 1.3: Tabla de verdad de O

O EXCLUSIVO

El símbolo $\oplus$ representa el O EXCLUSIVO (XOR), que es incluido en muchos lenguajes de programación. Una proposición $P \oplus Q$ se lee como "P o Q pero no ambos". En el cuadro 1.4 se muestra la tabla de verdad de XOR.

P	Q	$P \oplus Q$
0	0	0
0	1	1
1	0	1
1	1	0

Cuadro 1.4: Tabla de verdad de XOR

IMPLICACION

Para dos declaraciones P,Q, decimos "P implica Q" y se escribe $P \to Q$ para denotar la implicación de Q por P. La proposición P es llamada la hipótesis o antecedente de la implicación; Q es llamada la conclusión o consecuente de la implicación. En el cuadro 1.5 se muestra la tabla de verdad de la IMPLICACION.

P	Q	$P \to Q$
0	0	1
0	1	1
1	0	0
1	1	1

Cuadro 1.5: Tabla de verdad de IMPLICACION

Como ejemplo, consideremos que el profesor dice a sus alumnos: "si obtienes 9 o más en el examen, aprobaras el curso". Entonces:

- P: Obtienes 9 o más en el examen.
- Q: Apruebas el curso.

Una vez que se termina el curso, existen 4 posibles situaciones:

1. La calificación del examen ha sido menor que 9 y no se aprobó el curso. La promesa no ha sido rota, pues no se cumplió con P.
2. La calificación del examen ha sido menor que 9 y se aprobó el curso. La promesa no ha sido rota, es posible que por otras razones se haya aprobado.
3. La calificación del examen ha sido mayor o igual que 9 y no se aprobó el curso. La promesa ha sido rota, pues se ha cumplido con P y no se ha aprobado el curso.
4. La calificación del examen ha sido mayor o igual que 9 y se aprobó el curso. La promesa ha sido cumplida.

SI Y SOLO SI

Otra declaración común en matemáticas es "P si y sólo si Q", o simbólicamente $P \leftrightarrow Q$. Esto es llamado la equivalencia de dos proposiciones, P, Q. Formulaciones alternativas son:

- si P entonces Q, y si Q entonces P
- Q es una condición necesaria y suficiente para P

La tabla de verdad de SII se muestra en el cuadro 1.6.

P	Q	$P \leftrightarrow Q$
0	0	1
0	1	0
1	0	0
1	1	1

Cuadro 1.6: Tabla de verdad de SII

1.2.1. Precedencia de Operador, Tautologías y Contradicciones

Una fórmula o forma lógica $f(x, y, z, \ldots)$ es una expresión lógica en la que $x, y, z, \ldots$ son proposiciones o variables lógicas. Por ejemplo $(x \to y) \to z$ y $(x \wedge \neg y) \vee z$ son fórmulas.

Por convención los conectores en una fórmula sin paréntesis son aplicados en el siguiente orden de precedencia:

- $\neg$ (precedencia más alta)
- $\wedge$
- $\vee$
- $\oplus$
- $\to$
- $\leftrightarrow$ (precedencia más baja)

los paréntesis refuerzan la prioridad para subexpresiones encerradas. Los conectores con la misma precedencia son aplicados de izquierda a derecha. Entonces la fórmula $(x \wedge \neg y) \vee z$ puede ser escrita sin paréntesis como $x \wedge \neg y \vee z$; las fórmulas $(x \to \neg y) \to z$, $x \to \neg(y \to z)$ y $x \to (\neg y \to z)$ son diferentes.

Tautología

Una fórmula que siempre es verdad se conoce como tautología. Entonces $x \vee \neg x$ es una tautología. Si dos fórmulas f y g tienen valores idénticos en sus tablas de verdad, entonces $f \leftrightarrow g$ es una tautología. Esto es, si dos fórmulas f y g son lógicamente equivalentes, denotado por $f \equiv g$, si y sólo si $f \leftrightarrow g$ es una tautología.

Por ejemplo, se quiere comprobar si $P \rightarrow Q \equiv \neg P \vee Q$ son lógicamente equivalentes, la tabla de verdad se muestra en el cuadro 1.7.

P	Q	$P \rightarrow Q$	$\neg P \vee Q$	$(P \rightarrow Q) \leftrightarrow (\neg P \vee Q)$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	1
1	1	1	1	1

Cuadro 1.7: Tautología para demostrar $P \rightarrow Q \equiv \neg P \vee Q$

Contradicción

Se dice que una fórmula es una contradicción si siempre es falsa. $x \wedge \neg x$ es una contradicción.

1.3. Tablas de Verdad de Proposiciones Compuestas

Para realizar la tabla de verdad de una Proposición compuesta, es necesario primero identificar los elementos sobre los cuales se aplicarán las operaciones en estricto apego a la precedencia de operador.

Para ilustrar la importancia que tiene la precedencia de operador, consideremos la expresión $x \rightarrow \neg y \rightarrow z$ y agrupemos las expresiones de distintas maneras usando paréntesis:

1. $(x \rightarrow \neg y) \rightarrow z$
2. $x \rightarrow \neg(y \rightarrow z)$
3. $x \rightarrow (\neg y \rightarrow z)$

En este caso, sólo la expresión #1 corresponde al resultado correcto. A continuación se presentan las tablas de verdad:

x	y	z	$\neg y$	$x \to \neg y$	$x \to \neg y \to z$
0	0	0	1	1	0
0	0	1	1	1	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	0	0	1
1	1	1	0	0	1

Cuadro 1.8: Tabla de verdad para $x \to \neg y \to z$, es equivalente a la expresión $(x \to \neg y) \to z$

x	y	z	$y \to z$	$\neg(y \to z)$	$x \to \neg(y \to z)$
0	0	0	1	0	1
0	0	1	1	0	1
0	1	0	0	1	1
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	0	0
1	1	0	0	1	1
1	1	1	1	0	0

Cuadro 1.9: Tabla de verdad para $x \to \neg(y \to z)$

De aquí podemos ver que los valores de verdad para las expresiones $x \to \neg y \to z$ y $x \to \neg(y \to z)$ son distintos (comparando la última columna de las respectivas tablas de verdad).

EJERCICIOS

Realiza las siguientes tablas de verdad:

1. $x \to (\neg y \to z)$, posteriormente compara con las tablas de verdad de $x \to \neg y \to z$ y $x \to \neg(y \to z)$ para reforzar la importancia de respetar la precedencia de operador.
2. $P \to R \to Q \to P \wedge R$
3. $P \to Q \wedge R \vee \neg P \wedge R$

Determina por medio de tabla de verdad, cuales de las siguientes expresiones son tautologías o contradicciones:

4. $P \to (Q \to P)$
5. $\neg Q \to \neg P \to (P \to Q)$

6. $P \vee Q \rightarrow P \wedge \neg R$

7. $P \wedge \neg(Q \rightarrow P)$

1.4. Construcción de equivalencias lógicas

Supongamos una función $f(P, Q)$ cuya tabla de verdad es conocida y se desea encontrar la expresión equivalente. Para encontrar f se pueden utilizar dos técnicas:

1. Forme expresiones lógicas utilizando el operador conjunción para generar un valor *verdadero* en los casos en los que la función regresa un valor verdadero. Finalmente, forme una disyunción con las expresiones encontradas.

2. Forme expresiones lógicas utilizando el operador disyunción para generar un valor *falso* en los casos en los que la función regresa un valor falso. Finalmente, forme una conjunción con las expresiones encontradas.

Ejemplo: Encuentre una función f que genere los valores de la tabla de verdad en el cuadro 1.10.

P	Q	R	$f(P, Q, R)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Cuadro 1.10: Tabla de verdad para $f(P, Q, R)$

A continuación se ilustran las expresiones obtenidas para valores verdaderos y para valores falsos.

P	Q	R	$f(P, Q, R)$	Técnica 1	Técnica 2
0	0	0	0		$P \vee Q \vee R$
0	0	1	0		$P \vee Q \vee \neg R$
0	1	0	0		$P \vee \neg Q \vee R$
0	1	1	1	$\neg P \wedge Q \wedge R$	
1	0	0	0		$\neg P \vee Q \vee R$
1	0	1	0		$\neg P \vee Q \vee \neg R$
1	1	0	1	$P \wedge Q \wedge \neg R$	
1	1	1	1	$P \wedge Q \wedge R$	

Dadas las expresiones que generan valores verdaderos, podemos deducir que:

$$f(P,Q,R) \equiv (\neg P \wedge Q \wedge R) \vee (P \wedge Q \wedge \neg R) \vee (P \wedge Q \wedge R)$$

y dadas las expresiones que generan valores falsos podemos deducir:

$$f(P,Q,R) \equiv (P \vee Q \vee R) \wedge (P \vee Q \vee \neg R) \wedge (P \vee \neg Q \vee R) \wedge (\neg P \vee Q \vee R) \wedge (\neg P \vee Q \vee \neg R)$$

EJERCICIOS

Obtenga, utilizando ambas técnicas, las expresiones que generan las tablas de verdad de f, g, h, j:

P	Q	R	$f(P,Q,R)$	$g(P,Q,R)$	$h(P,Q,R)$	$j(P,Q,R)$
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	1	0	1	1	0	1
0	1	1	1	1	0	1
1	0	0	0	0	0	1
1	0	1	1	0	0	1
1	1	0	1	1	0	1
1	1	1	0	1	1	1

Cuadro 1.11: Tabla de verdad de las expresiones f, g, h, j

1.5. Conjunto de equivalencias lógicas

Leyes de idempotencia

$$P \equiv P \vee P$$
$$P \equiv P \wedge P$$

Leyes conmutativas

$$P \vee Q \equiv Q \vee P$$
$$P \wedge Q \equiv Q \wedge P$$

Leyes asociativas

$$(P \vee Q) \vee R \equiv P \vee (Q \vee R)$$
$$(P \wedge Q) \wedge R \equiv P \wedge (Q \wedge R)$$

Leyes distributivas

$$P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$$
$$P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$$

Leyes de absorción

$$P \vee 0 \equiv P$$
$$P \vee 1 \equiv 1$$
$$P \wedge 0 \equiv 0$$
$$P \wedge 1 \equiv P$$
$$P \wedge (P \vee Q) \equiv P$$
$$P \vee (P \wedge Q) \equiv P$$

Leyes de De Morgan

$$\neg(P \vee Q) \equiv \neg P \wedge \neg Q$$
$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$$

Leyes de complemento

$$\neg 1 \equiv 0$$
$$\neg 0 \equiv 1$$
$$P \vee \neg P \equiv 1$$
$$P \wedge \neg P \equiv 0$$
$$\neg(\neg P) \equiv P$$

Ley de implicación

$$P \to Q \equiv \neg P \vee Q$$
$$P \to Q \equiv \neg Q \to \neg P$$

Ley de doble implicación

$$P \leftrightarrow Q \equiv (P \to Q) \wedge (Q \to P)$$

Ley de o-exclusivo

$$P \oplus Q \equiv P \wedge \neg Q \ \vee \ \neg P \wedge Q$$

1.5.1. Uso de Leyes de De Morgan

Considere la siguiente situación mientras se está programando algún sistema. Se sabe que cuando NO SUCEDE alguna condición es necesario efectuar alguna acción, el código más o menos sería:

```
if(!condicion)
{
    /* accion a ejecutar */
}
```

Ahora consideremos que está condición requiere que una variable tenga uno de dos valores distintos, es decir, "`a==b || a==c`". Sin embargo, se pretende que la acción se ejecute cuando esa condición no es satisfecha. El código resultante, frecuentemente es:

```
/* CODIGO INCORRECTO */
if(a!=b || a!=c)
{
    /* accion a ejectuar */

}
```

sin embargo, es INCORRECTO. Este es un error muy común en programación al momento de negar condiciones. El error consiste en negar cada una de las subcondiciones, ignorando el rol que tiene el operador lógico en la expresión.

El código correcto, en donde se niega la condición haciendo uso de paréntesis es:

```
/* CODIGO CORRECTO (usa parentesis para negar la expresion) */
if(!(a==b || a==c))
{
    /* accion a ejectuar */
}
```

y si queremos quitar el paréntesis que niega la expresión y en su lugar utilizar el operador `!=`, lo correcto es utilizar las Leyes de De Morgan. Estas leyes nos indican que la negación de una disyunción es la conjunción de las negaciones de cada subexpresión. Análogamente, existe una ley para la negación de conjunción. De vuelta a nuestro problema, tenemos que el código correcto utilizando las leyes de De Morgan es:

```
/* CODIGO CORRECTO (usando Leyes de De Morgan) */
if(a!=b && a!=c)
{
    /* accion a ejectuar */
}
```

EJERCICIOS

Aplique las leyes de De Morgan para aplicar la negación a una expresión agrupada en paréntesis:

1. $\neg(P \vee Q \vee R)$
2. $\neg(P \vee Q \wedge R)$
3. $\neg(P \wedge \neg Q \vee \neg P \wedge \neg Q)$
4. $\neg((P \vee Q) \wedge (Q \vee \neg R) \wedge \neg(P \vee \neg Q))$

Aplique las leyes de De Morgan para obtener una expresión equivalente que sea la negación de una proposición compuesta agrupada en paréntesis:

5. $\neg P \vee Q$
6. $P \wedge Q \vee \neg P$

Aplique las equivalencias lógicas y posteriormente las Leyes de De Morgan según sea necesario para obtener una proposición que NO CONTENGA negaciones de expresiones agrupadas y que sólo utilice conjunción, disyunción y negación como operadores lógicos:

7. $\neg(P \to Q)$
8. $\neg(P \leftrightarrow Q \wedge R)$
9. $(\neg P \vee \neg Q) \wedge (R \to Q) \wedge (S \to P) \to \neg(R \wedge S)$

1.6. Lógica Proposicional en Programación

La lógica proposicional es fundamental para la programación. Hasta ahora hemos visto un ejemplo de errores comunes en el uso de las Leyes de De Morgan en el apartado 1.5.1. En esta sección además explicaremos como se relacionan los operadores lógicos con la programación utilizando como ejemplo el Lenguaje C.

1.6.1. Operadores lógicos

Las operaciones lógicas típicas están disponibles en el Lenguaje C, a continuación mostramos el operador asociado en el lenguaje.

Operación Lógica	Operador Lógico	Operador en C	Operación en C
Negación	$\neg$	`!`	not
Conjunción	$\wedge$	`&&`	and
Disyunción	$\vee$	`||`	or

Cuadro 1.12: Operaciones, operadores lógicos y sus equivalentes en C

Operaciones más complejas como la implicación $\rightarrow$, doble implicación $\leftrightarrow$ o el o-exclusivo $\oplus$, deben ser implementadas utilizando únicamente los operadores disponibles. Recuerde que es posible encontrar una expresión lógica equivalente a cualquier proposición utilizando únicamente negación, conjunción y disyunción, vea las secciones 1.4 y 1.5.

Valores de verdad en C

En el Lenguaje C, las expresiones lógicas toman valores numéricos y éstos son interpretados para determinar si una expresión es falsa o verdadera. Un valor numérico de 0 indica que la expresión es falsa, mientras que *cualquier* valor distinto indicará que la expresión es verdadera.

Esta es una consideración importante que debe tomarse, pues no necesariamente las expresiones verdaderas tendrán un valor numérico de 1. De esta forma, la condición en "`if(5)`" evaluará a verdadero y se ejecutará el bloque correspondiente.

1.6.2. Evaluación corto circuito

La evaluación corto circuito es utilizada automáticamente por el Lenguaje C para optimizar el rendimiento del programa. Este tipo de evaluaciones aprovecha las características de la conjunción y la disyunción al momento de definir el valor de verdad de una expresión. Así, para una conjunción, la evaluación corto circuito consiste en no evaluar el operando de la derecha cuando el operando de la izquierda es falso. De manera análoga, en la disyunción, la optimización consiste en no evaluar el segundo operando cuando el primero es verdadero.

En caso que no se cumpla la condición necesaria en el primer operando de acuerdo a la optimización, se procede a evaluar el segundo operando y será éste el que determine el valor de verdad de la expresión:

Operación	Valor de verdad	Nota
$0 \wedge B$	0	B no es evaluado
$1 \wedge B$	B	A y B son evaluados
$0 \vee B$	B	A y B son evaluados
$1 \vee B$	1	B no es evaluado

Cuadro 1.13: Considere una operación sobre proposiciones A y B, en algunos casos, es posible conocer el valor de verdad de la expresión utilizando únicamente el valor de A.

Ejemplo en C

```c
int A()
{
    printf("Funcion A ejecutada\t");
    return 1;
}

int B()
{
    printf("Funcion B ejecutada\t");
    return 0;
}

int main()
{
    /* Conjuncion */
    printf("**CONJUNCION -- Evaluacion corto circuito**\n");
    printf("Probando A && B, es decir, (1 && 0)\n");
    if( A() && B() )
        printf("\nA && B es verdad\n");
    else
        printf("\nA && B es falso\n");

    printf("\nProbando B && A, es decir, (0 && 1)\n");
    if( B() && A() )
        printf("\nB && A es verdad\n");
    else
        printf("\nB && A es falso\n");
```

```
    /* Disyuncion */
    printf("\n\n**DISYUNCION -- Evaluacion corto circuito**\n");
    printf("Probando A || B, es decir, (1 || 0)\n");
    if( A() || B() )
        printf("\nA || B es verdad\n");
    else
        printf("\nA || B es falso\n");

    printf("\nProbando B || A, es decir, (0 || 1)\n");
    if( B() || A() )
        printf("\nB || A es verdad\n");
    else
        printf("\nB || A es falso\n");

    return 0;
}
```

1.6.3. Operaciones con bits

Los operadores con bits son distintos a los operadores lógicos, pues mientras los operadores lógicos trabajan con el valor de verdad de un entero, las operaciones con bits son aplicadas a cada bit que forma un entero. Visto de otra forma, se considera cada bit de un entero como un valor de verdad y el operador con bits aplica la misma operación bit a bit, abarcando la totalidad del entero. Por ejemplo, una operación en bits *and* sobre 0011 y 1010 daría como resultado 0010, de igual manera una operación *or* sobre los mismos valores tendría como resultado 1011.

Operación con bits en C	Operador en C
bitwise not	~
bitwise and	&
bitwise or	|
bitwise xor	^

Cuadro 1.14: Operaciones con bits en C

EJERCICIO

Realizar un programa en C que, dados dos enteros de 32-bits:

- Imprima cada entero en binario
- Imprima en binario el resultado del bitwise not
- Imprima en binario el resultado después de aplicar un bitwise and, or y xor

Capítulo 2

Lógica de predicados

Frecuentemente nos encontramos con proposiciones que representan hechos sobre una colección de objetos. Por ejemplo:

- Algunos programadores son inteligentes.
- Todos los municipios tienen escuelas públicas.
- Todos los matemáticos son tenaces.
- Existe un número impar que no es primo.

Cada declaración conlleva una aserción común a algunos objetos que pertenecen a un universo. Puesto que las declaraciones *para todos* y *existe* (o *para algún*) no están disponibles en lógica proposicional, ninguna de estas declaraciones puede ser escrita en forma lógica. Cuando agregamos símbolos para estas declaraciones junto con las reglas de uso en lógica proposicional, obtenemos lógica de predicados. El lenguaje de lógica de primer orden es obtenido cuando símbolos de función son agregados a lógica de predicados.

Contrario a las constantes, las variables no tienen un significado por sí mismas. Una sentencia como $2 + 3 = 5$ es una aserción cuyo valor de verdad es conocido. Sin embargo sentencias como $x > 6$, "él es abogado" y "y es un entero" contienen variables: x, "él", y. Dichas sentencias no pueden ser comprobadas ni refutadas. Sin embargo, cuando asignamos valores a estas variables el valor de verdad puede ser conocido. Por lo tanto cuando $x = 3$, "$x > 6$" es falso; cuando "él" es reemplazado por Juan, la declaración "Juan es abogado" tendrá un valor de verdad; cuando $y = 2$ la declaración "y es un entero" es verdadera. Tales declaraciones cuyos valores de verdad dependen de los valores que tengan las variables se conocen como *predicados*.

Es importante observar que en lógica proposicional una variable toma valores (falso, verdadero) y en lógica de predicados una variable toma valores de un universo de discurso U. Las variables $x_1, x_2, \ldots, x_n$ en $P(x_1, x_2, \ldots, x_n)$ son llamadas variables libres del predicado. Por consecuencia, el valor de verdad de $P(x_1, x_2, \ldots, x_n)$ varia conforme $x_1, x_2, \ldots, x_n$ asumen diferentes valores en U.

Por lo que los predicados en lógica de predicados son las variables de lógica proposicional, y la *atadura* transforma un predicado en una proposición.

Por ejemplo, en $P(x) =$ "x es abogado" y $Q(y) =$ "y es hombre", podríamos formar un predicado $P(x) \wedge Q(y)$ (donde $P(x)$ y $Q(y)$ son variables de lógica proposicional) y por medio de atadura podríamos convertirlo a una proposición del tipo $P(Juan) \wedge Q(Pedro)$.

2.1. Cuantificador universal

El cuantificador universal $\forall$ es utilizado para crear una proposición $\forall x P(x)$, leída como "para todo x, $P(x)$ es verdadero". Esta proposición es verdad si y sólo si $P(a)$ es verdad para cada a en un universo U. Esto es:

$$\begin{aligned} \forall x P(x) &= P(x_1) \wedge P(x_2) \wedge P(x_3) \wedge \cdots \\ &= \bigwedge_{x_i \in U} P(x_i) \end{aligned}$$

2.2. Cuantificador existencial

El cuantificador existencial $\exists$ es usado para formar una proposición $\exists x P(x)$, leída como "existe un x tal que $P(x)$ es verdadero" o "para algún x, $P(x)$ es verdadero". Esta proposición es verdad si y sólo si $P(a)$ es verdad para al menos un a en U. Esto es:

$$\begin{aligned} \exists x P(x) &= P(x_1) \vee P(x_2) \vee P(x_3) \vee \cdots \\ &= \bigvee_{x_i \in U} P(x_i) \end{aligned}$$

2.3. Escritura de declaraciones

Una sentencia que afirma que todo bajo cierta categoría tiene una propiedad se traduce como:

$$\forall x(\_\_ \rightarrow \_\_)$$

donde el antecedente es una proposición verdadera únicamente si se cumple el criterio de la categoría. Si queremos expresar: "Todas las manzanas son malas" escribiríamos $\forall x(A(x) \rightarrow B(x))$ y para expresar "Todas las manzanas verdes son malas" escribiríamos: $\forall x(A(x) \wedge G(x) \rightarrow B(x))$.

Una sentencia que afirma que algún objeto u objetos bajo cierta categoría tienen una propiedad se traduce como:

$$\exists x(\_\_ \wedge \_\_)$$

por ejemplo, "Algunas manzanas son malas" se escribe como $\exists x(A(x) \wedge B(x))$.

Se debe tener cuidado de no confundir los dos patrones, por ejemplo: $\forall x(A(x) \wedge B(x))$ se traduce como "Todo es una manzana y es malo", que es una aserción

mucho más fuerte. De manera similar, $\exists x(A(x) \to B(x))$ se traduce como "Hay algo que es malo, si es una manzana".

Ejemplos:

1. Cada entero es un número racional: $\forall x(x \in \mathbb{Z} \to x \in \mathbb{Q})$

2. No hay un número racional x tal que $x^2 = 2$: $\forall x(x \in \mathbb{Q} \to x^2 \neq 2) \equiv \neg\exists x(x \in \mathbb{Q} \wedge x^2 = 2)$

3. Para números reales x y y, existe un número real z tal que $z^2 = x^2 + y^2$: $\forall x \forall y\{x \in \mathbb{R} \wedge y \in \mathbb{R} \to \exists z[\mathbb{R}(z) \wedge z^2 = x^2 + y^2]\}$

4. Cada entero par positivo mayor que 2 es la suma de dos primos: $\forall x\{x \in \mathbb{Z} \wedge (x/2 \in \mathbb{Z}) \wedge (x > 2) \to \exists y, z(y \in \mathbb{P} \wedge z \in \mathbb{P} \wedge x = y + z)\}$

5. Cada entero puede ser expresado como la suma de cuatro cuadrados: $\forall x\{x \in \mathbb{Z} \to \exists q, r, s, t(x = q^2 + r^2 + s^2 + t^2)\}$

6. Algunos enteros pueden ser expresados como la suma de tres cuadrados: $\exists x\{x \in \mathbb{Z} \wedge \exists q, r, s(x = q^2 + r^2 + s^2)\}$

7. Un entero $n > 1$ es primo si 1 y n son sus únicos divisores. Primero expresemos la declaración en otras palabras: "Si 1 y n son los únicos divisores de n, $n > 1$ y n es entero entonces n es primo". Que nuevamente puede ser expresado como: "Para todo n, si n es entero, $n > 1$ y no existe un x entero diferente de 1 y n tal que n/x sea entero entonces n es primo". $\forall n\{[n \in \mathbb{Z} \wedge (n > 1) \wedge \neg\exists x(x \in \mathbb{Z} \wedge x \neq 1 \wedge x \neq n \wedge n/x \in \mathbb{Z})] \to n \in \mathbb{P}\}$

2.4. Propiedades de los cuantificadores

Los cuantificadores del mismo tipo pueden ser intercambiados y combinados sin cambiar el valor de verdad de las declaraciones:

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

Pero esto no puede ser hecho con cuantificadores de diferentes tipos.

Ejemplo 1: Para números reales a,b,c es sabido que si $a < b$ y $b < c$ entonces $a \leq c$. Esta propiedad de transitividad está dada por:

$$\forall a \forall b\{MENOR(a, b) \to \forall c[MENOR(b, c) \to MENOR(a, c)]\}$$

puesto que el orden de los primeros dos cuantificadores no importa y la variable c no aparece en el predicado $MENOR(a, b)$, la fórmula puede ser reescrita combinando todos los cuantificadores existenciales al frente:

$$(\forall a, b, c)\{MENOR(a, b) \to [MENOR(b, c) \to MENOR(a, c)]\}$$

Ejemplo 2: Considere el predicado $P(n,m) : n > m^2$ sobre $\mathbb{N} \times \mathbb{N}$. La proposición

$$\forall m \exists n(n > m^2)$$

es equivalente a

$$\forall m[\exists n(n > m^2)]$$

Consideremos la expresión $\exists n(n > m^2)$, donde n está atada pero m es libre. Esta proposición dice que existe un $n \in \mathbb{N}$ con $n > m^2$. Esto es verdad si escogemos $n = m^2 + 1$, por lo que la proposición $\exists n(n > m^2)$ es verdad y, además, es verdad para cada $m \in \mathbb{N}$. Por consecuencia, la proposición $\forall m \exists n(n > m^2)$ es verdad. Ahora si se intercambian los cuantificadores:

$$\exists n \forall m(n > m^2)$$

que es equivalente a $\exists n[\forall m(n > m^2)]$. Una vez más consideremos la expresión interna $\forall m(n > m^2)$. Esta proposición es falsa para $m = n$ y por lo tanto $\exists n \forall m(n > m^2)$ es falso.

Ya hemos visto las leyes para intercambiar cuantificadores idénticos, sin embargo existen otras leyes:

1. $\neg \forall x P(x) \equiv \exists x \neg P(x)$ Decir "no todas las x son P" es equivalente a decir "existe un x que no es P".

2. $\neg \exists x P(x) \equiv \forall x \neg P(x)$ Decir "no existe una x que sea P" es equivalente a decir "toda x no es P".

3. $[\forall x P(x)] \wedge [\forall x Q(x)] \equiv \forall x[P(x) \wedge Q(x)]$ Decir "todo x es P y todo x es Q" equivale a decir "todo x es P y Q".

4. $[\exists x P(x)] \vee [\exists x Q(x)] \equiv \exists x[P(x) \vee Q(x)]$ Decir "existe un x que es P o existe un x que es Q" equivale a decir "existe un x que es P o Q".

2.5. Ejercicios

1. Traduzca a la notación de predicados y cuantificadores:

   a) "Si un jugador de baseball es pitcher entonces no es buen bateador"

   b) "Todos los jugadores de futbol mexicano admiran a algún portero europeo"

2. Sea $L(x,y)$: "x quiere a y", escriba con predicados y cuantificadores:

   a) Todos quieren a Jaime.

   b) Todo el mundo quiere a alguien.

   c) Hay alguien a quien todo mundo quiere.

   d) Hay alguien a quien Lidia no quiere.

3. Sea $Q(x, y)$: "x ha enviado un e-mail a y", y sea el dominio de x e y los estudiantes de la clase. Escriba en lenguaje natural:

   a) $\exists x \exists y \; Q(x, y)$

   b) $\exists x \forall y \; Q(x, y)$

   c) $\forall x \exists y \; Q(x, y)$

   d) $\exists y \forall x \; Q(x, y)$

   e) $\forall y \exists x \; Q(x, y)$

   f) $\forall x \forall y \; Q(x, y)$

Capítulo 3

Reglas de Inferencia

3.1. Inferencia en Lógica Proposicional

En lógica proposicional, utilizamos reglas de inferencia para deducir proposiciones verdaderas de aquellas que se saben son verdad. Utilizamos $A \Rightarrow B$ para indicar que B es verdadero siempre y cuando A sea verdadero.

Modus Ponens

$$P \wedge (P \to Q) \quad \Rightarrow \quad Q$$

Modus Tollens

$$\neg Q \wedge (P \to Q) \quad \Rightarrow \quad \neg P$$

Adición disyuntiva

$$P \quad \Rightarrow \quad P \vee Q$$

Simplificación conjuntiva

$$P \wedge Q \quad \Rightarrow \quad P$$

$$P \wedge Q \quad \Rightarrow \quad Q$$

Simplificación disyuntiva

$$(P \vee Q) \wedge \neg Q \quad \Rightarrow \quad P$$

$$(P \vee Q) \wedge \neg P \quad \Rightarrow \quad Q$$

Regla de la cadena

$$(P \to Q) \wedge (Q \to R) \quad \Rightarrow \quad P \to R$$

Tautologías

$$P \to (Q \to P)$$

$$P \to (Q \to R) \to ((P \to Q) \to (P \to R))$$

$$(\neg Q \to \neg P) \to (P \to Q)$$

Estas reglas no son equivalencias, meramente son proposiciones que siempre serán válidas bajo ciertas circunstancias. Para ilustrar esto, en la tabla 3.1 analizamos el Modus Ponens.

P	Q	$P \to Q$	$P \wedge (P \to Q)$
0	0	1	0
0	1	1	0
1	0	0	0
1	**1**	1	**1**

Cuadro 3.1: Tabla de verdad para ilustrar cómo se cumple $P \wedge (P \to Q) \Rightarrow Q$

Note que cuando $P \wedge (P \to Q)$ es verdad Q también es verdadero. Cabe señalar que en un caso $P \wedge (P \to Q)$ es falso y Q es verdadero, este caso no es de nuestro interés, pues no se trata de equivalencias lógicas, meramente de poder inferir valores de verdad.

Ahora analicemos la regla de la cadena:

P	Q	R	$P \to Q$	$Q \to R$	$(P \to Q) \wedge (Q \to R)$	$P \to R$
0	0	0	1	1	**1**	**1**
0	0	1	1	1	**1**	**1**
0	1	0	1	0	0	1
0	1	1	1	1	**1**	**1**
1	0	0	0	1	0	0
1	0	1	0	1	0	1
1	1	0	1	0	0	0
1	1	1	1	1	**1**	**1**

Nuevamente, se puede observar como siempre que $(P \to Q) \wedge (Q \to R)$ es verdadero, $P \to R$ es verdadero también.

3.2. Inferencia con Expresiones Cuantificadas

1. Generalización Universal: Si escogemos un elemento arbitrario c del dominio U y probamos $P(c)$, entonces podemos inferir $\forall x P(x)$. Por ejemplo $(x+2)^2 = x^2 + 4x + 4$ es verdad para cualquier real. No hay restricciones en la forma de escoger x, y por lo tanto podemos inferir que $\forall x[(x+2)^2 = x^2 + 4x + 4]$.

2. Generalización Existencial: Si podemos probar que $P(c)$ es verdad para algún c en el universo U, entonces podemos inferir $\exists x P(x)$. De manera abreviada, podemos expresar esta regla como $P(c) \Rightarrow \exists x P(x)$

3. Especificación Universal: De la declaración $\forall x P(x)$ podemos inferir $P(c)$ para cada c en el universo. Por ejemplo, de "cada entero es un racional" podemos inferir "2 es un número racional". De manera abreviada, podemos expresar esta regla como $\forall x P(x) \Rightarrow P(c)$

4. Especificación Existencial: De la declaración $\exists x P(x)$ podemos inferir que es posible escoger c en el universo tal que $P(c)$ es verdadero.

5. $[\forall x P(x)] \vee [\forall x Q(x)] \Rightarrow \forall x[P(x) \vee Q(x)]$ Decir "todo x es P o todo x es Q" implica "todo x es P o Q". Esta ley no puede aplicarse en el otro sentido.

6. $\exists x[P(x) \wedge Q(x)] \Rightarrow [\exists x P(x)] \wedge [\exists x Q(x)]$ Decir "existe un x que es P y Q" implica "existe un x que es P y existe un x que es Q".

7. En un predicado de dos lugares $\exists x \forall y P(x, y) \Rightarrow \forall y \exists x P(x, y)$

3.3. Argumentos válidos

Un patrón general de inferencia o argumento es usualmente presentado como una serie de declaraciones $P_1, P_2, \ldots, P_n$ seguidos de una conclusión Q. Las proposiciones $P_1, P_2, \ldots, P_n$ son llamadas *premisas* y Q es llamado *consecuencia*. El argumento $P_1 \wedge P_2 \wedge \cdots \wedge P_n \Rightarrow Q$ es válido si y sólo si $P_1 \wedge P_2 \wedge \cdots \wedge P_n \rightarrow Q$ es una tautología. Un argumento que no es válido se conoce como *falacia*.

En otras palabras, para que un argumento sea válido es necesario que cuando todas las premisas sean verdaderas, la consecuencia también lo sea.

3.4. Ejemplos de Falacias

La construcción de argumentos en lenguaje natural comúnmente incluye falacias, algunas de estas son tan sutiles que pueden aceptarse como válidas. En esta sección veremos algunas de las falacias más comunes.

Afirmando la consecuencia: $(P \rightarrow Q) \wedge Q \Rightarrow P$

Esta falacia es muy similar al Modus Ponens, pero en este caso se sabe que la consecuencia Q es verdadero y por lo tanto se asume que el antecedente P es verdadero también.

Un ejemplo de este tipo de falacias es: "Si llueve el suelo se moja. El suelo está mojado, por lo tanto llovió". La falla en el argumento es fácil de encontrar, pues el suelo puede estar mojado por muchas razones, incluso por la lluvia. Sin embargo, la lluvia no es la única causa por la que el suelo se moja.

Negación del antecedente: $(P \rightarrow Q) \wedge \neg P \Rightarrow \neg Q$

Muy similar al caso anterior, pero ahora se pretende inferir que una consecuencia es falsa mediante la negación del antecedente. Esto es claramente incorrecto puesto que se asume que $P \rightarrow Q$ implica que $\neg P \rightarrow \neg Q$, pero esto no es así.

Un ejemplo es: "Si está lloviendo está nublado. No está lloviendo, por lo tanto no está nublado".

Ad Hominem

En este tipo de falacia pretende desacreditar la validez de un argumento ligándolo a una característica o creencia de la persona que lo propone. "La persona A propone el argumento a, y la persona B afirma que A tiene una característica indeseable, y en base a esto, concluye que el argumento a debe ser inválido".

Este tipo de falacias es utilizado para evitar discutir la validez de un argumento, y en su lugar realizar un "ataque personal". En la Jerarquía del Desacuerdo de Graham, las respuestas *Ad Hominem*, son consideradas muy comunes, de poco o nulo valor argumentativo y marginalmente diferentes de un insulto directo.

Simplificación Causal

En esta falacia se asume que existe una única y simple causa que produce un resultado cuando en realidad pueden existir otras. Se reduce lógicamente a decir que $P \to Q$, por lo tanto, P es la única causa por la que Q puede suceder.

Un ejemplo es: "Los impuestos sirven para tener policia, carreteras y escuelas, esto demuestra la necesidad de los impuestos".

Capítulo 4

Métodos de Demostración

Un patrón general de inferencia o argumento es usualmente presentado como una serie de declaraciones $P_1, P_2, \ldots, P_n$ seguidos de una conclusión Q. Las proposiciones $P_1, P_2, \ldots, P_n$ son llamadas *premisas* y Q es llamado *consecuencia*. El argumento $P_1 \wedge P_2 \wedge \cdots \wedge P_n \Rightarrow Q$ es válido si y sólo si $P_1 \wedge P_2 \wedge \cdots \wedge P_n \rightarrow Q$ es una tautología. Un argumento que no es válido se conoce como *falacia*.

En otras palabras, para que un argumento sea válido es necesario que cuando todas las premisas sean verdaderas, la consecuencia también lo sea.

4.1. Prueba por Tabla de Verdad

Dado un argumento $P_1 \wedge P_2 \wedge \cdots \wedge P_n \Rightarrow Q$, es posible demostrar su validez mediante la tabla de verdad de $P_1 \wedge P_2 \wedge \cdots \wedge P_n \rightarrow Q$. Se verifica si se trata de una tautología o no, y en base a esto, se determina si el argumento es válido o falaz.

4.2. Prueba Directa

La prueba directa consiste en inferir la conclusión del argumento utilizando únicamente las premisas, reglas de inferencia, equivalencias lógicas o tautologías. La prueba directa, por sí misma consiste en develar el proceso lógico que lleva a la consecuencia.

Para probar si un argumento $P \Rightarrow Q$ es válido:

1. Se sustituye P por una secuencia de declaraciones $P_1, P_2, \ldots, P_n$, donde cada P_i está en P o es una tautología,

2. o puede ser derivado de declaraciones P_j, P_k anteriores $(j, k < i)$ por medio de reglas de inferencia o de equivalencia lógica.

3. La prueba está completa cuando se encuentra que $P_i \equiv Q$

Ejemplo 1. Probar la declaración $[P \to (Q \to R)] \Rightarrow [Q \to (P \to R)]$:

1.	$P \to (Q \to R)$	Premisa
2.	$[P \to (Q \to R)] \to [(P \to Q) \to (P \to R)]$	Tautología
3.	$(P \to Q) \to (P \to R)$	Modus Ponen 1,2
4.	$Q \to (P \to Q)$	Tautología
5.	$Q \to (P \to R)$	Regla de la cadena 4,3

Ejemplo 2. Demostrar $P \Rightarrow P \to P$:

1.	P	Premisa
2.	$P \to (P \to P)$	Tautología
3.	$P \to P$	Modus Ponens 1,2

Ejemplo 3. *Estoy cansado o estoy enfermo. Si estoy enfermo me voy a mi casa. No me voy a mi casa. Entonces estoy cansado.* Suponemos que los primeras tres declaraciones son verdaderas, queremos comprobar la verdad de la última declaración, que es la consecuencia. Denotemos "estoy cansado" con P, "estoy enfermo" con Q, y "me voy a mi casa" con R. La secuencia de declaraciones se convierte en:

$$\begin{array}{c} P \vee Q \\ Q \to R \\ \neg R \\ P \end{array}$$

y el argumento a probar es $(P \vee Q) \wedge (Q \to R) \wedge \neg R \Rightarrow P$.

1.	$P \vee Q$	Premisa
2.	$Q \to R$	Premisa
3.	$\neg R$	Premisa
4.	$\neg R \to \neg Q$	Implicación 2
5.	$\neg Q$	Modus Ponens 3,4
6.	P	Simplificación disyuntiva 1,5

Ejercicios

Demuestre por tabla de verdad y prueba directa los siguientes argumentos:

1. $\neg P \vee Q,\ S \vee P,\ \neg Q \Rightarrow S$
2. $P \wedge Q,\ Q \to R,\ S \to P \Rightarrow R \vee S$
3. $A \to B,\ \neg(B \vee C) \Rightarrow \neg A$
4. $P \to Q,\ \neg P \to \neg R,\ R \wedge \neg S \Rightarrow Q$
5. $P \to R,\ Q \to S,\ P \vee Q \Rightarrow S \vee R$
6. Si bajan los impuestos se eleva el ingreso. El ingreso se elevó, por lo tanto, los impuestos bajaron.

4.3. Pruebas Vacuas

Toda implicación es verdadera cuando la premisa es falsa, por lo tanto, si es posible demostrar que P es falso en $P \Rightarrow Q$, el argumento es válido.

Ejemplo 1. *Si* $0 > 1$ *entonces* $0^2 > 0$, usemos P para denotar la proposición $0 > 1$ y Q para denotar $0^2 > 0$. Entonces la prueba consiste en demostrar que $P \Rightarrow Q$.

Puesto que P es evidentemente falso y siempre que la premisa es falsa la implicación es verdadera, se demuestra por prueba vacua que $P \Rightarrow Q$, es decir *Si* $0 > 1$ *entonces* $0^2 > 0$, es un argumento válido.

4.4. Pruebas Triviales

Si se tiene una implicación y se conoce que la consecuencia es verdadera, entonces la implicación es verdadera. La prueba trivial consiste en demostrar que en $P \Rightarrow Q$, Q es verdadero.

Ejemplo 1. *Si* $a \geq b$ *entonces* $a^0 \geq b^0$. Puesto que $a^0 = b^0 = 1$, se tiene que la consecuencia es verdad y por lo tanto queda demostrado que *Si* $a \geq b$ *entonces* $a^0 \geq b^0$ es un argumento válido.

4.5. Prueba por Contradicción

Considere un teorema $P \Rightarrow Q$, donde P representa las premisas $P_1 \wedge P_2 \wedge \cdots \wedge P_n$. Este método de prueba está basado en la equivalencia $P \rightarrow Q \equiv \neg P \vee Q \equiv \neg(P \wedge \neg Q)$. Lo que indica que si $P \Rightarrow Q$, entonces $P \wedge \neg Q$ es siempre falso. Esto indica que un método de prueba válido es negar la consecuencia del teorema e incluir esta negación a las premisas. Si una contradicción puede ser implicada desde este conjunto de proposiciones, la prueba está completa.

Ejemplo 1. Probar la declaración $P \rightarrow R, Q \rightarrow S, P \vee Q \Rightarrow S \vee R$ por contradicción:

1.	$P \rightarrow R$	Premisa
2.	$Q \rightarrow S$	Premisa
3.	$P \vee Q$	Premisa
4.	$\neg(S \vee R)$	Consecuencia negada
5.	$\neg S \wedge \neg R$	De Morgan 4
6.	$\neg S$	Simplificación conjuntiva 5
7.	$\neg R$	Simplificación conjuntiva 5
8.	$\neg Q$	Modus Tollens 6,2
9.	$\neg P$	Modus Tollens 7,1
10.	P	Simplificación disyuntiva 3,8
11.	$P \wedge \neg P$	Conjunción 10,9

y dado que $P \wedge \neg P$ es una contradicción, el argumento es válido.

Ejemplo 2. Probar $P \vee Q,\ Q \to R,\ \neg R \Rightarrow P$ por contradicción:

1.	$P \vee Q$	Premisa
2.	$Q \to R$	Premisa
3.	$\neg R$	Premisa
4.	$\neg P$	Consecuencia negada
5.	Q	Simplificación disyuntiva 1,4
6.	R	Modus Ponens 5,2
7.	$R \wedge \neg R$	Conjunción 3,6

por lo que el argumento es válido (prueba por contradicción).

Ejercicios

Demostrar por contradicción los siguientes argumentos:

1. $P \to Q \vee R,\ Q \to R,\ R \to S \Rightarrow P \to S$
2. $(P \to Q) \wedge (R \to S),\ (Q \to T) \wedge (S \to U),\ \neg(T \wedge U),\ P \to R \Rightarrow \neg P$
3. $P \to Q,\ R \to P \wedge S,\ Q \wedge S \to P \wedge T,\ \neg T \Rightarrow P \to \neg R$

4.6. Prueba por Casos

Considere un argumento de la forma $P_1 \vee P_2 \vee \ldots \vee P_n \Rightarrow Q$, es decir donde se tiene una disyunción de premisas en lugar de una conjunción de éstas. Para demostrar argumentos de este tipo, es posible utilizar la equivalencia lógica:

$$P_1 \vee P_2 \vee \ldots \vee P_n \to Q \equiv (P_1 \to Q) \wedge (P_2 \to Q) \wedge \ldots \wedge (P_n \to Q)$$

Por lo que es posible demostrar que el argumento es válido probando que todos los casos $P_i \Rightarrow Q$ son argumentos válidos. La demostración de cada subargumento, puede hacerse por alguno de los métodos de demostración ya expuestos.

Ejemplo 1. Probar la declaración $P \to R, Q \to S, P \vee Q \Rightarrow S \vee R$ por casos. Lo primero que se debe observar, es que el argumento no es propiamente una disyunción de premisas. Sin embargo, la premisa $P \vee Q$ sí lo es, por lo que el argumento puede ser reescrito de la siguiente manera:

$$\underbrace{(P \to R) \wedge (Q \to S) \wedge P}_{\text{Caso } P} \vee \underbrace{(P \to R) \wedge (Q \to S) \wedge Q}_{\text{Caso } Q} \Rightarrow S \vee R$$

y bajo este escenario, existen dos casos o subargumentos a probar.

Caso P:

1.	$P \to R$	Premisa
2.	$Q \to S$	Premisa
3.	P	Caso
4.	R	Modus Ponens 3,1
5.	$S \vee R$	Adición disyuntiva 4

por lo que el caso P es un argumento válido (prueba directa).

Caso Q:

1.	$P \to R$	Premisa
2.	$Q \to S$	Premisa
3.	Q	Caso
4.	S	Modus Ponens 3,2
5.	$S \vee R$	Adición disyuntiva 4

por lo que el caso Q es un argumento válido (prueba directa).

Puesto que todos los subargumentos son válidos, queda demostrado que el argumento es válido (prueba por casos).

Ejercicios

1. $(P \to Q) \vee (P \to R),\ P \Rightarrow Q \vee R$
2. Si los Rocos ganan entonces celebran en Morelia y si Polos gana entonces celebran en Guadalajara. Alguno de los dos gana. Si Rocos gana entonces no celebran en Guadalajara. Si Polos gana entonces no celebran en Morelia. Por lo tanto, en Morelia celebran si y sólo si en Guadalajara no celebran.

4.7. Argumentos con Predicados y Cuantificadores

Una fórmula bien formada se dice *cerrada* si todas las variables en la fórmula están cuantificadas, en caso contrario se dice que está *abierta*. Un predicado con argumentos constantes es una proposición, también llamada formula *atómica cerrada*. Una fórmula atómica cerrada es un *hecho* si es verdad.

Existen cuatro reglas fundamentales en un esquema de inferencia de fórmulas cuantificadas:

1. Generalización Universal: Si escogemos un elemento arbitrario c del dominio U y probamos $P(c)$, entonces podemos inferir $\forall x P(x)$. Por ejemplo $(x+2)^2 = x^2 + 4x + 4$ es verdad para cualquier real. No hay restricciones en la forma de escoger x, y por lo tanto podemos inferir que $\forall x[(x+2)^2 = x^2 + 4x + 4]$.
2. Generalización Existencial: Si podemos probar que $P(c)$ es verdad para algún c en el universo U, entonces podemos inferir $\exists x P(x)$.

3. Especificación Universal: De la declaración $\forall x P(x)$ podemos inferir $P(c)$ para cada c en el universo. Por ejemplo, de "cada entero es un racional" podemos inferir "2 es un número racional".

4. Especificación Existencial: De la declaración $\exists x P(x)$ podemos inferir que es posible escoger c en el universo tal que $P(c)$ es verdadero.

Ejemplo 1. Considere las siguientes declaraciones:

1. Todas las personas inteligentes son nobles.

2. Todos son inteligentes o tontos.

3. Algunas personas no son tontas.

4. Por lo tanto, algunas personas son nobles.

definamos los predicados como:

- $I(x) : x$ es inteligente.
- $N(x) : x$ es noble.
- $T(x) : x$ es tonto.

En notación formal se desea probar: $\forall x[I(x) \rightarrow N(x)], \forall x[I(x) \vee T(x)]$, $\exists x[\neg T(x)] \Rightarrow \exists x[N(x)]$, asumiendo que el universo de discurso es de personas.

1.	$\forall x[I(x) \rightarrow N(x)]$	Premisa
2.	$\forall x[I(x) \vee T(x)]$	Premisa
3.	$\exists x[\neg T(x)]$	Premisa
4.	$\neg T(c)$	Especificación existencial 3
5.	$I(c) \vee T(c)$	Especificación universal 2
6.	$I(c)$	Simplificación disyuntiva 4,5
7.	$I(c) \rightarrow N(c)$	Especificación universal 1
8.	$N(c)$	Modus Ponens 6,7
9.	$\exists x[N(x)]$	Generalización existencial 8

con lo cual se comprueba que el argumento es válido (prueba directa).

Ejemplo 2. Considere los siguientes hechos y reglas:

1. Jorge y Karla son miembros del Club ABC.

2. Jorge está casado con María.

3. Juan es un hermano de María y está casado con Karla.

4. Jorge y Juan se reúnen en casa de Jorge.

5. El cónyuge de cada persona en el club ABC es también miembro del club.

6. Las personas casadas viven juntas.

7. Si dos personas viven juntas y una es visitada, la otra también.

De esto queremos determinar la verdad de las siguientes declaraciones:

a) Jorge, Karla, Juan y María son miembros del club ABC.

b) Juan visita la casa de su hermana.

definamos los predicados como:

- $H(x)$: x es miembro del club ABC.
- $M(x, y)$: x está casado con y.
- $L(x, y)$: x y y viven juntos.
- $B(x, y)$: x es hermano de y.
- $V(x, y)$: x visita la casa de y.

Los hechos y reglas están representados con predicados de la siguiente forma:

1. $H(Jorge) \wedge H(Karla)$
2. $M(Jorge, Maria)$
3. $B(Juan, Maria) \wedge M(Juan, Karla)$
4. $V(Juan, Jorge)$
5. $(\forall x, y)[H(x) \wedge M(x, y) \rightarrow H(y)]$, $\forall x, y[H(y) \wedge M(x, y) \rightarrow H(x)]$
6. $(\forall x, y)[M(x, y) \rightarrow L(x, y)]$
7. $(\forall x, y, z)[V(x, y) \wedge L(y, z) \rightarrow V(x, z)]$

Para resolver "Jorge, Karla, Juan y María son miembros del club ABC.":

1.	$M(Jorge, Maria)$	Hecho
2.	$H(Jorge)$	Hecho
3.	$(\forall x, y)[H(x) \wedge M(x, y) \rightarrow H(y)]$	Regla
4.	$H(Jorge) \wedge M(Jorge, Maria) \rightarrow H(Maria)$	S.3 $x = Jorge, y = Maria$
5.	$H(Maria)$	Modus Ponens 1,2 y 4
6.	$H(Karla)$	Hecho
7.	$M(Juan, Karla)$	Hecho
8.	$(\forall x, y)[H(y) \wedge M(x, y) \rightarrow H(x)]$	Regla
9.	$H(Karla) \wedge M(Juan, Karla) \rightarrow H(Juan)$	S.8 $x = Juan, y = Karla$
10.	$H(Juan)$	Modus Ponens 6,7 y 9
11.	$H(Jorge) \wedge H(Karla) \wedge H(Juan) \wedge H(Maria)$	Conjunción 2,6,10,5

con lo cual se comprueba que el argumento es válido (prueba directa).

Para resolver "Juan visita la casa de su hermana."

1.	$(\forall x, y, z)[V(x, y) \wedge L(y, z) \rightarrow V(x, z)]$	Regla
2.	$M(Jorge, Maria)$	Hecho
3.	$V(Juan, Jorge)$	Hecho
4.	$B(Juan, Maria)$	Hecho
5.	$(\forall x, y)[M(x, y) \rightarrow L(x, y)]$	Regla
6.	$M(Jorge, Maria) \rightarrow L(Jorge, Maria)$	S.5 $x = Jorge, y = Maria$
7.	$L(Jorge, Maria)$	Modus Ponens 2,6
8.	$V(Juan, Jorge) \wedge L(Jorge, Maria) \rightarrow V(Juan, Maria)$	S.1 $x = Juan, y = Jorge, z = Maria$
9.	$V(Juan, Maria)$	Modus Ponens 3,7 y 8
10.	$B(Juan, Maria) \wedge V(Juan, Maria)$	Conjunción 4,9

con lo cual se comprueba que el argumento es válido (prueba directa).

Ejercicios

1. Ejercicios 9-12, 15 y 16, página 67 del libro de Rosen.

Bibliografía

[Aho 1995] Alfred V. Aho, Jeffrey D. Ullman *Foundations of Computer Science: C Edition*, W.H. Freeman and Company, 1995

[Alagar 1989] Alagar, Vangalur S., *Fundamentals of Computing: Theory and Practice*, Prentice Hall, 1989

[Doerr 1985] Doerr, Alan, *Applied Discrete Structures for Computer Science*, Science Research Associates, Inc., 1985

[Enderton 2000] Herbert B. Enderton, *A Mathematical Introduction to Logic*, Academic Press, 2000

[Hopcroft 1979] Hopcroft, John E., *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, 1979

[Knuth 1989] Knuth, Donald E.,*Concrete Mathematics*, 2nd Edition, Addison-Wesley, 1989

[Rosen 1999] Rosen, Kenneth H., *Discrete Mathematics and Its Applications*, 4th Edition, McGraw-Hill, 1999

[Tourlakis 1984] Tourlakis, George J., *Computability*, Reston Publishing Company, Inc., 1984