

MATEMÁTICAS DISCRETAS

Segunda edición

Favio Ezequiel Miranda Perea
Elisa Viso Gurovich

TEMAS DE COMPUTACIÓN



Favio E. Miranda
Elisa Viso G.

Matemáticas discretas

Facultad de Ciencias, UNAM



Miranda, Favio E.

Matemáticas discretas / Favio E. Miranda, Elisa Viso G.

--2a ed.-- México : UNAM, Facultad de Ciencias, 2016.

vi, 359 p. ; 22 cm.

Incluye índice

Bibliografía: p. 341

ISBN 978-607-02-8095-5

1. Lógica matemática. 2. Lógica matemática – Estudio y Enseñanza (Superior).

I. Viso Gurovich, Elisa. II. Universidad Nacional Autónoma de México.

Facultad de Ciencias. III. t.

511.3-scdd20

Biblioteca Nacional de México

Matemáticas discretas

1ª edición, 2 de marzo de 2010

2a edición, 16 de mayo de 2016

D.R. © 2016 Universidad Nacional Autónoma de México

Facultad de Ciencias.

Ciudad Universitaria. Delegación Coyoacán.

C. P. 04510, Ciudad de México.

editoriales@ciencias.unam.mx

ISBN: 978-607-02-8095-5

Diseño de portada: Laura Uribe

Diseño y tipografía de interiores: Elisa Viso

Prohibida la reproducción parcial o total de la obra por cualquier medio, sin la autorización por escrito del titular de los derechos patrimoniales

Impreso y hecho en México

Índice general

I	Lógica matemática	1
1.	Introducción	3
1.1.	Expresiones	4
1.2.	Mecanismos formales para expresiones	6
1.3.	Gramáticas y árboles de derivación	9
2.	Lógica proposicional	17
2.1.	El lenguaje de la lógica proposicional	17
2.1.1.	Argumentos lógicos	17
2.1.2.	Proposiciones	19
2.1.3.	Sintaxis de la lógica proposicional	22
2.1.4.	Semántica de la lógica proposicional	23
2.1.5.	Tautologías y contradicciones	31
2.1.6.	Argumentos correctos	33
2.2.	Evaluación de expresiones	38
2.2.1.	Estados y evaluación	38
2.2.2.	Precedencia y asociatividad	39
2.2.3.	Sustitución textual	41
2.3.	Análisis sintáctico de expresiones lógicas	45
2.3.1.	Esquemas	46
2.3.2.	Rango y conectivo principal	48
2.3.3.	Análisis de proposiciones compuestas	49
2.3.4.	Tautologías y sustitución	52
2.4.	Equivalencia lógica	55
2.4.1.	Razonamiento ecuacional	56
2.4.2.	Álgebra de equivalencias lógicas	60
2.5.	Conceptos semánticos importantes	67
2.5.1.	Interpretaciones	67
2.5.2.	Consecuencia lógica	69
2.6.	Análisis de argumentos	72
2.6.1.	Tablas de verdad	72

2.6.2.	Uso de interpretaciones	74
2.6.3.	Derivaciones	80
2.7.	Tableaux en cálculo proposicional	89
2.7.1.	El concepto de tableau	90
2.7.2.	Eliminación de ramas del tableau	93
2.7.3.	Reglas para los tableaux	95
2.7.4.	Modelo de una fórmula	98
2.7.5.	Algoritmos para la lógica proposicional	98
3.	Circuitos digitales	103
3.1.	Fórmulas proposicionales y circuitos digitales	104
3.1.1.	Costo de los circuitos lógicos	111
3.1.2.	Minimización de circuitos usando equivalencias lógicas	112
3.2.	Mapas de Karnaugh	118
3.2.1.	Mapas de Karnaugh de dos variables	118
3.2.2.	Mapas de Karnaugh con tres y cuatro variables	120
3.2.3.	Construcción del circuito digital mínimo	125
3.3.	Método de Quine-McCluskey para minimización	130
3.3.1.	Tabla de mintérminos	130
3.3.2.	Eliminación de variables combinando mintérminos	131
3.4.	Dispositivos digitales combinatorios	136
3.4.1.	Multiplexores	138
3.4.2.	Decodificadores binarios	140
3.4.3.	Retardos y tiempo total	140
3.5.	Circuitos digitales secuenciales	142
3.5.1.	Circuitos secuenciales asíncronos	144
3.5.2.	Pulsos de reloj	149
3.5.3.	Circuitos secuenciales síncronos	151
4.	Lógica de predicados	167
4.1.	Predicados	168
4.1.1.	Variables y cuantificadores	170
4.2.	Sintaxis de la lógica de predicados	171
4.2.1.	Términos	172
4.2.2.	Fórmulas	173
4.2.3.	Fórmulas cuantificadas	174
4.2.4.	Variables libres y ligadas	175
4.3.	Especificación formal	180
4.3.1.	Juicios aristotélicos	182
4.3.2.	Negaciones	184
4.3.3.	Contando objetos	185

4.3.4.	Micromundos	185
4.4.	Semántica informal	191
4.4.1.	Dominios de interpretación	191
4.4.2.	Noción informal de verdad	194
4.4.3.	Verdad en micromundos	196
4.4.4.	Algunas equivalencias lógicas	198
4.4.5.	Algunos argumentos correctos	203
4.5.	Predicados y tipos	207

II Inducción y recursión 211

5.	Inducción y recursión	213
5.1.	Los números naturales	214
5.1.1.	Axiomas de Peano	215
5.2.	Inducción en los números naturales	216
5.2.1.	Cambio de la base de la inducción	219
5.2.2.	Inducción completa	221
5.3.	Definiciones recursivas	227
5.3.1.	Definición de funciones recursivas	230
5.4.	Inducción estructural	236
5.4.1.	Inducción en listas	237
5.4.2.	Inducción en fórmulas	239
5.4.3.	Inducción en árboles	242

III Relaciones 249

6.	Relaciones	250
6.1.	Producto cartesiano	250
6.2.	Relaciones	255
6.2.1.	Operaciones con relaciones	256
6.3.	Relaciones binarias	258
6.3.1.	Representación de relaciones binarias	260
6.3.2.	Operaciones con relaciones binarias	264
6.3.3.	Propiedades de las relaciones binarias	266
6.3.4.	Operaciones de cerradura	270
6.4.	Relaciones de equivalencia	282
6.4.1.	Clases de equivalencia	285
6.4.2.	Estudio de un caso: conjuntos mediante listas	289
6.5.	Relaciones de orden	297
6.5.1.	Diagramas de Hasse	303

6.6. Elementos extremos 308

6.6.1. Órdenes totales y ordenamiento topológico 310

6.6.2. Cotas 313

6.7. Latices 319

6.8. Bases de datos relacionales 328

Índice analítico **343**

Parte I

Lógica matemática

Introducción | 1

El libro está dividido fundamentalmente en tres partes: *Lógica matemática*, *Inducción y recursión*, y *Relaciones* (sobre todo de orden). De inducción y de recursión tal vez no hemos oído, pero de lógica y relaciones sí; todos conocemos el significado de “relación” –como la familiar, la de amigos, la de compañeros– que agrupa a individuos respecto a algo en común (una característica o propiedad); el término *lógica* lo usamos de manera bastante liberal en nuestra vida diaria en frases como las que siguen:

- No es lógico lo que estás diciendo.
- No entiendo la lógica de este asunto.
- Presentas un argumento que no es coherente.
- Es falso lo que estás suponiendo.

Todos nosotros sabemos que existe más precisión cuando estamos en el terreno matemático que cuando estamos hablando de experiencias de la vida común. Realmente, en el lenguaje *natural*¹ dejamos mucho a la subjetividad de los hablantes y al contexto que se supone conocen ambos. Decimos que este tipo de lenguaje es *informal*, mientras que el lenguaje que se usa en matemáticas o los lenguajes de programación son *lenguajes formales*.

Distinguimos entre un objeto informal y uno formal porque este último está claramente definido y especificado por un conjunto de reglas. Uno de los atractivos del formalismo es el poder expresar ideas de forma concreta, breve y precisa. Pero no sólo nos interesan estos aspectos del formalismo sino su aplicación, la cual nos obliga a formalizar nuevas ideas o experiencias y, en este proceso, precisar y encontrar contradicciones que pudieran

¹Llamaremos así al lenguaje que habla cualquier ser humano, en nuestro caso el español.

causar mucho daño, como en el caso de un programa de computadora que pudiese contener ambigüedades.

Si nos referimos a un objeto de manera formal, podemos construir un *modelo* de ese objeto. Algunas de las ventajas de los modelos matemáticos son las siguientes:

- Un modelo matemático es, por lo general, más preciso, entendible, conciso y riguroso que una descripción informal escrita en lenguaje natural.
- A través de un modelo matemático podemos calcular directamente respuestas a problemas sobre el objeto modelado.
- Las matemáticas, en particular la lógica, nos proporcionan métodos de razonamiento: para manipular objetos, para demostrar propiedades de y sobre objetos, y para obtener resultados nuevos a partir de resultados ya conocidos, lo cual genera una extensión del conocimiento.

Este último punto es, tal vez, uno de los aspectos más importantes de los modelos matemáticos, que tiene una enorme utilidad en ciencias de la computación: tener la seguridad científica de que algo funciona como esperamos; poder extender las posibilidades de una computadora, un lenguaje de programación o un algoritmo para usos distintos a los que fue creado; en fin, para poder continuar con el impresionante desarrollo que han tenido las ciencias de la computación en este siglo.

1.1. Expresiones

La lógica, en especial la lógica matemática, juega un papel muy importante en el desarrollo de las matemáticas en general y de las ciencias de la computación en particular. En el ámbito de las ciencias de la computación es importante distinguir entre argumentos válidos o inválidos; es decir, entre aquellos que son sólidos lógicamente hablando y los que no lo son.

La lógica presenta ciertos elementos de estudio que no son tan distintos a los que estamos acostumbrados en matemáticas. Durante muchos años hemos manipulado expresiones numéricas con incógnitas; es decir, ecuaciones con variables y constantes, para obtener un resultado final. Mientras que en el álgebra estamos trabajando con números fundamentalmente, en lógica trabajamos con *proposiciones lógicas* o simplemente *proposiciones*. Al igual que en álgebra, utilizamos variables (*símbolos* que nos sirven para representar a los objetos elementales), constantes (true, false) y operadores, que también son símbolos pero con un significado especial.

Cuando tenemos una expresión aritmética hablamos de los operadores y operandos, entendiendo a los operadores como operaciones que se tienen que realizar, utilizando para ello a los operandos. Los operadores pueden ser: *unarios*, cuando utilizan o actúan sobre

un único operando; *binarios*, cuando actúan sobre dos operandos y, en general, *n-arios*² cuando actúan sobre n operandos. Entre los operadores unarios aritméticos que conocemos está el signo de menos $\boxed{-}$ y en algunas ocasiones también podemos considerar el signo de más $\boxed{+}$ (nos tendremos que poner de acuerdo, antes de empezar, a cuál aceptamos y a cuál no). Entre los operadores binarios podemos mencionar a la multiplicación y a la división (que las podemos representar con $\boxed{\cdot}$, $\boxed{\times}$ o con $\boxed{*}$ como se acostumbra en los lenguajes de programación a la primera; y con $\boxed{\div}$ o $\boxed{/}$ a la segunda). Un ejemplo de operador ternario puede ser $\boxed{\text{entre}(a, b, c)}$, que decide cuál de los tres números, a , b o c , se encuentra entre los otros dos; o el operador $\boxed{\text{raíces}(a, b, c)}$, que devuelve las raíces del polinomio cuadrático $ax^2 + bx + c$.

Hay tres estilos para escribir expresiones, los cuales se distinguen por cómo se coloca al operador en relación a sus operandos. Si el operador es unario o n -ario sólo tenemos dos opciones:

Notación prefija: El operador se coloca antes del operando

$$-a \quad (+ \ a \ b \ c)$$

Notación sufija o polaca: El operador se coloca después del operando

$$a \uparrow \qquad (a \ b \ *)$$

apuntador en Pascal

Si el operador es binario, además de estas dos formas tenemos la que es más usual:

Notación infija: Es aquella donde el operador se encuentra *entre* sus operandos.

$$a + b \quad 3 \cdot (7 + 5)$$

Las diferencias entre estas tres notaciones no son nada más de forma, pues cada una de ellas tiene propiedades particulares que veremos después. Por lo pronto trabajaremos con la notación infija que, como ya mencionamos, es la más usual.

Estas maneras de escribir expresiones tienen que ver con su *sintaxis*, término que se refiere a la *forma* que deben tener las cadenas de letras y símbolos para que califiquen como expresiones bien construidas. Aún no hemos hablado del *significado* de una expresión, aunque pronto lo haremos. Los aspectos relacionados con el significado conforman la *semántica* de las expresiones.

Ejercicios

1.1.1.- Mencione tres relaciones de su vida cotidiana.

1.1.2.- ¿Cuál cree usted que es la diferencia entre la lógica (coloquial) y la lógica matemática?

²Se lee “enarrio”.

1.2. Mecanismos formales para describir expresiones

Cuando describimos una expresión aritmética podemos hacerlo de varias maneras. Una de ellas es dando tantos ejemplos como podamos y dejando al lector que encuentre patrones que describan al mayor número posible de expresiones. Es claro que con este método no vamos a poder describir a todas las expresiones aritméticas posibles, ya que tenemos un número infinito de ellas. Una segunda manera es dando reglas para *construir* expresiones aritméticas correctas. Estas reglas de formación pertenecen, queremos insistir, a la sintaxis de las expresiones aritméticas.

A continuación formalizamos reglas para construir expresiones aritméticas sencillas, para posteriormente contrastar con lo que es una expresión lógica:

Definición 1.1 Una *expresión aritmética* es alguna de las que siguen:

1. Un *objeto elemental*: un número (una constante) o una variable.
2. Si E es una expresión aritmética, (E) es una expresión aritmética.
3. Si \triangleright es un operador unario y E es una expresión aritmética, entonces $\triangleright E$ es una expresión aritmética.
4. Si \diamond es un operador binario infijo y E y F son dos expresiones aritméticas, entonces $E \diamond F$ es una expresión aritmética.
5. Si \star es un operador n -ario y E_1, E_2, \dots, E_n son expresiones aritméticas, entonces $\star(E_1, E_2, \dots, E_n)$ es una expresión aritmética.
6. Éstas, y sólo éstas, son expresiones aritméticas válidas.

A primera vista esta definición parece incorrecta pues en algunas partes se utiliza el mismo concepto de expresión que se está definiendo. Esta clase de definiciones, llamadas definiciones recursivas, son omnipresentes en ciencias de la computación. Más adelante las estudiaremos con detalle. Estamos suponiendo que sabemos cuáles son los operadores unarios: $+$ (positivo), $-$ (negativo); cuáles los binarios: $-$ (resta), $+$ (suma), \times , \cdot , $*$ (multiplicación), \div , $**$ (\wedge , exponenciación); y conocemos algunos n -arios:

$$f(x_1, x_2, \dots), \max(\dots), \min(\dots) \dots$$

Veamos a continuación algunos ejemplos de construcción de expresiones aritméticas enfatizando su proceso de construcción.

Ejemplo 1.1. Cada uno de los siguientes es un objeto elemental:

$a \quad i \quad x \quad 3.0 \quad 1$

Por lo tanto también son expresiones aritméticas.

Ejemplo 1.2. Considérese los siguientes operadores unarios, que representan el signo de un número en aritmética: $+$ y $-$:

-17 $-$ reemplaza \triangleright y 17 es una expresión, por ser un número.

$+a$ $+$ reemplaza $a \triangleright$ y a es una expresión, por ser una variable.

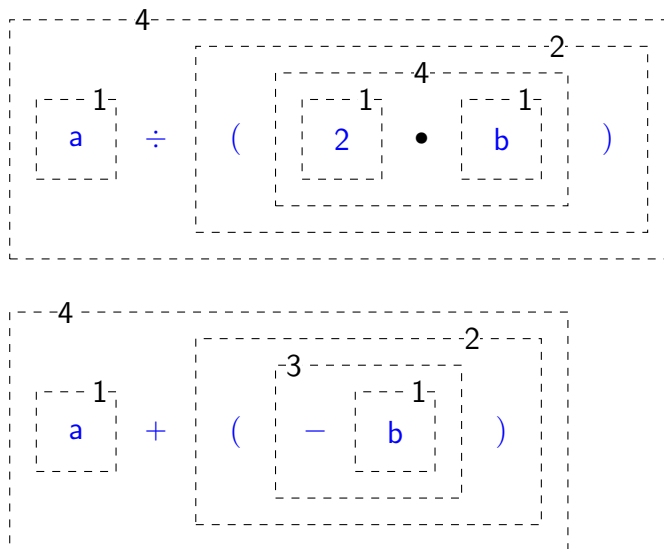
Podemos también ir encerrando en cuadrados contenidos uno dentro del otro, con una anotación de la regla a la que corresponden, a las distintas expresiones que conforman la expresión original. Cada rectángulo encierra a una expresión. Veamos a continuación:



Ejemplo 1.3. Considérese los operadores binarios \cdot y \div .

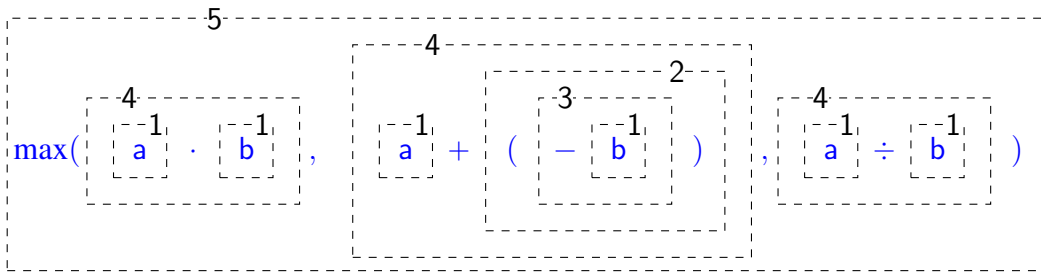
$a \div (2 \cdot b)$ \div y \cdot son operadores binarios; a y $(2 \cdot b)$ son expresiones.

$a + (-b)$ $+$ es un operador binario; a y $(-b)$ son expresiones.

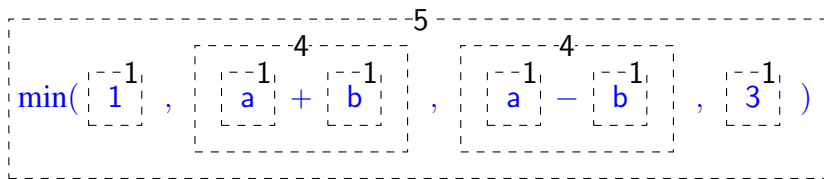


Ejemplo 1.4. Supongamos que tenemos dos operadores, máx y mín .

$\text{máx}(a \cdot b, a + (-b), a \div b)$ máx es un operador n -ario con $n = 3$; $a \cdot b$ es una expresión; $a + (-b)$ es una expresión. $a \div b$ es una expresión.

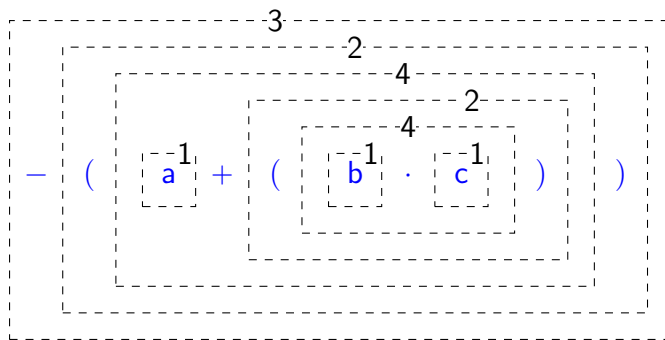


$\min(1, \max(a + b, a - b, 3))$ \min es un operador n -ario con $n = 2$, es decir, binario; 1 es una expresión; $\max(a + b, a - b, 3)$ es una expresión.



Ejemplo 1.5. La expresión $-(a + (b \cdot c))$ es una expresión aritmética porque:

- Como b y c son expresiones, por ser variables y \cdot es un operador binario, entonces $b \cdot c$ es una expresión.
- Como $b \cdot c$ es una expresión, entonces $(b \cdot c)$ es una expresión.
- Como a y $(b \cdot c)$ son expresiones y $+$ es un operador binario entonces $a + (b \cdot c)$ es una expresión.
- Como $a + (b \cdot c)$ es una expresión, entonces $(a + (b \cdot c))$ es una expresión.
- Como $(a + (b \cdot c))$ es una expresión y $-$ es un operador unario, entonces $-(a + (b \cdot c))$ es una expresión.



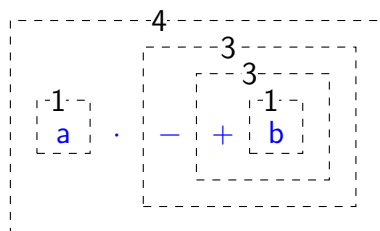
Ejemplo 1.6. Observemos la expresión $(a \cdot b) - (4 \cdot a \cot b - 2 \cdot b)$.

- a y b son variables, por lo que a su vez son expresiones.
- \cdot es un operador binario, por lo que, junto con el inciso anterior, $a \cdot b$ es una expresión.
- Como $a \cdot b$ es una expresión, también lo es $(a \cdot b)$.
- Como 4 es una constante y a una variable, $4 \cdot a$ es una expresión.

Hasta acá vamos bien. Pero ninguna de nuestras reglas indica que \cdot sea un operador binario y no existe la posibilidad de que haya una sucesión de variables sin nada entre ellas. Por lo que ésta no es una expresión aritmética bien construida.

Ejemplo 1.7. La expresión $a \cdot - + b$ es una expresión aritmética que utiliza el uso de las siguientes reglas para su construcción:

- a es una expresión
- b es una expresión
- $+b$ es una expresión pues b es una expresión y $+$ es un operador unario.
- $- + b$ es una expresión pues $+b$ es una expresión y $-$ es un operador unario.
- $a \cdot - + b$ es una expresión pues tanto a como $- + b$ son expresiones y \cdot es un operador binario.



Ejemplo 1.8. La sucesión de símbolos $\cdot - + a b$ no es una expresión aritmética correcta, pues no se puede obtener a partir de las reglas anteriores. ¿Por qué?

De los ejemplos anteriores se observa que para mostrar que una sucesión dada de símbolos s es una expresión aritmética, debemos identificar cada uno de sus componentes y asociarlos con alguna de las reglas de formación de expresiones. Este método puede resultar demasiado tedioso de aplicar, por lo que debemos buscar métodos más sencillos y susceptibles de ser automatizados.

1.3. Gramáticas y árboles de derivación

Otra manera de mostrar cómo se construyen las expresiones es mediante lo que se conoce como *gramática formal*, que es un mecanismo sencillo de especificación de reglas de construcción, llamadas *producciones o reglas de reescritura*, con las cuales se pueden generar expresiones de un lenguaje. Las formas que pueden tomar estas reglas de reescritura son muy variadas, pero nos ocuparemos sólo de una de éstas. Las reglas de reescritura de las que nos ocuparemos tienen la siguiente forma:

$$\boxed{\text{símbolo}} ::= \boxed{\text{cadena}}$$

El símbolo “ $::=$ ” se lee “se puede reescribir como” y al aplicar una regla particular sustituimos el lado izquierdo de este símbolo por la cadena que se encuentra del lado derecho.

Para las expresiones aritméticas, por ejemplo, tendríamos las reglas de reescritura que aparecen en la tabla 1.1. En ellas, los símbolos que aparecen en gris o azul (con este tipo de letra) son aquellos que ya no pueden ser reescritos, pues no aparecen del lado izquierdo de ninguna regla de reescritura. A estos símbolos les llamamos *símbolos terminales*, pues son los que *terminan* las cadenas de reescritura. A los símbolos que pueden ser reescritos les llamamos *no terminales* o *variables*. El símbolo “|” juega el papel de separador de opciones, para ahorrar trabajo.

Tabla 1.1. Reglas de reescritura para expresiones aritméticas

$$S ::= E \quad (1.1)$$

$$E ::= var \quad (1.2)$$

$$E ::= const \quad (1.3)$$

$$E ::= \triangleright E \quad (1.4)$$

$$E ::= E \diamond E \quad (1.5)$$

$$E ::= (E) \quad (1.6)$$

$$var ::= a \mid b \mid \dots \quad (1.7)$$

$$const ::= 0 \mid 1 \mid 2 \mid 17 \mid 3.5 \mid \dots \quad (1.8)$$

$$\triangleright ::= + \mid - \quad (1.9)$$

$$\diamond ::= + \mid - \mid * \mid \div \quad (1.10)$$

A una colección de reglas de reescritura como la anterior le llamamos *gramática* porque nos describe la *forma* o *reglas sintácticas* que deben tener las expresiones aritméticas bien construidas. Para producir “oraciones” (cadenas, palabras, expresiones) correctas de acuerdo a las reglas de la gramática, empezamos con el símbolo a la izquierda del $::=$ de la primera regla, y utilizamos las reglas de reescritura, que nos dicen que en cualquier momento podemos sustituir parte de (o toda) la expresión, si en ella localizamos una subexpresión³ que corresponda al lado izquierdo de cualquiera de las reglas y la sustituimos por el lado derecho. Cada vez que hacemos una sustitución, encontramos en la frase el primer símbolo desde el extremo izquierdo que aparece a la izquierda de $::=$ de alguna de las reglas y lo sustituimos por la cadena a la derecha de $::=$ en esa regla. En cada paso únicamente sustituimos a un símbolo. Decimos entonces que estamos haciendo una sustitución *por la izquierda*. Es importante mencionar que el orden en que se hagan las sustituciones no es obligatoriamente por la izquierda y no afecta el resultado final, aunque es bueno convenir en hacerlo por la izquierda en aras de obtener un método único de construcción, es decir un método *determinista*. Veamos algunos ejemplos en la figura 1.1, que se encuentra en la siguiente página.

Una secuencia de aplicación de reglas de reescritura, como las que se muestran en la figura 1.1, se conoce como una *derivación* de una expresión; esta expresión es la que figura

³Una *subexpresión* es una expresión que aparece dentro de otra. Esto implica que debe estar bien construida.

Figura 1.1. Proceso de generación de expresiones aritméticas
$$-(a * (b + c))$$

Frase	Regla usada	
S	inicio	(—)
E	$S ::= E$	(1.1)
$\triangleright E$	$E ::= \triangleright E$	(1.4)
$-E$	$\triangleright ::= -$	(1.9)
$-(E)$	$E ::= (E)$	(1.6)
$-(E \diamond E)$	$E ::= E \diamond E$	(1.5)
$-(var \diamond E)$	$E ::= var$	(1.2)
$-(a \diamond E)$	$var ::= a$	(1.7)
$-(a * E)$	$\diamond ::= *$	(1.10)
$-(a * (E))$	$E ::= (E)$	(1.6)
$-(a * (E \diamond E))$	$E ::= E \diamond E$	(1.5)
$-(a * (var \diamond E))$	$E ::= var$	(1.2)
$-(a * (b \diamond E))$	$var ::= b$	(1.7)
$-(a * (b \diamond E) + E)$	$\diamond ::= +$	(1.10)
$-(a * (var + var))$	$E ::= var$	(1.2)
$-(a * (b + c))$	$var ::= c$	(1.7)

$$a$$

Frase	Regla usada	
S	inicio	(—)
E	$S ::= E$	(1.1)
var	$E ::= var$	(1.2)
a	$var ::= a$	(1.7)

Gramática:

$S ::= E$	(1.1)
$E ::= var$	(1.2)
$E ::= const$	(1.3)
$E ::= \triangleright E$	(1.4)
$E ::= E \diamond E$	(1.5)
$E ::= (E)$	(1.6)
$var ::= a b \dots$	(1.7)
$const ::= 0 1 2 17 3.5 \dots$	(1.8)
$\triangleright ::= + -$	(1.9)
$\diamond ::= + - * \div$	(1.10)

$$(3.0 + 21)$$

Frase	Regla usada	
S	inicio	(—)
E	$S ::= E$	(1.1)
(E)	$E ::= (E)$	(1.6)
$(E \diamond E)$	$E ::= E \diamond E$	(1.5)
$(const \diamond E)$	$E ::= const$	(1.3)
$(3.0 \diamond E)$	$const ::= 3.0$	(1.8)
$(3.0 + E)$	$\diamond ::= +$	(1.10)
$(3.0 + const)$	$E ::= const$	(1.3)
$(3.0 + 21)$	$const ::= 21$	(1.8)

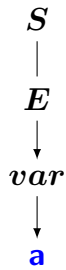
en el último de sus renglones. Estas derivaciones pueden presentarse así o gráficamente utilizando un *árbol*. Un árbol es una gráfica que remeda a un árbol biológico, excepto que elegimos pintarlo “de cabeza”. Nuestro árbol tiene un nodo inicial llamado raíz, que corresponde al origen de las sustituciones, y va creciendo de la siguiente manera:

- Cada nodo tiene un símbolo asociado.
- Para que de un nodo salgan flechas (o simplemente líneas o aristas, ya que la dirección es siempre hacia abajo) se requiere que el símbolo que está en ese nodo aparezca del lado izquierdo de alguna producción.
- Las flechas (o líneas) apuntan a cada uno de los símbolos que aparecen del lado derecho de la producción utilizada.
- Es importante observar que un nodo tiene un único símbolo asociado.
- Si el símbolo en un nodo se reescribe en una sucesión de tres símbolos, entonces deberán salir tres líneas de él, una por cada símbolo, y los símbolos deberán aparecer **exactamente** en el orden horizontal en que aparecen en la regla.

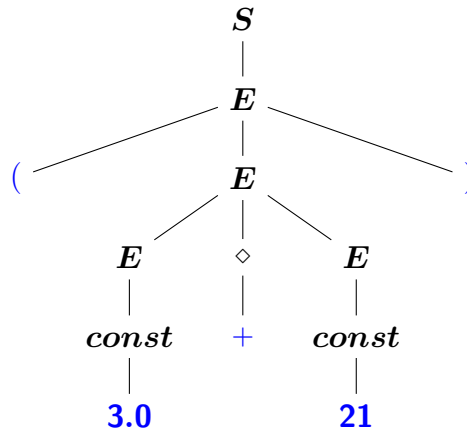
- Aquellos nodos de los que no salen líneas les llamamos *hojas*.
- Para determinar cuál es la expresión que corresponde a un determinado árbol, vamos listando las hojas del árbol de izquierda a derecha. A la cadena que se obtiene de esta manera le vamos a llamar el *resultado del árbol*.

Figura 1.2. Ejemplos de árboles de derivación

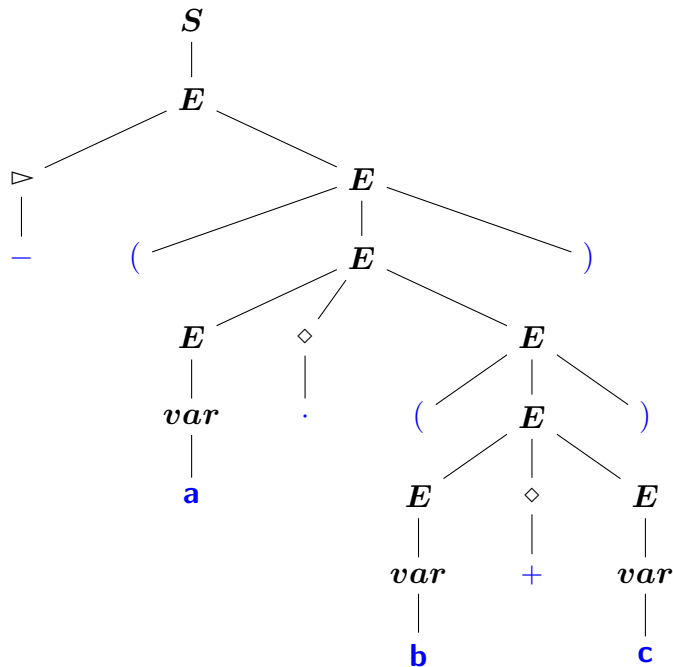
(a) **a**



(b) **(3.0 + 21)**



(c) **-(a · (b + c))**



En el árbol se pierde el orden en que se hacen las sustituciones; en cada nivel se muestran las sustituciones elegidas, de entre las posibles, en la frase que se encuentra en ese

nivel. Aquellos elementos que aparecen en gris (o en azul o con este tipo de letra) son los que no aparecen del lado izquierdo de ninguna regla y se encuentran colocados en las hojas del árbol. A estos símbolos les llamamos *símbolos terminales* y son los únicos que pueden aparecer en una expresión correcta. Los niveles intermedios no corresponden a expresiones, sino a *descripciones* de lo que puede convertirse en una expresión si se hacen los reemplazos necesarios. Podemos observar este proceso en la figura 1.2.

En este contexto, decimos que una expresión aritmética está *bien construida*, o que es *correcta*, si podemos construir el árbol que representa a su derivación. Este proceso de construcción consiste en las sustituciones que fuimos realizando en cada nivel hasta llegar a un árbol donde todas sus hojas son símbolos terminales. En otras palabras, una expresión aritmética es válida si es el resultado de algún árbol de derivación.

Una vez que tenemos bien escrita una expresión aritmética, queremos obtener su valor. Éste se obtiene reemplazando cada una de las variables que aparecen en la expresión por algún valor permitido y realizando las operaciones correspondientes. Si la expresión es aritmética, el resultado será algún número.

Expresiones de paréntesis balanceados

Como otro ejemplo del uso de gramáticas y árboles de derivación presentamos las expresiones de paréntesis balanceados. Este lenguaje es parte esencial de cualquier lenguaje de programación. La gramática que lo define se encuentra en la tabla 1.2.

Tabla 1.2. Gramática que define a paréntesis bien balanceados

$$E ::= () \quad (1.11)$$

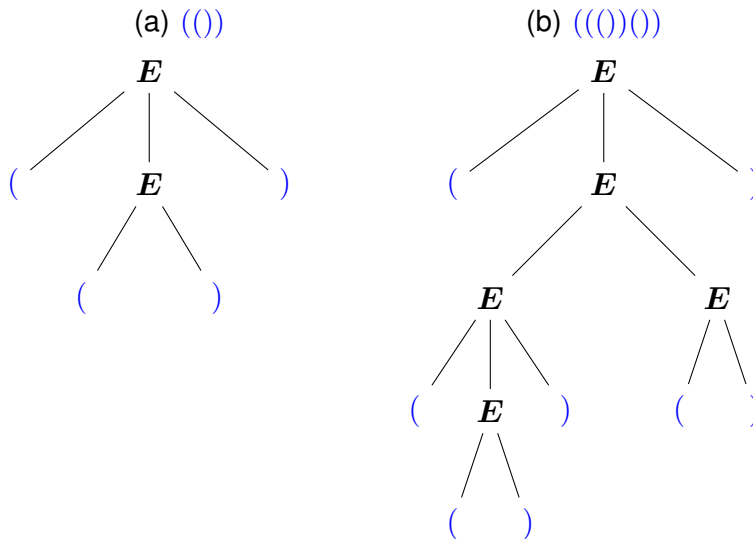
$$E ::= (E) \quad (1.12)$$

$$E ::= EE \quad (1.13)$$

Obsérvese que esta gramática produce expresiones que constan únicamente de paréntesis balanceados, no hay números ni otra clase de objetos que no sean paréntesis. La regla 1.11 corresponde a la expresión más simple de paréntesis balanceados $()$, mientras que la regla 1.12 corresponde a encerrar entre paréntesis una expresión anterior. Por otra parte, la regla 1.13 representa la generación de una nueva expresión con paréntesis balanceados al “pegar” o concatenar dos expresiones previas. Veamos un par de ejemplos.

Figura 1.3. Expresiones de paréntesis balanceados

(())			((()))		
Frase	Regla usada		Frase	Regla usada	
E	inicio	(—)	E	inicio	(—)
(E)	$E ::= (E)$	(1.12)	(E)	$E ::= (E)$	(1.12)
$(())$	$E ::= ()$	(1.11)	$((EE))$	$E ::= EE$	(1.13)
			$((()E))$	$E ::= ()$	(1.11)
			$((()))$	$E ::= ()$	(1.11)

Figura 1.4. Ejemplos de árboles de derivación para paréntesis balanceados

Ejercicios

1.3.1.- Dadas las producciones para construir expresiones aritméticas, para cada una de las siguientes expresiones decir si se pueden o no construir con esas producciones. Justifica tu respuesta.

a) $- + -a$

b) $2(b \cdot b)$

c) $\frac{1}{2}(a + b)$

1.3.2.- Usando la gramática que dimos para expresiones aritméticas, dibujar los árboles que corresponden a cada una de las expresiones que siguen:

a) $-a \cdot b + c$

b) $(-b + (b \cdot b - 4 \cdot a \cdot c)) \div (2 \cdot a)$

c) $-a + b$

1.3.3.- Dadas las expresiones aritméticas del ejercicio 1.3.2, dar la secuencia de producciones que se usan, haciendo siempre la sustitución por la izquierda.

1.3.4.- Dada las siguientes producciones:

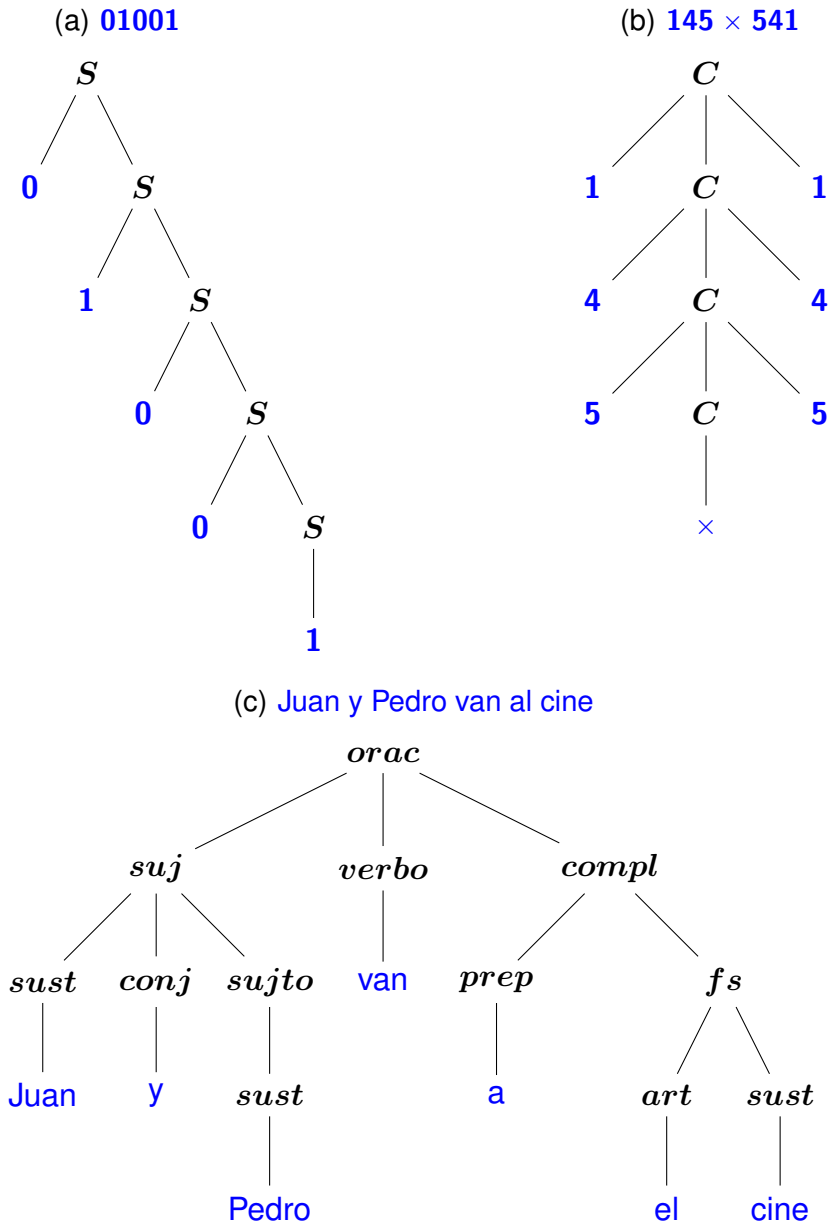
$$S ::= aSb \quad (1.14) \quad S ::= bSa \quad (1.16)$$

$$S ::= ab \quad (1.15) \quad S ::= ba \quad (1.17)$$

Dar tres expresiones que se puedan derivar de esta gramática.

1.3.5.- Para cada uno de los árboles de la figura 1.5 en la siguiente página, dar las producciones que tuvieron que utilizarse para construirlos:

Figura 1.5. Ejercicio 1.3.5



1.3.6.- Para cada uno de los árboles del ejercicio 1.3.5, dar otras dos expresiones o frases distintas a la dada que se puedan construir usando las mismas producciones.

1.3.7.- Considere la siguiente gramática G :

$S ::= [] \mid N : S \mid (S)$

$N ::= D \mid ND$

$D ::= 0 \mid 1 \mid \dots \mid 9$

Determine si las siguientes cadenas pueden derivarse mediante G . Exhiba la derivación completa y el árbol de derivación en caso de que la derivación pueda llevarse a cabo. En caso negativo explique por qué.

$$a) 7 : (13 : [])$$

$$b) 90 : (3 : []) : []$$

1.3.8.- Considere la siguiente gramática G :

$$S ::= A \mid O \mid (T$$

$$T ::=) \mid ST$$

$$O ::= + \mid - \mid * \mid /$$

$$A ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

Determine si las siguientes cadenas pueden derivarse mediante G . Exhiba la derivación completa y el árbol de derivación, en caso de que la derivación pueda llevarse a cabo; en caso negativo explique por qué no se puede.

$$a) (+ (- 2) 7)$$

$$b) (/ - (8)$$

1.3.9.- La siguiente gramática genera cadenas que representan las cartas de la baraja francesa:

$$S ::= FN \mid FL \mid Jk$$

$$F ::= \clubsuit \mid \heartsuit \mid \diamondsuit \mid \spadesuit$$

$$N ::= 2 \mid 3 \mid \dots \mid 10$$

$$L ::= A \mid J \mid Q \mid K$$

Realice lo siguiente:

a) Construya una derivación para las siguientes cartas: el rey de tréboles, el as de espadas, el 9 de diamantes y el 7 de corazones.

b) Construya los árboles de derivación para las cartas del inciso anterior.

c) Agregue una regla a la gramática, de manera que se generen manos de tres cartas, con el siguiente formato:

$$\{C1, C2, C3\}$$

donde $C1, C2, C3$ denotan cartas arbitrarias.

d) Construya una derivación y un árbol de derivación para la mano formada por las siguientes cartas: As de diamantes, 6 de tréboles, comodín

Lógica proposicional | 2

2.1. El lenguaje de la lógica proposicional

En esta sección nos dedicamos a definir la sintaxis y semántica del lenguaje formal de la lógica proposicional, empezando con una discusión acerca de argumentos lógicos.

2.1.1. Argumentos lógicos

Uno de los aspectos más importantes de la lógica matemática es el decidir si un argumento es correcto o no. Entre nuestros objetivos está el de utilizar la lógica como una herramienta para evidenciar (*deducir*) la solidez o correctud de un argumento lógico. Pero para empezar debemos contestar ¿qué es un argumento lógico? En general un argumento o argumentación se da en lenguaje natural presentando ciertos hechos –“alegatos”, verdades, situaciones– así como una *conclusión* que, si la argumentación es correcta, debe ser evidente de los hechos anteriores a los cuales llamamos *premisas*. Veamos algunos ejemplos:

Si llueve, me quedo en casa. Si me quedo en casa, leo un libro. Por lo tanto, si llueve, leo un libro.

Si me gusta el curso, pongo atención; si pongo atención, entiendo el material. Luego entonces, si me gusta el curso, entiendo el material.

x es mayor o igual que y o bien x es menor que y . x no es mayor o igual que y . De manera que x es menor que y .

Ahora bien, ¿cómo distinguimos entre las premisas y la conclusión del argumento? Esto depende de ciertas frases del lenguaje natural que nos dan la pauta para hacer la distinción, frases como *por lo tanto*, *luego entonces*, *de manera que*, entre otras.

Una vez identificadas la conclusión y las premisas se puede reescribir el argumento de una forma estructurada omitiendo ciertas frases del lenguaje natural, como en los siguientes ejemplos:

- (a)
 - 1. Si llueve, me quedo en mi casa
 - 2. Si me quedo en mi casa, leo un libro
 -
 - 3. Si llueve, leo un libro
- (b)
 - 1. Si me gusta el curso, pongo atención
 - 2. Si pongo atención, entiendo el material
 -
 - 3. Si me gusta el curso, entiendo el material
- (c)
 - 1. x es mayor o igual que y o bien x es menor que y
 - 2. x no es mayor o igual que y
 -
 - 3. x es menor que y
- (d)
 - 1. Los libros son baratos o son caros
 - 2. Los libros no son caros
 -
 - 3. Los libros son baratos
- (e)
 - 1. Este programa funciona mal o los datos son incorrectos
 - 2. Los datos son correctos
 -
 - 3. Este programa funciona mal

Obsérvese que la conclusión está separada de las premisas mediante una línea horizontal. Además, de acuerdo a nuestra intuición, todos los argumentos anteriores parecen correctos, pero formalmente ¿cuándo un argumento es correcto? o ¿cómo decidimos si un argumento es correcto? Para responder a esta pregunta nos serviremos de la lógica matemática, la cual nos proporcionará un conjunto de reglas operacionales que, en particular, permitirán obtener —deducir, encontrar, conformar, derivar— un nuevo hecho a partir de ciertos hechos dados. Un argumento lógico será *correcto o sólido* si la verdad de sus premisas causan necesaria y obligatoriamente la verdad de su conclusión, lo cual puede mostrarse mediante las reglas lógicas de operación.

Aristóteles fue el primero que para poder manipular argumentos lógicos optó por asignarles letras a ciertas frases consideradas de estructura lógica simple, llamadas proposiciones atómicas. De esta manera podemos ver en forma concisa los argumentos lógicos. Procedamos a hacer esto con los argumentos anteriores para poderlos mostrar a la manera aristotélica.

Tabla 2.2. Forma aristotélica para proposiciones atómicas

(a) p llueve q me quedo en mi casa r leo un libro 1. Si p, q 2. Si q, r <hr/> 3. Si p, r	(b) p Me gusta el curso q pongo atención r entiendo el material 1. Si p, q 2. Si q, r <hr/> 3. Si p, r
(c) p x es mayor y q x es igual a y r x es menor que y 1. $p \vee q \vee r$ 2. no $(p \vee q)$ <hr/> 3. r	(d) p Los libros son baratos q Los libros son caros 1. $p \vee q$ 2. no q <hr/> 3. p
(e) p Este programa funciona mal q Los datos son correctos 1. $p \vee \text{no } q$ 2. q <hr/> 3. p	

Se observa que el uso de letras deja ver patrones o esquemas comunes, aunque aún tenemos algunas palabras del español. Para deshacernos de ellas y formalizar completamente el estudio de argumentos lógicos, introducimos ahora el lenguaje formal de la lógica proposicional. Observen que en el inciso (c), cuando decimos “no $(p \vee q)$ ” estamos manifestando “ni p ni q ”.

2.1.2. Proposiciones

De manera similar a como construimos expresiones aritméticas, vamos ahora a definir y construir expresiones lógicas. Empecemos por ver cuáles son los objetos elementales de la lógica. En la aritmética teníamos valores numéricos (constantes) y de forma similar la lógica tiene constantes, pero sólo dos: 0 (*falso*) y 1 (*verdadero*). Estas constantes se conocen como valores lógicos o booleanos. Usar los valores de 0 y 1 como sinónimos de *falso* y *verdadero* es una libertad que nos damos las personas dedicadas a computación. También podemos hablar de los valores F y T (*false* y *true* respectivamente), aunque a lo largo de este texto usaremos 0 y 1 pues es así como se van a representar en la computadora.

Las expresiones lógicas se conocen también como *proposiciones* y son enunciados u oraciones a las que les podemos asociar un valor lógico (tienen valor de 0 o 1). En general

las proposiciones se dan en lenguaje natural; un enunciado es una proposición solamente si se puede decir, en un contexto dado, si es falso o verdadero. En el ejemplo (a) que acabamos de dar, la proposición $p = \text{llueve}$ es falsa o verdadera dependiendo del momento en que se diga, si en ese momento está lloviendo o no.

Cuando decimos que una proposición es falsa o verdadera, estamos *determinando* el valor de dicha proposición. De manera similar para las expresiones aritméticas en las que podemos hablar del *valor* de la expresión, que nos va a dar una constante numérica calculada a partir de los valores de cada una de las variables y constantes involucradas en la expresión. Al conjunto de valores de variables y constantes involucradas en la expresión es a lo que le llamamos el *estado* en el que se evalúa la expresión –a lo que anteriormente llamamos el contexto de una proposición–. Podemos definir entonces un *estado* como un conjunto de parejas, donde cada pareja tiene el nombre de una variable y el valor de esa variable. Un ejemplo de cómo especificamos un estado se encuentra a continuación.

$$\text{estado} = \{(x, 5), (y, 7), (p, \text{falso})\}$$

En este estado tenemos los valores para tres variables, dos numéricas y la tercera lógica. Cada elemento del conjunto es una pareja ordenada, donde primero se da el nombre de la variable y después el valor de esa variable en el estado.

Regresando a las expresiones lógicas, son proposiciones:

- ☞ Está lloviendo
- ☞ Juan es más grande que Pedro
- ☞ $x \geq z$
- ☞ El libro es rojo
- ☞ Roberto es el asesino
- ☞ Esta materia es fácil

No son proposiciones:

- ☞ ¡Mario, llévate esto!
- ☞ ¿Estás seguro?
- ☞ $x + y$
- ☞ Ni modo
- ☞ ¡Viva Pancho Villa!

Intuitivamente, las proposiciones se pueden evaluar, es decir, decidir si son falsas o verdaderas. Pero como mencionamos antes, este valor depende del *estado* que tomen sus variables. Por ejemplo, la proposición (c) de la tabla 2.2 es verdadera si el estado es $\{(x, 5.6), (z, 3.0)\}$. En este estado particular, la proposición tiene el valor de verdadero. Evaluada en el estado $\{(x, 2.3), (y, 4.0)\}$ la proposición tiene el valor de falso. La cuarta proposición tendrá el valor de verdadero en el caso de que el libro de que estemos hablando sea rojo, es decir cuando estemos en el estado $\{(\text{color del libro}, \text{rojo})\}$.

Más adelante hablaremos formalmente de estados y del proceso de evaluación. Por ahora sigamos con el estudio de las proposiciones.

Definición 2.1 Una **proposición** es un enunciado que puede calificarse como falso (0) o verdadero (1), dependiendo del estado en que se evalúe.

Decimos que una proposición es *atómica* si no puede subdividirse en proposiciones más simples. Las proposiciones anteriores son todas atómicas. En contraste, las siguientes proposiciones no son atómicas:

- ☞ Juan y Pedro están hambrientos
- ☞ Está nublado, por lo que va a llover, entonces no saldremos
- ☞ $0 \leq x \leq 10$
- ☞ El libro es rojo o azul

Estas proposiciones se llaman *compuestas* pues cada una de ellas se puede descomponer en dos o más proposiciones atómicas como a continuación se muestra:

- Juan y Pedro están hambrientos
 - ☞ Juan está hambriento **y** ☞ Pedro está hambriento
- Está nublado, por lo que va a llover; entonces no saldremos
 - ☞ Está nublado, **por lo que** ☞ va a llover
 - entonces**
 - ☞ **no** saldremos
- $0 \leq x \leq 10$
 - ☞ $0 \leq x$ **y** ☞ $x \leq 10$
- El libro es rojo o azul
 - ☞ el libro es rojo **o** ☞ el libro es azul

Las proposiciones atómicas son aquellas que están a continuación de ☞ y hasta el final del renglón. Encerramos en un marco a la palabra o frase que relaciona a la primera proposición atómica con la siguiente y así sucesivamente. A estas palabras les llamamos *conectivos*.

A continuación vamos a pasar de las proposiciones en lenguaje natural al estudio de un lenguaje formal de expresiones lógicas. En el proceso de traducción o especificación de lenguaje natural al formal se acostumbra asociar identificadores (letras) a las proposiciones atómicas, para poder escribir de manera más fluida y así representar y manipular adecuadamente a las proposiciones. Obsérvese que esto ya lo hicimos en la semi formalización de argumentos al inicio de este capítulo. A estos identificadores se les conoce como *variables proposicionales*. Ya tenemos entonces variables y constantes (0 y 1), pero para construir expresiones más complejas necesitamos de operadores lógicos que corresponden, estos últimos, a las frases en lenguaje natural que hemos llamado conectivos.

2.1.3. Sintaxis de la lógica proposicional

En esta sección definimos un lenguaje formal para la lógica proposicional mediante una gramática para expresiones lógicas.

Las reglas para construir proposiciones son las siguientes:

- $P ::= VarProp$ (2.1)
- $P ::= ConstLog$ (2.2)
- $P ::= \neg P$ (2.3)
- $P ::= P \diamond P$ (2.4)
- $P ::= (P)$ (2.5)
- $VarProp ::= a, b, \dots, p, q, \dots$ variables proposicionales (2.6)
- $ConstLog ::= false, true$ constantes lógicas¹ (2.7)
- $\neg ::= \neg$ negación (*not*) (2.8)
- $\diamond ::= \wedge,$ y, además, pero (*and*) (2.9)
- $\vee,$ o, o bien (*or*) (2.10)
- $\rightarrow,$ implica, (2.11)
- si ... entonces, por lo que,
de ... se sigue (*implies*)
- \leftrightarrow si y sólo si, sii, syss, iff (2.12)
- (*if and only if*)

Veamos ahora el paso del español al lenguaje formal de proposiciones mediante algunos ejemplos. Considérese la siguiente asignación de significados a variables proposicionales:

<i>Prop. atómica</i> ²	<i>Var. Prop.</i> ³	<i>Prop. atómica</i>	<i>Var. Prop.</i>
Juan está hambriento	a	$0 < x$	p
Pedro está hambriento	b	$x < 10$	q
está nublado	c	el libro es rojo	r
va a llover	d	el libro es azul	s
saldremos	e		

Las proposiciones no atómicas de los ejemplos anteriores, usando la tabla que acabamos de ver, se representan de la siguiente manera:

- Juan y Pedro están hambrientos

$a \wedge b$
- Está nublado por lo que va a llover; entonces no saldremos

$(c \rightarrow d) \rightarrow \neg e$

¹Recuerden que en computación usaremos 0 y 1.

²Proposición atómica.

³Variable proposicional

- $0 < x < 10$

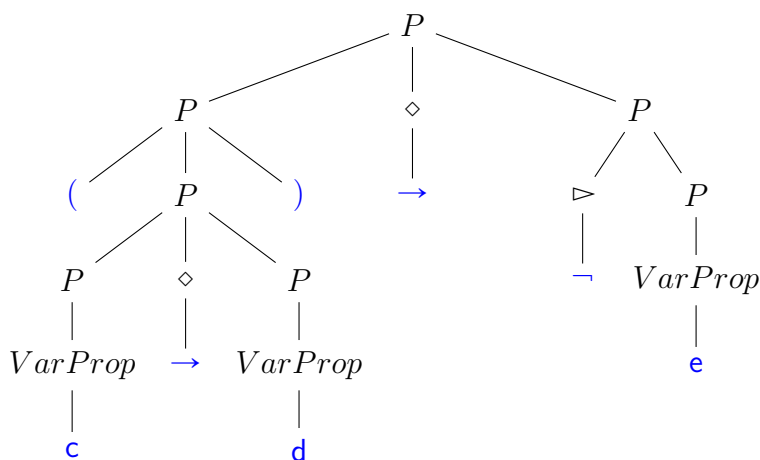
$$\boxed{p \wedge q}$$

- El libro es rojo o el libro es azul

$r \vee s$

Veamos ahora el árbol de derivación para alguna de estas expresiones, digamos $(c \rightarrow d) \rightarrow \neg e$, en la figura 2.1 que se encuentra a continuación.

Figura 2.1. Derivación de $(c \rightarrow d) \rightarrow \neg e$



Nuevamente, los símbolos terminales están en distinto tipo y color.

2.1.4. Semántica de la lógica proposicional

Una vez que hemos discutido informalmente qué es una proposición, así como la sintaxis de un lenguaje formal para proposiciones, es momento de hablar de su significado. Los aspectos relacionados con el significado de cualquier clase de expresiones forman lo que se conoce como la *semántica* del lenguaje. En nuestro caso ya conocemos el significado intuitivo de las proposiciones, de hecho le hemos dado a los operadores lógicos un nombre relativo a su significado. Por ejemplo la proposición $\neg p$ se lee “no p ” y representa a la negación de la información especificada por p . En analogía a las expresiones aritméticas, cuyo significado es un número calculado al hacer las operaciones dadas en la expresión, de acuerdo a un estado particular de sus variables, cada proposición tiene como significado un valor booleano, que depende tanto del valor particular de sus variables proposicionales como del significado de las constantes y operadores lógicos. De manera que, para poder entender el significado de una proposición, debemos empezar por definir el significado o funcionamiento de cada constante u operador lógico. El significado de las constantes lógicas debe ser claro: la constante `true` significa verdadero (1) y la constante `false` significa

falso (0). La manera más fácil para definir el significado de un operador lógico es mediante lo que se conoce como *tablas de verdad*, que consisten de una lista de todos los estados posibles en los se evalúa una proposición lógica dada. En lo que sigue se usan mayúsculas para denotar proposiciones que pueden ser compuestas. A continuación analizamos cada operador lógico.

La negación

La negación de una proposición P se denota de alguna de las siguientes formas:

$$\neg P, \sim P, \overline{P}, P'$$

- Nosotros usaremos $\neg P$ exclusivamente.

$$\boxed{\neg P}$$

- Su significado en español es:

no P
no es cierto que P
es falso que P

- Su tabla de verdad es:

	negación
P	$\neg P$
1	0
0	1

Este tipo de tablas merece algunas observaciones. Para calcular la tabla de verdad de una proposición cualquiera E es necesario considerar todos los *estados* posibles de los operandos de la expresión E . Cada operando puede estar en uno de dos estados posibles, 1 para verdadero y 0 para falso. Cada renglón de la tabla corresponde a un estado particular de los operandos. En este caso nuestra expresión es $\neg P$, que tiene como único operando a P , que independientemente de que sea una proposición atómica o no, sólo puede estar en dos estados posibles, por lo que la tabla de verdad sólo tiene dos renglones. En esta tabla, la primera columna es la que indica el estado del operando P mientras que la segunda nos indica el resultado de la evaluación de la expresión deseada, en este caso $\neg P$. Como se ve en la tabla anterior, el operador \neg lo que hace es “invertir” o negar el valor original de la proposición dada.

Veamos a continuación la semántica de los operadores lógicos binarios.

La conjunción

La conjunción de dos proposiciones P y Q se denota de alguna de las siguientes formas:

$$P \wedge Q, P \& Q, P \cdot Q, PQ.$$

- Nosotros usaremos $P \wedge Q$ exclusivamente.

$$\boxed{P \wedge Q}$$

- Su significado en español es:

$$\begin{array}{c} P \text{ y } Q \\ P \text{ además de } Q \\ P \text{ pero } Q \end{array}$$

Puede observarse aquí cierta incapacidad de la lógica para representar al español: ciertamente al usar la palabra *pero* se le está dando cierta intención a una afirmación que no corresponde a la simple conjunción, como en la frase *Te llevo al cine, pero haces la tarea*, la cual sólo puede representarse con una conjunción que corresponde a *Te llevo al cine y haces la tarea*. Desafortunadamente, en lógica la única posibilidad para representar un *pero* es la conjunción.

- Su tabla de verdad es:

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

En esta ocasión, al haber dos operandos (P y Q), tenemos cuatro posibles estados para el sistema:

- Que ambas proposiciones valgan 1 (1,1)
- Que P valga 1 y Q valga 0 (1,0)
- Que P valga 0 y Q valga 1 (0,1)
- Que ambas proposiciones valgan 0 (0,0)

La disyunción

La disyunción de dos proposiciones P y Q se denota de alguna de las siguientes formas:

$$P \vee Q, P \mid Q, P + Q.$$

- Nosotros usaremos $P \vee Q$ exclusivamente.

$$\boxed{P \vee Q}$$

- Su significado en español es:

P o bien Q

o P o Q

- Su tabla de verdad es:

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

Observando el primer renglón de la tabla de verdad nos damos cuenta de que este uso de la disyunción es inclusivo, es decir, la disyunción es cierta también en el caso en que ambos operandos sean ciertos.

La implicación

La implicación o condicional de dos proposiciones P y Q se denota de alguna de las siguientes formas:

$P \rightarrow Q$, $P \Rightarrow Q$, $P \supset Q$

- Nosotros usaremos $P \rightarrow Q$ exclusivamente.

$P \rightarrow Q$

- Su significado en español es:

si P entonces Q

P implica Q

P es (condición) suficiente para Q

Q , si P

Q , siempre que P

P sólo si Q

Q se sigue de P

Q es (condición) necesaria para P

- Su tabla de verdad es la que sigue:

P	Q	Implicación o condicional $P \rightarrow Q$
1	1	1
1	0	0
0	1	1
0	0	1

Nos sorprende en esta tabla la evaluación del tercer y cuarto renglones, pues parece, a primera vista, contrario a la intuición. Veamos un ejemplo:

Ejemplo 2.1. Démosle significado a la implicación en lenguaje natural.

p	es	una botella contiene ácido
q	es	la botella tiene una calavera en la etiqueta
$p \rightarrow q$	es	si una botella tiene ácido, entonces tiene una calavera en la etiqueta

Como se ve en este ejemplo, la verdad de p (que la botella contenga ácido) nos permite garantizar la verdad de q (que hay una calavera en la etiqueta). Pero si la botella no contiene ácido, pudiera ser que la botella contenga algún otro compuesto venenoso y que de todos modos tenga una calavera en la etiqueta, estado que tenemos en el tercer renglón de la tabla; pero también pudiera ser que la botella no tenga ácido y que no tenga calavera en la etiqueta, estado que se presenta en el último renglón de la tabla. Lo que no puede suceder (el resultado es 0) es que la botella, conteniendo ácido **no** tenga una calavera en la etiqueta, estado representado por el segundo renglón de la tabla.

Veamos otro ejemplo, esta vez en matemáticas.

Ejemplo 2.2. Considérese la siguiente proposición⁴:

$$((x > y) \wedge (y > z)) \rightarrow (x > z)$$

Evaluemos esta expresión en el estado $\{(x, 8), (y, 6), (z, 4)\}$. En este estado, el antecedente de la implicación es verdadero $((8 > 6) \text{ y } (6 > 4))$, por lo que podemos garantizar que $x > z$, pues en efecto, $8 > 4$. Sin embargo, veamos que sucede en el estado $\{(x, 7), (y, 8), (z, 6)\}$. En este caso el antecedente es falso pero el consecuente es verdadero. El valor de la proposición es, de acuerdo a la definición en su tabla de verdad, verdadero. Otro estado que ilustra el primer caso es $\{(x, 4), (y, 6), (z, 5)\}$. También este estado hace que la proposición se evalúe a verdadero, porque una vez que el antecedente es falso, el estado del consecuente puede ser cualquiera. Por otra parte, si el antecedente es verdadero, no puede suceder que el consecuente sea falso, es decir, no existe un estado en el cual $(x > y)$ y $(y > z)$ y que sin embargo tengamos $(x \leq z)$.

Los valores de verdadero y falso de la implicación simplemente nos dicen cuáles estados pueden presentarse y cuáles no. En el primer ejemplo que dimos, si llueve es seguro que me quedo en casa, pero si no llueve, el estado del consecuente puede ser cualquiera. Recordemos que sólo hay dos estados posibles para las proposiciones lógicas, falso o verdadero.

Cada implicación $P \rightarrow Q$ tiene asociadas otras implicaciones que involucran a las mismas proposiciones P y Q que a continuación definimos:

⁴Usamos aquí tantos paréntesis como se requieran para definir sin ambigüedades la estructura de la expresión lógica.

- La *recíproca* o *inversa* de $P \rightarrow Q$ es la fórmula $Q \rightarrow P$.
- La *contrapositiva* de $P \rightarrow Q$ es la fórmula $\neg Q \rightarrow \neg P$.
- La *contrarrecíproca* de $P \rightarrow Q$ es la fórmula $\neg P \rightarrow \neg Q$.

Ejemplo 2.3. Considérese la oración *si tengo un triángulo entonces tengo un polígono*, formalizada como $t \rightarrow p$. Sus implicaciones asociadas son:

- Recíproca: $p \rightarrow t$ que significa *si tengo un polígono entonces tengo un triángulo*.
- Contrapositiva: $\neg p \rightarrow \neg t$ que significa *si no tengo un polígono entonces no tengo un triángulo*.
- Contrarrecíproca: $\neg t \rightarrow \neg p$ que significa *si no tengo un triángulo entonces no tengo un polígono*

Más adelante veremos la relación existente entre una implicación y sus implicaciones asociadas.

La equivalencia

La equivalencia o bicondicional de dos proposiciones P y Q se denota de alguna de las siguientes formas:

$$P \leftrightarrow Q, P \Leftrightarrow Q, P \equiv Q.$$

- Nosotros usaremos $P \leftrightarrow Q$ exclusivamente.

$$\boxed{P \leftrightarrow Q}$$

- Su significado en español es:

P si y sólo si Q

P es equivalente a Q

P es (condición) necesaria y suficiente para Q

- Su tabla de verdad es:

P	Q	Equivalencia o bicondicional $P \leftrightarrow Q$
1	1	1
1	0	0
0	1	0
0	0	1

En este caso, la equivalencia es verdadera si ambas proposiciones se evalúan a lo mismo: ambas se evalúan a falso o ambas se evalúan a verdadero.

Tablas de verdad para proposiciones compuestas

Al conocerse el significado de cada conectivo lógico mediante su tabla de verdad es posible obtener el significado de cualquier fórmula mediante una tabla de verdad que combine

las tablas de cada subfórmula componente de la fórmula original. Veamos un ejemplo.

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

Como se observa, las tablas de verdad crecen tanto en columnas como en renglones, al volverse más compleja la fórmula en cuestión. ¿Cuántos renglones tiene la tabla de verdad de una fórmula que tiene n variables proposicionales?

Propiedades de los conectivos lógicos

Vimos en las secciones anteriores lo que constituye una proposición, así como el significado de los principales operadores o conectivos lógicos. De conocer las tablas de verdad para estos conectivos, podemos observar algunas de sus propiedades importantes.

Conmutatividad: Esta propiedad nos dice que el orden en que aparecen las proposiciones relacionadas por el conectivo lógico no afecta el resultado de la operación. Por ejemplo, la evaluación de $p \wedge q$ da siempre el mismo resultado que la evaluación de $q \wedge p$. Esta propiedad, exclusiva de los operadores binarios, la tienen asimismo los operadores aritméticos de suma y multiplicación. De las expresiones aritméticas sabemos, por ejemplo, que ni la resta ni la división son operadores conmutativos: No es lo mismo $7 - 5$ que $5 - 7$; tampoco se evalúa a lo mismo $8 \div 2$ que $2 \div 8$. También en el caso de los conectivos lógicos no todos son conmutativos. Los conectivos $\vee, \wedge, \leftrightarrow$ son conmutativos pues:

El valor de:	es el mismo que el de:
$p \vee q$	$q \vee p$
$p \wedge q$	$q \wedge p$
$p \leftrightarrow q$	$q \leftrightarrow p$

De su tabla de verdad es fácil ver que la implicación (\rightarrow) no es conmutativa.

Asociatividad: En aritmética es claro que $(a + b) + c = a + (b + c)$. Decimos entonces que la suma es *asociativa*. En el caso de los conectivos lógicos no todos tienen

esta propiedad, exclusiva de los operadores binarios, llamada *asociatividad*. Mientras que en la aritmética la suma y la multiplicación son asociativos, esto no es así con otros operadores binarios como la resta y la división. Por ejemplo, en el estado $\{(a, 5), (b, 7), (c, 3)\}$, $a - (b - c) = 1$, mientras que $(a - b) - c = -5$. También, $(a \div b) \div c = 5/21 \approx 0.24$, mientras que $a \div (b \div c) = 15/7 \approx 2.1$. En el caso de la lógica matemática los conectivos binarios que son asociativos son la conjunción (\wedge), la disyunción (\vee) y la equivalencia (\leftrightarrow). Nuevamente, la condicional (\rightarrow) tampoco presenta esta propiedad.

El valor de:	es el mismo que el de:
$(p \wedge q) \wedge r$	$p \wedge (q \wedge r)$
$(p \vee q) \vee r$	$p \vee (q \vee r)$
$(p \leftrightarrow q) \leftrightarrow r$	$p \leftrightarrow (q \leftrightarrow r)$

Elemento identidad: En general, un *elemento identidad* para un operador binario \star es aquel valor que al operarlo con una expresión el resultado es esa misma expresión, es decir e es una identidad para \star si $e \star x = x = x \star e$ para cualquier expresión x . (Noten que estamos requiriendo la conmutatividad del operador con respecto al elemento identidad y cualquier otro elemento.) En el caso de la suma, el elemento identidad es el 0 puesto que $a + 0 = a = 0 + a$, mientras que en el caso de la multiplicación el elemento identidad es el 1 ya que $a \cdot 1 = a = 1 \cdot a$. Como se ve, el elemento identidad va a depender del operador o conectivo particular. En el caso de los conectivos lógicos, los elementos identidad de cada operador se dan a continuación. Para ver que, en efecto, son elementos identidad, sugerimos desarrollar las tablas de verdad correspondientes.

Operador	Identidad	El valor de	es el valor de
\wedge	true	$p \wedge \text{true}$	p
\vee	false	$p \vee \text{false}$	p
\leftrightarrow	true	$p \leftrightarrow \text{true}$	p

Elemento neutro: También conocido como *dominante*, es aquella constante que al operar con cualquier otro valor, el resultado es la constante misma. Es decir, e es un elemento neutro para el operador binario \star si $x \star e = e = e \star x$ para cualquier expresión x . En el caso de la aritmética, el 0 (cero) con el producto tiene ese efecto. Hay que notar que la suma, la resta y la división no tienen elemento neutro (el elemento nulo tiene que ser el mismo para todos los valores que puedan participar en la operación). En el caso de las operaciones lógicas binarias, el elemento neutro de la disyunción (\vee) es la constante true y de la conjunción (\wedge) es la constante false.

El valor de:	es el valor de:
$p \vee \text{true}$	true
$p \wedge \text{false}$	false

Idempotencia: Esta propiedad habla de un operador binario que al tener dos operandos iguales el resultado es el operando mismo. Por ejemplo, si tenemos la proposición $p \wedge p$ podemos observar de la tabla de verdad, que su valor es el mismo que el de p . Los operadores \wedge y \vee son idempotentes:

p	$p \wedge p$	$p \vee p$
1	1	1
0	0	0

Para la implicación hay otras proposiciones interesantes que vale la pena notar. Se caracterizan porque al operar con la constante false o true dan siempre como resultado el valor de 1:

p	$\text{false} \rightarrow p$	$p \rightarrow \text{true}$
1	1	1
0	1	1

2.1.5. Tautologías y contradicciones

Las tablas de verdad nos permiten observar el valor de una fórmula en *todos* sus posibles estados. Esto nos permite clasificar a las fórmulas de la siguiente manera:

tautologías Aquellas fórmulas que se evalúan a *verdadero* en todos los estados posibles

contradicciones Aquellas fórmulas que se evalúan a *falso* en todos los posibles estados

fórmulas contingentes o contingencias Aquellas fórmulas que no son ni tautologías ni contradicciones

Conocemos ya varias tautologías, como es el caso de $p \vee \neg p$, $p \rightarrow p \vee q$, $p \wedge p \leftrightarrow p$. Para convencernos, veamos a continuación sus tablas de verdad:

p	q	$p \vee \neg p$		$p \rightarrow p \vee q$		$p \wedge p \leftrightarrow p$	
		\vee	\neg	\rightarrow	\vee	\wedge	\leftrightarrow
1	1	1	0	1	1	1	1
1	0	1	0	1	1	1	1
0	1	1	1	1	1	0	1
0	0	1	1	1	0	0	1

Como las tautologías son muy importantes, se elige una notación especial para representarlas. Para ello utilizamos un *metalenguaje*, el cual nos sirve para decir algo respecto al lenguaje formal que estamos utilizando. Ya nos encontramos con metalenguajes con anterioridad. Por ejemplo, nuestras gramáticas con sus producciones corresponden a un metalenguaje, ya que si bien nos describen perfectamente lo que es una expresión, las producciones en sí *no* son expresiones. Podemos pensar también en los esquemas de fórmula que utilizamos (E , $P \vee Q$, $A \rightarrow B$, entre otros) como *meta expresiones*, ya que los usamos para describir a objetos de nuestro lenguaje particular, pero ellos no forman parte del lenguaje. Más adelante hablaremos de esquemas con más detalle.

Volviendo al cálculo proposicional, si A es una proposición que es tautología, escribimos $\models A$. Insistimos en que el símbolo \models **no** es un operador de la lógica proposicional y la expresión $\models P$ **no** es una proposición, sino que nos habla *acerca* de la proposición P , diciéndonos que P es una tautología.

Como ejemplos de tautologías de gran importancia tenemos:

- $p \vee \neg p$ *Ley del tercero excluido*, nos dice que toda proposición tiene que evaluarse a falso o verdadero, que no hay ningún otro valor posible.
- $\text{false} \rightarrow p$ *Falso implica cualquier cosa*. Cuando el antecedente es falso, se puede concluir cualquier proposición.
- $p \rightarrow \text{true}$ Cuando el consecuente es verdadero, cualquier proposición lo implica (lo “justifica”).

Contradicciones

Una *contradicción* es una expresión que se evalúa a falso en todos los estados posibles. Podemos cotejar que una expresión es una contradicción utilizando para ello tablas de verdad, como en el caso de las tautologías. Por ejemplo, $P \leftrightarrow \neg P$ y $P \wedge \neg P$ son ambas contradicciones, como se muestra en las tablas de verdad correspondientes en la siguiente página.

P	$\neg P$	$P \leftrightarrow \neg P$	$P \wedge \neg P$
1	0	0	0
0	1	0	0

Las contradicciones están íntimamente relacionadas con las tautologías. Si A es una tautología, entonces $\neg A$ es una contradicción y viceversa.

2.1.6. Argumentos correctos

Una vez que hemos definido la sintaxis y la semántica de las fórmulas de la lógica proposicional, así como el concepto de tautología, podemos dar la definición formal de argumento lógico e introducir formalmente la noción de argumento correcto.

Definición 2.2 Un argumento lógico es una sucesión de fórmulas A_1, \dots, A_n llamadas *premisas* y una fórmula B llamada *conclusión*. Dicha sucesión se escribe usualmente como

$$\begin{array}{c} A_1 \\ \vdots \\ A_n \\ \hline \therefore B \end{array} \quad \text{o bien} \quad A_1, \dots, A_n / \therefore B$$

Nuestro problema fundamental es decidir cuándo un argumento es correcto o válido, lo cual sucederá, como ya mencionamos anteriormente, si y sólo si **suponiendo** que sus premisas son verdaderas, entonces necesariamente la conclusión también lo es. Obsérvese que esta definición corresponde a los llamados *argumentos deductivos*. En contraste, en un *argumento inductivo* se aceptan como válidas conclusiones basadas en observación o probabilidad. Nosotros nos dedicaremos sólo a los argumentos deductivos.

Como ya tenemos a nuestra disposición la definición de tautología, nos servimos de ésta para dar una definición formal de argumento correcto.

Definición 2.3 (argumento correcto) El argumento

$$A_1, A_2, \dots, A_n / \therefore B$$

es correcto si y sólo si

$$\models A_1 \wedge A_2 \dots A_n \rightarrow B.$$

A la fórmula $A_1 \wedge A_2 \dots A_n \rightarrow B$ se le llama *fórmula asociada al argumento lógico*.

Por lo tanto, verificar la correctud de un argumento es equivalente a verificar que su fórmula asociada es tautología, para lo cual basta construir su tabla de verdad. Veamos algunos ejemplos.

Ejemplo 2.4. El argumento $p \rightarrow q, p / \therefore q$, es correcto. La fórmula a analizar es $p \wedge (p \rightarrow q) \rightarrow q$.

p	q	p	\wedge	$(p \rightarrow q)$	\rightarrow	q
1	1	1	1	1	1	1
1	0	1	0	0	1	0
0	1	0	0	1	1	1
0	0	0	0	1	1	0

Como muestra la tabla, tenemos una tautología y el argumento es correcto.

Ejemplo 2.5. Analizar el siguiente argumento.

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

Frases como “por lo tanto”, “así que”, “luego entonces”, “de forma que”, entre otras, señalan la conclusión del argumento.

La representación formal del argumento es:

$$\frac{v \rightarrow s \quad s}{\therefore v}$$

De manera que el argumento es correcto si y sólo si $\models (v \rightarrow s) \wedge s \rightarrow v$. La tabla de verdad es:

v	s	$(v \rightarrow s)$	\wedge	s	\rightarrow	v
1	1	1	1	1	1	1
1	0	0	0	0	1	1
0	1	1	1	1	0	0
0	0	1	0	0	1	0

El tercer renglón de la tabla muestra que la fórmula no es una tautología por lo que el argumento es incorrecto.

Ejemplo 2.6. Mostrar la correctud del siguiente argumento:

$$\frac{p \wedge q \rightarrow r \quad p}{\therefore q \rightarrow r}$$

La tabla de verdad de la fórmula asociada al argumento es:

p	q	r	$(p \wedge q \rightarrow r)$	\wedge	p	\rightarrow	$(q \rightarrow r)$
1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	0
1	0	1	1	1	1	1	1
1	0	0	1	1	1	1	1
0	1	1	1	0	0	1	1
0	1	0	1	0	0	1	0
0	0	1	1	0	0	1	1
0	0	0	1	0	0	1	1

Por lo que $\models ((p \wedge q) \rightarrow r) \wedge p \rightarrow (q \rightarrow r)$ y el argumento es correcto.

El ejemplo anterior deja ver que el método de tablas de verdad para mostrar la correctud de un argumento puede resultar complicado, al crecer el número de variables involucradas en el mismo. Por esta razón, resulta obligatorio buscar métodos alternativos, cosa que haremos más adelante.

Ejercicios

2.1.1.- ¿Cuáles de las siguientes oraciones son proposiciones atómicas, cuáles proposiciones no atómicas y cuáles no son proposiciones? Justifica tu respuesta.

- a) El cielo está nublado
- b) Por favor ven a verme
- c) $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
- d) $0 \leq x \leq 10$
- e) Juan y Pedro van al cine
- f) Estoy a dieta porque es necesario para bajar de peso

2.1.2.- Exprese los siguientes enunciados en el lenguaje de la lógica proposicional:

- a) Un triángulo equilátero tiene sus tres ángulos iguales.
- b) Siempre que come fresas le da alergia.
- c) $0 \leq x \leq y \leq 15$.
- d) Todo número par es divisible entre 2.
- e) Para que vayas al cine tienes que hacer tu tarea.

2.1.3.- Use variables proposicionales p , q y r para formalizar los siguientes argumentos lógicos. Liste cómo asignas las variables a las proposiciones atómicas.

- a) Si hay exámenes cada semana, los estudiantes se quejan; y si no hay exámenes cada semana, los estudiantes se quejan; de cualquier forma los estudiantes se quejan.
- b) Si n es número primo, no puede ser divisible entre 2; sabemos que 24 es divisible entre 2, por lo que no es número primo.
- c) Si lo mató, fue un crimen pasional; y si es un crimen pasional, el asesino sale corriendo; sabemos que ella no salió corriendo; entonces no lo mató.
- d) No hay otra manera de pasar la materia más que estudiando.
- e) Hay que llegar temprano para agarrar buen lugar.

2.1.4.- Usando las variables proposicionales ℓ y s para denotar a las proposiciones atómicas *Juan es muy listo* y *Juan está satisfecho* respectivamente, denote con estas variables proposicionales y los conectivos lógicos a las siguientes proposiciones:

- a) Juan es muy listo y está satisfecho.
- b) Si Juan no fuera listo, no estaría satisfecho.
- c) Juan es listo o está satisfecho.
- d) Juan está satisfecho únicamente si es listo.
- e) Si Juan es listo entonces está satisfecho.
- f) Juan es listo pero no está satisfecho.

2.1.5.- En los siguientes enunciados, identifique las proposiciones atómicas y asígneles variables proposicionales. Una vez hecho esto, convierta los enunciados a proposiciones lógicas.

- a) Si Juan fue al cine, seguro que Lupe fue también.
- b) Las noticias no son buenas.
- c) Te darán clave para la red sólo si estás inscrito en el curso.
- d) Si asistió a las clases, debió pasar la materia.
- e) El asesino era de tez blanca o clara.

2.1.6.- Formalice las siguientes implicaciones y construya las fórmulas de sus implicaciones asociadas (se definieron en la página 27).

- a) Si un número es divisible entre 2 entonces es par.
- b) Si Elke es austriaca entonces es europea.
- c) Una condición necesaria para que Lourdes lleve el curso de algoritmos es que apruebe Estructuras Discretas.
- d) El programa es legible sólo si está bien estructurado.
- e) La audiencia dormirá si el ponente diserta sobre lógica medieval.

2.1.7.- Para el siguiente enunciado, asigne variables proposicionales a las proposiciones atómicas y escriba la proposición completa usando esas variables proposicionales.

- (a) María fue al teatro el lunes en la noche sólo en el caso de que no tuviera clase el martes temprano.
- (b) Si Juan llevó su Mustang al desfile es porque le cambió el amortiguador el día anterior.

- (c) Si los tres lados de un triángulo son congruentes, entonces los tres ángulos del triángulo son congruentes.
- (d) Si x es mayor que 3 entonces también es mayor que 2.
- (e) Nunca ha nevado en Cuernavaca.
- (f) Si n es un entero, entonces $n^3 - n$ es par.

2.1.8.- Para cada pareja de enunciados que se listan, escriba las fórmulas para la disyunción de ambos y la conjunción de ambos. Para cada fórmula, indique si es verdadera o no.

- a) p : Uno es un entero par q : Nueve es un entero positivo
- b) p : Chihuahua está en la frontera con EEUU q : Brasil está en África
- c) p : La naranja es una fruta q : La papa es una verdura
- d) p : Los pájaros tienen cuatro patas q : Los conejos vuelan
- e) p : Los cardenales son rojos q : Los ruiséñores son azules

2.1.9.- Para cada uno de los siguientes enunciados, asigne variables proposicionales y escriba la fórmula o argumento lógico correspondiente al enunciado.

- a) Si hoy es viernes, iré al cine.
- b) Si termino la tarea voy a tomar un descanso.
- c) Si Pepito compite en natación va a ganar el primer lugar. Si Juanito compite en natación va a ganar el primer lugar. Alguno de los dos no va a quedar en primer lugar en la competencia de natación. Por lo tanto o Pepito no compite o Juanito no compite.
- d) Los perros son mamíferos. Los mamíferos no tienen agallas. Por lo tanto los perros no tienen agallas.
- e) Voy a comer tacos o quesadillas. Decidí no comer quesadillas. Entonces comeré tacos.

2.1.10.- Elabore la tabla de verdad para el operador $\boxed{\text{nand}}$, donde $p \boxed{\text{nand}} q$ está definido como $\neg (p \wedge q)$.

2.1.11.- Elabore las tablas de verdad para $p \wedge p$, $p \vee \neg p$, $p \vee p$, $p \wedge \text{true}$, $p \wedge \text{false}$, $p \vee \text{true}$ y $p \vee \text{false}$. Observe cada una de estas tablas de verdad y diga la relación que tienen con la variable p original.

2.1.12.- Construya la tabla de verdad para cada una de las siguientes fórmulas, clasificando si se trata de una tautología, contradicción o contingencia.

- a) $(p \wedge q) \rightarrow \neg(r \wedge q)$
- b) $(p \wedge (r \wedge q)) \rightarrow r$
- c) $((p \rightarrow q) \wedge \neg q) \rightarrow p$
- d) $(s \vee t) \leftrightarrow (s \wedge t)$
- e) $(r \rightarrow s) \wedge \neg t$
- f) $(q \vee p) \rightarrow (\neg p \rightarrow q)$

2.1.13.- Analizar mediante tablas de verdad la correctud de los siguientes argumentos.

$$a) p, q / \therefore p \wedge q$$

$$b) q, r, s / \therefore q \wedge t$$

$$c) p \rightarrow q, \neg q / \therefore \neg p$$

$$a) p \rightarrow q \vee r, \neg q / \therefore p \rightarrow r$$

$$b) p \rightarrow q, \neg p / \therefore \neg q$$

2.2. Evaluación de expresiones

Si bien el proceso de evaluación de una expresión nos queda intuitivamente claro y en muchos casos es un proceso mental completamente automático, vamos a formalizarlo en esta sección. El objetivo de esta formalización radica principalmente en la necesidad de la misma para un estudio en abstracto de la evaluación y sus propiedades, mediante el cual se podrá automatizar el proceso de evaluación más fácilmente.

2.2.1. Estados y evaluación

Regresamos al concepto de *estado* que, como dijimos en la sección anterior, está íntimamente relacionado con la evaluación de expresiones (o, en nuestro caso, de proposiciones).

Definición 2.4 (estado) Un **estado** es una función que asigna a una variable dada x un valor v , elegido de entre aquellos que pueden asignarse a esa variable. Un estado se representa usualmente mediante un conjunto de parejas ordenadas (x, v) (o bien $x = v$), donde en cada pareja el primer elemento x es una variable y el segundo elemento v es un valor.

Definición 2.5 (evaluación) La *evaluación de una expresión E en un cierto estado* se logra reemplazando todas las variables en E por los valores que éstas tienen en ese estado, calculando después el valor de la expresión resultante, dictado por el significado de los operadores que figuran en la expresión.

Esta definición, si bien debe ser intuitivamente clara, no es completamente formal; más adelante definiremos formalmente qué significa reemplazar una variable por un valor o una expresión. Veamos algunos ejemplos en la tabla 2.3 que se encuentra en la siguiente página.

Si sucede que hay variables en la expresión que no aparecen en el estado (es decir, que no tienen un valor asignado), entonces la evaluación de la expresión incluirá presencias de esas variables a las que no les podemos asignar valor (quedará con *incógnitas*, como les hemos llamado a este tipo de variables). En estos casos, lo más común es que la expresión obtenida mediante esta evaluación parcial, interactúe más adelante con otro estado para terminar su evaluación. Sin embargo, en algunas ocasiones se puede evaluar completamente una expresión aun cuando el valor de alguna de sus variables no esté definido en el estado.

Esto sucede cuando el valor de la expresión no depende de dicha variable. Por ejemplo, si llegamos a una expresión como $0 \cdot (a + b)$ es irrelevante el valor ya sea de a o de b , porque esta expresión se evalúa a 0 (cero); lo mismo para las expresiones $0 \rightarrow p$ o bien $p \rightarrow 1$, pues sabemos que el resultado de ambas expresiones es verdadero (1). Es útil, entonces, para ahorrarnos algo de trabajo, conocer las propiedades de los operadores y de algunas expresiones en las que están involucradas constantes.

Tabla 2.3. Evaluación de algunas expresiones

Expresión	Estado	Evaluación
$m \div n$	$\{ m = 63, n = 7 \}$	9
$m \div n$	$\{ m = 8, n = 48 \}$	$\frac{1}{6}$
$i = 1$	$\{ i = 2, j = 1 \}$	0
$i = 1$	$\{ i = 1 \}$	1
$i = 1$	$\{ j = 1 \}$	$i = 1$
$a + (b \cdot c)$	$\{ a = 3, b = 5, c = 2 \}$	13
$a + (b \cdot c)$	$\{ a = 4, b = 5, c = 7, d = 8 \}$	39
$a + (b \cdot c)$	$\{ a = 13, b = 11, d = 2 \}$	$13 + (11 \cdot c)$
$(p \wedge q) \vee r$	$\{ p = 0, q = 1, r = 1 \}$	1
$(p \wedge q) \vee r$	$\{ p = 1, q = 0, r = 0 \}$	0
$(p \rightarrow q) \rightarrow r$	$\{ p = 0, q = 0, r = 0 \}$	0
$(p \rightarrow q) \rightarrow r$	$\{ p = 1, q = 0, r = 0 \}$	1

2.2.2. Precedencia y asociatividad

Hemos utilizado paréntesis en expresiones. Los paréntesis nos indican *agregación*. Por ejemplo, en la expresión $3 + (4 \cdot 5)$ los paréntesis agregan la expresión $4 \cdot 5$ como el segundo operando de la suma para indicar que la operación que queremos realizar es la suma de 3 con el producto de 4 y 5, cuyo resultado es 23. Si la expresión tuviera los paréntesis $(3 + 4) \cdot 5$, se estaría agregando la suma de 3 y 4 como operando del producto, dando como resultado 35. Para reducir el número de paréntesis en una expresión se asignan *precedencias* a los operadores. En particular, los lenguajes de programación hacen esto, pues el uso excesivo de paréntesis resulta ser una carga para el programador y oscurece el significado de la expresión para el lector humano. Si el operador op_1 tiene mayor precedencia que el operador op_2 , eso quiere decir que primero evaluamos la operación de op_1 y después la de

*op*₂. Por ejemplo, como usualmente la multiplicación tiene mayor precedencia que la suma, en la expresión $3 + 4 \cdot 7$ se debe evaluar primero el producto $4 \cdot 7$ y ese resultado usarlo para la suma con 3. En otras palabras, es como si los paréntesis aparecieran alrededor del producto, $3 + (4 \cdot 7)$ y, de hecho, una vez definido el orden de precedencia, es posible restaurar los paréntesis originales siguiendo este orden.

Otro concepto que se relaciona con el orden de evaluación de una expresión, es el de *asociatividad*. Esta propiedad nos permite decidir el orden de evaluación, en ausencia de paréntesis, cuando tenemos dos o más aplicaciones consecutivas de un mismo operador binario en una expresión. Por ejemplo, en la expresión $p \rightarrow q \rightarrow r$, ¿cuál de los dos operadores \rightarrow debe evaluarse primero, el de la izquierda o el de la derecha? El resultado de la evaluación es distinta, dependiendo de la asociatividad que hayamos convenido:

p	q	r	$(p \rightarrow q) \rightarrow r$	<i>Valor</i>	$p \rightarrow (q \rightarrow r)$	<i>Valor</i>
0	0	0	$1 \rightarrow 0$	0	$0 \rightarrow 1$	1

Por supuesto que si un operador binario es asociativo, como por ejemplo \wedge o $+$, entonces cualquiera de las dos posibilidades, izquierda o derecha, dará el mismo resultado. Sin embargo se debe elegir de forma permanente una de las dos para evitar ambigüedades.

Como se puede ver, tanto la precedencia como la asociatividad determinan, en ausencia de paréntesis, el orden de evaluación de las subexpresiones. Los paréntesis se usan, como ya dijimos, para alterar la precedencia y asociatividad natural o bien para que quede explícita la precedencia y asociatividad que deseamos. En la tabla 2.4 damos las precedencias y asociatividades de los operadores aritméticos y lógicos más comunes. En el orden en que aparecen, la precedencia va de mayor a menor. Los operadores que tienen la misma precedencia aparecen en el mismo renglón de la tabla.

Tabla 2.4. Precedencia y asociatividad de operadores comunes

Operador	Descripción	Asociatividad
$+$ $-$ \neg	operadores unarios prefijos	no aplica
$**$	exponenciación	derecha
\cdot $/$ \div mod gcd	producto, división, módulo y máximo común divisor	izquierda
$+$ $-$	suma y resta binarias	izquierda
$=$ $<$ $>$	comparadores	izquierda
\wedge \vee	conjunción y disyunción	izquierda
\rightarrow	implicación	derecha
\leftrightarrow	bicondicional	izquierda

Como podemos observar de la tabla anterior, en ausencia de paréntesis la evaluación de $p \rightarrow q \rightarrow r$ debe realizarse asociando $p \rightarrow (q \rightarrow r)$, ya que el operador \rightarrow asocia a la derecha. Esto quiere decir que evaluamos de derecha a izquierda, como si hubiera paréntesis alrededor de $q \rightarrow r$. En el caso de la expresión $3 + 4 \cdot 7$, la precedencia de \cdot es mayor que la de $+$ binario, por lo que se evalúa a 31 ($3 + (4 \cdot 7)$). Sin embargo, esta tabla no nos ayuda a determinar los paréntesis implícitos en expresiones como $P \wedge Q \vee R$, ya que \wedge y \vee tienen la misma precedencia, pero no son el mismo operador, por lo que no podemos utilizar la asociatividad para dirimir el conflicto. En este tipo de expresiones es costumbre poner **siempre** paréntesis para indicar la precedencia, ya que de otra manera la evaluación de la expresión es ambigua. Por lo tanto debemos escribir $(P \wedge Q) \vee R$ o bien $P \wedge (Q \vee R)$, dependiendo de cuál es la precedencia deseada.

Puede haber estados en los que la evaluación sea la misma. Veamos la evaluación de estas dos asociatividades en un estado en el que no se obtiene el mismo valor, para corroborar que en ese estado no producen el mismo resultado y por lo tanto las dos expresiones no son equivalentes.

P	Q	R	$(P \wedge Q) \vee R$	valor	$P \wedge (Q \vee R)$	valor
0	0	1	0 1 1	1	0 0 1	0

Insistimos en que el concepto de asociatividad sólo se puede aplicar cuando se trata de dos o más presencias *consecutivas* del *mismo* operador; no son los niveles de precedencia los que definen la asociatividad.

2.2.3. Sustitución textual

Supongamos que tenemos dos expresiones⁵ E y R ; sea x una variable (usualmente presente en E). Usamos la notación $E[x := R]$ o E_R^x para denotar la expresión que es la misma que E , pero donde cada presencia (*ocurrencia*, *aparición*)⁶ de x en la expresión E ha sido sustituida por la expresión (R) —así, con los paréntesis explícitos—. Llamamos *sustitución textual* al acto de sustituir todas las presencias de x en E por (R) . Cuál de las dos notaciones utilizar no es relevante, excepto que se debe elegir una de ellas y mantener esa elección. La notación $E[x := R]$ es más apropiada para computación, pero la notación E_R^x es la utilizada por los profesionales de la lógica matemática.

Es conveniente notar que la sustitución textual es una operación unaria sufixa y podemos considerar a $[x := R]$, o bien $_R^x$, como el operador. En este caso no es una operación de números a números, como la suma y el producto, o de proposiciones a proposiciones como la negación o conjunción, sino que se trata de una operación de expresiones cualesquiera en expresiones cualesquiera. Este operador es el primero que aparece en una tabla

⁵Nos referimos a expresiones de cualquier tipo.

⁶Del inglés *occurrence*.

de precedencias (se le asigna la precedencia más alta de todos los operadores). Esto debe tomarse en cuenta cuando veamos a cuál expresión es a la que afecta la sustitución: no es lo mismo

$$a + b[a := x + y] \text{ (o bien } a + b_{x+y}^a \text{) que } (a + b)[a := x + y] \text{ (o bien } (a + b)_{x+y}^a \text{),}$$

pues en el primer caso la única expresión a la que se refiere la sustitución es b , mientras que en el segundo caso es $(a + b)$. En ambos casos, y dado que la sustitución textual es lo primero que se va a ejecutar, toma como operando al grupo que se encuentra a su izquierda, que en el primer caso consiste únicamente de b mientras que en el segundo caso, dado que se usaron paréntesis, consiste de $(a + b)$.

Podemos ver algunos ejemplos en la tabla 2.5 que se encuentra a continuación, utilizando ambas notaciones por el momento, aunque después usaremos la que indicamos como la más adecuada para computación.

Tabla 2.5. Ejemplos de sustitución textual

Expresado como $E[x := R]$	Expresado como E_R^x	Resultado
1. $a + b[a := x + y]$	$a + b_{x+y}^a$	$a + b$
2. $(a + b)[a := x + y]$	$(a + b)_{x+y}^a$	$((x + y) + b)$
3. $(x \cdot y)[x := z + 2]$	$(x \cdot y)_{z+2}^x$	$((z + 2) \cdot y)$
4. $(4 \cdot a \cdot b)[a := b]$	$(4 \cdot a \cdot b)_b^a$	$(4 \cdot (b) \cdot b)$
5. $(p \rightarrow q)[p := \text{false}]$	$(p \rightarrow q)_{\text{false}}^p$	$((\text{false}) \rightarrow q)$
6. $(p \rightarrow p \vee q)[p := p \vee q]$	$(p \rightarrow p \vee q)_{p \vee q}^p$	$((p \vee q) \rightarrow (p \vee q) \vee q)$
7. $(5 \cdot x)[x := 2 + 6]$	$(5 \cdot x)_{2+6}^x$	$(5 \cdot (2 + 6))$

Deseamos hacer hincapié sobre los siguientes puntos:

- Debe quedar claro el porqué la sustitución se define poniendo entre paréntesis a R dentro de E : si no lo hiciésemos así, correríamos el riesgo de alterar los paréntesis implícitos de la expresión. En la sustitución 7 del ejemplo, si evaluamos la expresión resultante obtenemos 40, pero si no pusiéramos los paréntesis alrededor de $2 + 6$, la expresión se evaluaría a 16, de acuerdo con la precedencia de los operadores en la expresión resultante.
- R , la expresión por la que vamos a sustituir, puede o no tener presencias de x , la variable a la que vamos a sustituir.
- Si E no tiene ninguna presencia de x , la expresión queda exactamente igual a como estaba, es decir $E[x := R] = E$ si x no aparece en E .
- Si hay varias presencias de x en E , como es el caso del ejemplo 6, se piensa en la sustitución hecha *simultáneamente* a cada presencia de x en E . Es como si marcáramos

mos las posiciones de x en E , después ponemos una caja en lugar de la variable y después colocamos en esas cajas a (R) . Si es que x aparece en R , no regresamos a sustituir estas presencias de x en el resultado.

- Es común que en el resultado queden paréntesis que no aportan nada, por ejemplo aquellos que rodean a una variable sola. En este caso, y cuando la eliminación de los paréntesis no afecte la precedencia y asociatividad del resultado, éstos pueden eliminarse. Esto también se refiere a los paréntesis que utilizamos para rodear a la expresión sobre la que queremos hacer la sustitución. En adelante mantendremos los paréntesis sólo en aquellos casos en que sean estrictamente necesarios, es decir, cuando quitarlos altere la precedencia y asociatividad de la expresión resultante.

Si tenemos una lista de variables $\mathbf{x} : x_1, x_2, \dots, x_n$ distintas y una lista de expresiones $\mathbf{R} : R_1, R_2, \dots, R_n$ (no forzosamente distintas), podemos definir la sustitución textual simultánea $E[\mathbf{x} := \mathbf{R}]$ ($E_{\mathbf{R}}^{\mathbf{x}}$) como el reemplazo simultáneo de cada una de las variables de la lista \mathbf{x} por su correspondiente expresión en la lista \mathbf{R} . Esto es, x_1 se reemplaza con R_1 , x_2 con R_2 y así sucesivamente. Por ejemplo, $(p \wedge q)[p, q := 1, 0]$ es $((1) \wedge (0))$, cuyo valor es 0, mientras que $(p \wedge q)[p, q := 1, p]$ es $(1 \wedge p)$, ya que no se puede “regresar” a hacer la sustitución textual de la variable x –si ésta aparece en la expresión R –, en la expresión resultante.

Un punto importante a notar es que la sustitución textual se utiliza únicamente para sustituir presencias de variables, **no** de expresiones ni de constantes.

Finalmente, observemos que una expresión $E[x := R][z := S]$ debe entenderse como $(E[x := R])[z := S]$, donde E , R y S son expresiones y x y z son variables; esta operación se define como una copia de E en la que las presencias de x fueron sustituidas por R y, en esa copia, las presencias de z fueron sustituidas por S . Es la aplicación de una segunda sustitución textual, la de la derecha, al resultado de una primera sustitución textual. Es importante notar que, en general, $E[x := R][z := S]$ es distinto a $E[x, z := R, S]$, como se puede ver en las siguientes sustituciones:

$$\begin{aligned} (p \rightarrow p \vee q)[p := q][q := p] & \text{ es } p \rightarrow p \vee p \\ (p \rightarrow p \vee q)[p, q := q, p] & \text{ es } q \rightarrow q \vee p \end{aligned}$$

Variables escondidas en la sustitución textual

Es usual asignar una variable a una expresión para que sea más sencillo manipularla. Por ejemplo, podemos decidir asignar a la variable Q :

$$Q : \frac{-b + \sqrt{b^2 - 4 \cdot a \cdot c}}{2 \cdot a}$$

y utilizar esta asociación para, en lugar de escribir

$$x_1 = (-b + \sqrt{b^2 - 4 \cdot a \cdot b}) / (2 \cdot a)$$

podamos escribir $x_1 = Q$. Pero entonces Q tiene tres variables escondidas, a , b y c , y una sustitución de la forma $Q[a := 3]$ se debe interpretar como

$$\left((-b + \sqrt{b^2 - 4 \cdot a \cdot b}) / (2 \cdot a) \right) [a := 3]$$

cuyo resultado es $(-b + \sqrt{b^2 - 4 \cdot 3 \cdot b}) / (2 \cdot 3) = (-b + \sqrt{b^2 - 12 \cdot b}) / 6$.

Queremos hacer notar que la evaluación de una expresión consiste en, simplemente, hacer una sustitución textual en la expresión, donde por cada variable definida en el estado en que se desea evaluar la expresión, se le sustituye por el valor de la variable en ese estado. Después, si es posible, se ejecutan las operaciones necesarias para obtener el valor de la expresión en ese estado. También usaremos sustitución textual cuando veamos razonamiento ecuacional.

Ejercicios

2.2.1.- Coloque los paréntesis en las siguientes expresiones de acuerdo a la precedencia y asociatividad de los operadores, sin preocuparse por la evaluación de la expresión:

- a) $-b + b * 2 - 4 \cdot a \cdot c / 2 \cdot a$ c) $a < b \wedge b < c \rightarrow a < b$
 b) $p \wedge q \vee r \rightarrow s \leftrightarrow p \vee q$ d) $a \cdot b < a \cdot c \leftrightarrow a > 0 \wedge b > c$

2.2.2.- Para cada expresión que se da a continuación, evalúe la expresión en cada uno de los estados que se proporcionan:

Expresión	Estados	Evaluación
a) $a^2 + (b \cdot c)$	$\{a = 5, b = 3, c = 6\}$	
	$\{a = -2, b = 1, c = 11, d = 3\}$	
	$\{d = 3, b = 4, c = 10\}$	
	$\{a = 3, b = 0\}$	
b) $p \rightarrow q \leftrightarrow q \rightarrow r$	$\{p = 1, q = 0, r = 1\}$	
	$\{p = 0, r = 1\}$	
	$\{p = 1, r = 0\}$	
	$\{p = 1, q = 1, r = 1\}$	

2.2.3.- Ejecute las siguientes sustituciones textuales, fijándose bien en la colocación de los paréntesis. Quite los paréntesis que no sean necesarios (que sean redundantes).

- a) $x[x := b + 2]$ c) $(x + x \cdot 2)[y := x \cdot y]$
 b) $(x + y \cdot x)[x := b + 2]$ d) $(x + x \cdot y + x \cdot y \cdot z)[x := x + y]$

2.2.4.- Ejecute las siguientes sustituciones textuales simultáneas, fijándose bien en la colocación de los paréntesis. Quite los paréntesis que sean redundantes.

$$a) x + y \cdot x[x, y := b + 2, x + 2]$$

$$c) (x + y \cdot 2)[y, x := x \cdot y, x \cdot x]$$

$$b) (x + y \cdot x)[x, y := x \cdot y, x \cdot y]$$

$$d) (x + x \cdot y + x \cdot y \cdot z)[x, y := y, x]$$

2.2.5.- Ejecute las siguientes sustituciones textuales, fijándose bien en la colocación de los paréntesis. Quite los paréntesis que no sean necesarios.

$$a) x + y \cdot x[x := y + 2][y := y \cdot x]$$

$$c) (x + x \cdot 2)[x, y := x, z][x := y]$$

$$b) (x + y \cdot x)[x := y + 2][y := y \cdot x]$$

$$d) (x + x \cdot y + x \cdot y \cdot z)[x, y := y, x][y := 2 \cdot y]$$

2.2.6.- Exprese la evaluación de las expresiones en la pregunta 2.2.2, utilizando sustitución textual simultánea.

2.3. Análisis sintáctico de expresiones lógicas

En general, una expresión es una cadena o palabra construida mediante símbolos de un alfabeto dado. Sin embargo, no todas las cadenas que construyamos simplemente pegando símbolos van a ser expresiones útiles, sino únicamente aquellas construidas de acuerdo a una gramática diseñada con ese propósito particular. El proceso de evaluación descrito anteriormente requiere que la expresión a evaluar sea sintácticamente válida; por ejemplo, no podemos ni debemos intentar evaluar una cadena de símbolos como $p \neg q$, puesto que ésta no es una expresión válida y el intento de evaluarla fracasará.

En nuestro caso, a las expresiones generadas de manera legítima por la gramática de la lógica proposicional les llamamos *expresiones lógicas*, *proposiciones* o bien *fórmulas*. Por ejemplo, $P \wedge Q$ es una fórmula si es que garantizamos que P y Q son, a su vez, fórmulas. El proceso de evaluación de una expresión debe ser precedido por el proceso de reconocer cuándo una cadena de símbolos es una fórmula bien construida o formada⁷; este proceso se conoce como *análisis sintáctico*. En nuestro caso particular, la pregunta que nos interesa responder es ¿cuándo una cadena de símbolos es una expresión lógica? Hasta ahora la única manera de responder es derivando dicha cadena mediante las reglas de la gramática; sin embargo, este proceso puede ser largo y tedioso, y si bien esta es la manera usual de implementar el proceso de análisis sintáctico, nos gustaría tener un proceso más simple y directo para nuestro uso. A continuación nos serviremos de la operación de sustitución textual para verificar cuándo una cadena de símbolos es una fórmula bien formada.

⁷En inglés se les conoce como *well formed formulas* o *wff*.

2.3.1. Esquemas

En matemáticas es común asociar *identificadores* a ciertas expresiones con el propósito de abreviar su escritura; podemos escribir por ejemplo A para denotar a la fórmula $p \vee q$. Sin embargo, A no es una variable proposicional, pues para obtener su valor es necesario evaluar la fórmula $p \vee q$, a partir de los valores de las variables proposicionales p y q .

Un identificador es entonces una especie de variable informal, conocida entre los lógicos como *meta variable*. A continuación fijamos una definición de esquema.

Definición 2.6 (esquema) Un *esquema* es una expresión construida de manera similar a las fórmulas pero usando, en algunos casos, identificadores en vez de variables proposicionales.

Si bien esta definición es informal pues el concepto de identificador no ha sido definido con precisión, con ella nos basta.

Ejemplo 2.7. Si A y B son identificadores, entonces $A \wedge B$ es un *esquema*.

Ejemplo 2.8. La expresión $(A \rightarrow B)$ es un esquema, y si $A = (p \wedge q)$ y $B = (p \vee q)$ entonces nos proporciona una forma más concisa de escribir

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

Ejemplo 2.9. Si p es una variable proposicional y $A = (p \rightarrow q)$, la fórmula $(p \wedge \neg A)$ es un esquema que proporciona una forma más concisa de escribir $(p \wedge \neg (p \rightarrow q))$.

La sustitución textual, en combinación con el concepto de esquema, proporciona una manera simple para decidir si una expresión es una fórmula bien formada. Por ejemplo, ¿cómo podemos verificar si la expresión $p \wedge \neg q \rightarrow r \wedge s$ es una implicación?; basta ver que dicha fórmula se obtiene a partir del esquema de implicación $A \rightarrow B$, en el caso particular en que los identificadores se sustituyan (*instancien*) con $A = p \wedge \neg q$ y $B = r \wedge s$.

Definición 2.7 Instanciar un esquema consiste en hacer una sustitución textual simultánea de cero o más identificadores en el esquema, por fórmulas bien construidas, que pueden o no involucrar a identificadores que aparecen originalmente en el esquema.

Un esquema tiene tantas instancias como fórmulas bien formadas podamos usar en la sustitución textual simultánea, esto es, un número infinito de instancias. Todo esquema es una instancia de sí mismo, ya que resulta de la sustitución textual simultánea de cero identificadores en el esquema o, visto de otra manera, donde cada identificador que aparece en el esquema es sustituido por sí mismo.

Si bien existen una infinidad de esquemas, basta identificar con nombre a los siguientes, llamados *básicos*:

	Llamamos	a una expresión de la forma
1.	negación	$(\neg A)$
2.	conjunción	$(A \wedge B)$
3.	disyunción	$(A \vee B)$
4.	condicional	$(A \rightarrow B)$
5.	equivalencia	$(A \leftrightarrow B)$

Obsérvese que toda fórmula debe ser atómica o bien corresponder a una o varias sustituciones textuales simultáneas de uno de estos cinco esquemas.

Ahora veamos ejemplos de fórmulas bien construidas. Utilizaremos paréntesis para presentar las distintas fórmulas y procederemos a comprobar que están bien construidas mediante esquemas. Haremos uso de la precedencia y asociatividad para eliminar paréntesis, cuando esto no afecte el significado de la fórmula.

Ejemplo 2.10. La expresión $((p \wedge q) \rightarrow (p \vee q))$ es una *condicional*. Para ver por qué se le asigna este nombre, veamos la sucesión de sustituciones textuales que se fueron realizando:

$$(p \rightarrow q)[p, q := p \wedge q, p \vee q] = ((p \wedge q) \rightarrow (p \vee q))$$

donde quitando los paréntesis superfluos queda

$$p \wedge q \rightarrow p \vee q.$$

Como el esquema original del que partimos es el de la implicación, la instanciación dada es por ende una implicación.

Ejemplo 2.11. El esquema $\neg A \rightarrow P \vee Q$ es una condicional, porque al restaurar los paréntesis implícitos en la expresión, dada la precedencia y asociatividad de los distintos operadores que aparecen, obtenemos $((\neg A) \rightarrow (P \vee Q))$.

$$\begin{aligned} (P \rightarrow Q)[P, Q := A, P \vee Q][A := \neg A] &= \\ &= ((A) \rightarrow (P \vee Q))[A := \neg A] \\ &= ((\neg A) \rightarrow (P \vee Q)), \end{aligned}$$

donde quitando los paréntesis superfluos, queda

$$\neg A \rightarrow P \vee Q,$$

Como el esquema original del que partimos es una condicional, decimos que el esquema $\neg A \rightarrow P \vee Q$ también es una condicional.

Ejemplo 2.12. La fórmula $(p \leftrightarrow q) \wedge (r \leftrightarrow p) \leftrightarrow (p \leftrightarrow q) \wedge (r \leftrightarrow q)$ es una equivalencia. Nuevamente veamos los paréntesis implícitos, de acuerdo a las reglas de precedencia y asociatividad:

$$\left(((p \leftrightarrow q) \wedge (r \leftrightarrow p)) \leftrightarrow ((p \leftrightarrow q) \wedge (r \leftrightarrow q)) \right)$$

y veamos la sucesión de sustituciones textuales a partir del esquema $(A \leftrightarrow B)$:

$$\begin{aligned} (A \leftrightarrow B) [A, B := P \wedge Q, P \wedge R] [P, Q, R := p \leftrightarrow q, r \leftrightarrow p, r \leftrightarrow q] &= \\ &= ((P \wedge Q) \leftrightarrow (P \wedge R)) [P, Q, R := p \leftrightarrow q, r \leftrightarrow p, r \leftrightarrow q] \\ &= (((p \leftrightarrow q) \wedge (r \leftrightarrow p)) \leftrightarrow ((p \leftrightarrow q) \wedge (r \leftrightarrow q))), \end{aligned}$$

donde quitando los paréntesis superfluos, nos lleva a:

$$(p \leftrightarrow q) \wedge (r \leftrightarrow p) \leftrightarrow (p \leftrightarrow q) \wedge (r \leftrightarrow q).$$

Obsérvese que en este ejemplo primero transformamos el esquema básico de implicación en un esquema más cercano a la fórmula original, para después instanciar con las fórmulas adecuadas y obtener el resultado deseado.

Del último ejemplo podemos concluir que el proceso de análisis mediante sustituciones textuales empieza a resultar complicado, por lo que nos gustaría dar una definición del proceso, susceptible de aplicarse mecánicamente, algo que desarrollamos a continuación.

2.3.2. Rango y conectivo principal

Para mecanizar el proceso de análisis sintáctico de una expresión nos serviremos, además de la sustitución textual y el uso de esquemas, de un proceso de descomposición en expresiones sintácticamente más simples, las cuales son más sencillas de analizar. Dicha descomposición utiliza los conceptos de rango de un conectivo lógico y conectivo principal de una fórmula que a continuación definimos.

Definición 2.8 (rango, alcance) El concepto de *rango* o *alcance* de un conectivo en una fórmula o esquema E se define, con base en los esquemas básicos, como sigue:

- Si E es instancia de $\neg A$, entonces el rango de \neg en E es A .
- Si E es instancia de uno de los esquemas básicos binarios $A \star B$, donde \star es un conectivo lógico binario, entonces el conectivo \star en E tiene un *rango izquierdo* que es A y un *rango derecho* que es B .

Obsérvese que el rango o rangos de un conectivo (operador) en una expresión corresponden a los operandos; en caso de que no estén explícitamente indicados, se obtienen tomando en cuenta las reglas de asociatividad y precedencia ya estudiadas. Por ejemplo:

- En el esquema $\neg A \wedge B$, el rango del operador \neg es únicamente el identificador A . Si queremos que el rango sea $A \wedge B$ debemos encerrar este esquema entre paréntesis, obteniendo $\neg (A \wedge B)$.
- En la fórmula $A \wedge B \wedge C$ el rango izquierdo del segundo conectivo \wedge es la fórmula $(A \wedge B)$, ya que como no hay paréntesis, las reglas de precedencia y asociatividad hacen que la colocación de los paréntesis implícita de la fórmula sea $((A \wedge B) \wedge C)$.

Otro concepto importante es el de *conectivo principal*. Si una expresión E resulta ser instancia de uno de los esquemas básicos, entonces el conectivo que observamos en

el esquema correspondiente será también el conectivo principal de E . Por ejemplo, si $E = (p \vee q) \wedge C$, entonces el conectivo principal de E es \wedge , puesto que

$$E = (A \wedge B)[A, B := p \vee q, C].$$

Veamos un ejemplo más elaborado.

Ejemplo 2.13. Consideremos el esquema $(A \wedge B \wedge C) \vee (A \rightarrow B \rightarrow C)$. El análisis sintáctico de este esquema es el siguiente:

- Para el esquema original:
 - El conectivo principal es \vee .
 - El rango izquierdo es $(A \wedge B \wedge C)$.
 - El rango derecho es $(A \rightarrow B \rightarrow C)$.
- Para el rango izquierdo:
 - Los paréntesis implícitos son $((A \wedge B) \wedge C)$.
 - El conectivo principal es el segundo \wedge .
 - El rango izquierdo corresponde a $(A \wedge B)$.
 - El rango derecho corresponde a C .
- Para el rango derecho, podemos observar que:
 - Los paréntesis implícitos son $(A \rightarrow (B \rightarrow C))$.
 - El conectivo principal es el primer \rightarrow .
 - El rango izquierdo es A .
 - El rango derecho es $(B \rightarrow C)$.

Este proceso puede seguir hasta que ya tengamos esquemas o fórmulas que no correspondan a los esquemas básicos, es decir esquemas que consistan de un único identificador o bien variables proposicionales, en las que no tienen ningún significado los conceptos de conectivo principal o rango. Estos casos corresponden al fin del proceso de análisis sintáctico. Como toda fórmula consiste de una combinación de conectivos y proposiciones atómicas, la descomposición en rangos no puede durar para siempre.

2.3.3. Análisis de proposiciones compuestas

Existen dos clases de métodos para el análisis de una expresión, los métodos *generadores* que construyen la expresión deseada a partir de símbolos o esquemas iniciales utilizando ciertas reglas u operaciones; y los métodos *analíticos* que consisten en partir de la supuesta expresión dada y realizar un proceso de descomposición hasta llegar a expresiones básicas, donde el proceso de análisis es directo. Los métodos de gramáticas y árboles de derivación y de instanciación de esquemas básicos son generadores.

A continuación veremos un método analítico basado en la descomposición de una expresión utilizando su conectivo principal y rangos correspondientes. Haremos explícita esta descomposición usando un árbol, cuya raíz consistirá de la fórmula completa. En cada nivel

que bajemos del árbol, identificaremos al conectivo principal de la fórmula y procederemos a colgar de la fórmula al conectivo y a su(s) rango(s). La idea principal es que si E es una expresión compuesta, los rangos del conectivo principal son expresiones, a las que les podemos aplicar el mismo procedimiento. Veamos un ejemplo:

Ejemplo 2.14. Si el equipo mexicano llega a cuartos de final del Mundial, todo mundo lo admirará y los jugadores se volverán ricos; pero si no llega, nada pasará.

Hagamos una asignación a variables proposicionales:

p : el equipo mexicano llega a cuartos de final r : los jugadores se vuelven ricos

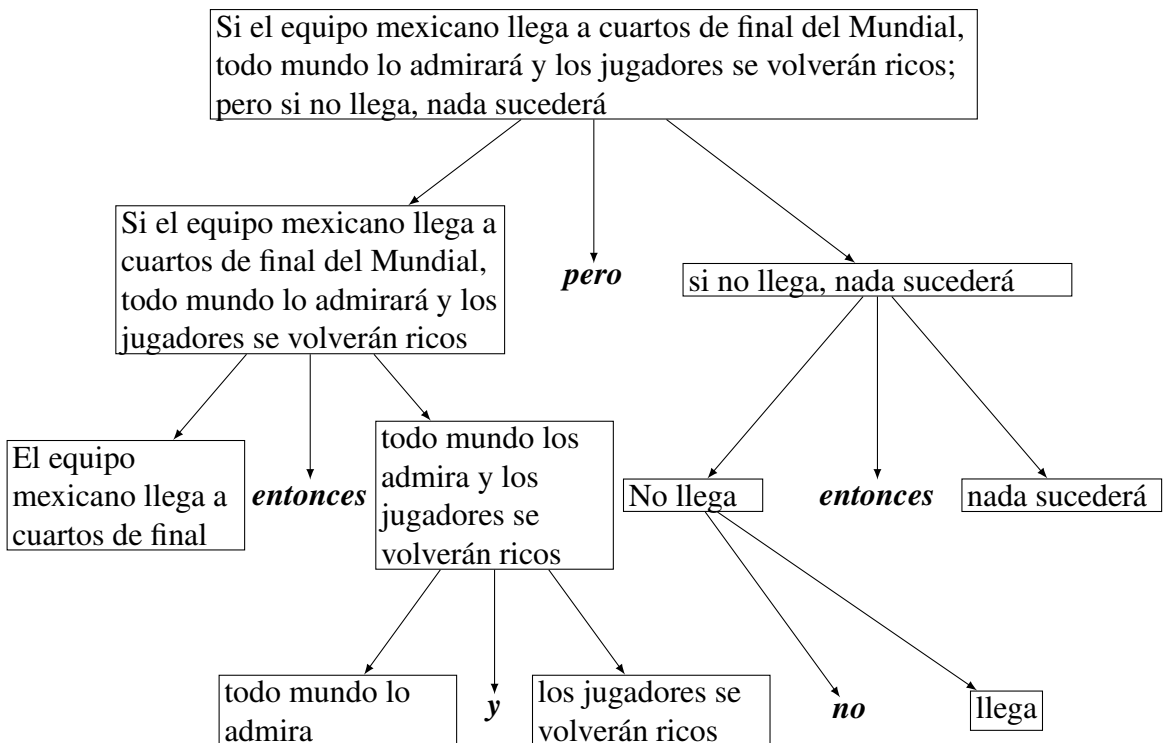
q : todo mundo admira al equipo mexicano s : nada pasará

Hagamos la traducción a una fórmula con paréntesis:

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

y veamos cómo queda el árbol producto del análisis sintáctico de esta fórmula en la figura 2.2. En este caso hemos elegido presentar el árbol con las frases en español de manera que se pueda observar la descomposición directamente con enunciados más simples. Obsérvese que las hojas de este árbol corresponden a proposiciones atómicas que ya no pueden descomponerse.

Figura 2.2. Análisis de proposición compuesta



El proceso de analizar una expresión es un proceso *recursivo*, que consiste de los siguientes pasos:

1. Si la proposición es atómica, el análisis termina.
2. Si la proposición no es atómica:
 - a) Definir el conectivo principal.
 - b) Si el conectivo es unario, analizar la proposición que corresponde al rango derecho.
 - c) Si el conectivo es binario, analizar la proposición que corresponde al rango izquierdo y la proposición que corresponde al rango derecho.

Veamos otro ejemplo, esta vez sin remitirnos en el árbol a las frases en español.

Ejemplo 2.15. Si el anuncio tiene éxito, toda la producción se va a vender y el dueño se volverá rico; pero si el anuncio no tiene éxito, la inversión se habrá perdido.

Variables proposicionales:

p : el anuncio tiene éxito

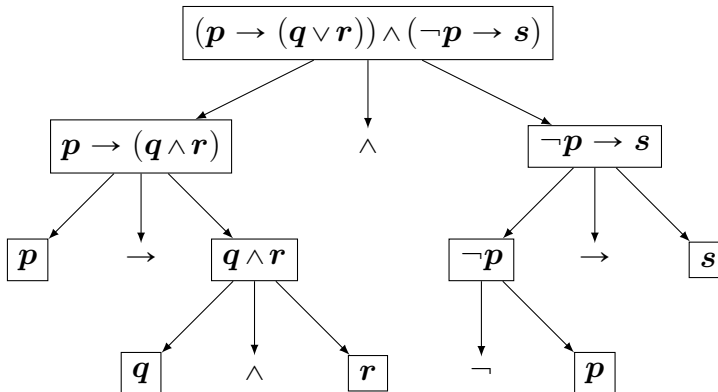
r : el dueño se vuelve rico

q : toda la producción se vende

s : la inversión se pierde

Podemos ver el árbol usando las variables proposicionales y los conectivos lógicos en la figura 2.3, que se encuentra a continuación.

Figura 2.3. Análisis de una proposición



En este momento resulta claro que dada una expresión sintácticamente válida, se puede construir el árbol de análisis sintáctico partiendo directamente de ella; si la expresión está completamente expresada con paréntesis (todos los paréntesis que definen precedencia y asociatividad son explícitos), el proceso es inmediato, mientras que si no es así habrá que usar los criterios de precedencia y asociatividad. Los niveles del árbol se van construyendo de adentro hacia afuera (de abajo hacia arriba) y de izquierda a derecha para aquellos operadores que asocian a la izquierda, pero de derecha a izquierda para aquellos que asocian a la derecha.

Más aún, el proceso de análisis sintáctico facilita el proceso de evaluación, puesto que una vez construido el árbol las hojas corresponden a fórmulas atómicas, las cuales se pueden evaluar directamente continuando el proceso de evaluación, según lo dictado por las tablas de verdad de los conectivos principales.

2.3.4. Tautologías y sustitución

Hasta ahora la única manera de verificar si una fórmula dada A es una tautología es construyendo su tabla de verdad. Sin embargo, al crecer el número de variables la tabla de verdad contiene cada vez más renglones y su construcción se vuelve complicada, ineficiente y eventualmente imposible. Como ejemplo considérese el esquema $A \wedge B \rightarrow B$. Es fácil ver mediante una tabla de verdad de cuatro renglones que $\models A \wedge B \rightarrow B$. Por otra parte, considérese la expresión $p_1 \wedge p_2 \wedge \dots \wedge p_{99} \wedge p_{100} \rightarrow p_{100}$; ¿cómo mostrar que se trata de una tautología? La tabla de verdad tendrá 2^{100} renglones, así que resulta imposible construirla. Afortunadamente, la operación de sustitución permite generar más tautologías a partir de tautologías conocidas.

Una vez que se conoce que $\models A$, no importa si en A sustituimos cualquier variable proposicional por una expresión, el resultado va a seguir siendo una tautología. Esto se formaliza en el siguiente teorema, cuya demostración omitimos.

Teorema 2.1 (Propiedad de sustitución) *Sea A una fórmula o esquema tal que $\models A$ y sean*

p_1, p_2, \dots, p_n variables proposicionales. Si B_1, B_2, \dots, B_n son expresiones lógicas o esquemas arbitrarios, entonces

$$\models A[p_1, p_2, \dots, p_n := B_1, B_2, \dots, B_n];$$

es decir, las sustituciones textuales en tautologías generan tautologías.

Usando este resultado y observando que

$$(A \wedge B \rightarrow B)[A, B := p_1 \wedge p_2 \wedge \dots \wedge p_{99}, p_{100}] = p_1 \wedge p_2 \wedge \dots \wedge p_{100} \rightarrow p_{100}$$

concluimos que $\models p_1 \wedge p_2 \wedge \dots \wedge p_{100} \rightarrow p_{100}$.

Veamos otros ejemplos.

Ejemplo 2.16. Demostrar que $\models (p \wedge q) \vee \neg (p \wedge q)$.

Identificamos en el ejemplo una disyunción de una expresión y su negación, por lo que buscamos algún esquema tautológico que tenga esta misma forma. Sabemos que $p \vee \neg p$ es una tautología. Entonces

$$(p \vee \neg p)[p := p \wedge q] = (p \wedge q) \vee \neg (p \wedge q)$$

por lo que esta expresión es también una tautología.

Ejemplo 2.17. Demostrar que $\models R \rightarrow (P \vee Q) \vee R$. Debemos buscar un esquema para “deshacer” las sustituciones que se hayan hecho. En el nivel más alto el esquema es

$$A \rightarrow B.$$

Busquemos ahora una tautología que involucre implicación y que en el rango derecho tenga una conjunción. Sabemos que $\models p \rightarrow p \vee q$ es una tautología (mostramos su tabla de verdad al inicio de la sección). Como la disyunción tiene la propiedad de conmutatividad, tenemos que $p \rightarrow p \vee q$ es lo mismo que $p \rightarrow q \vee p$. Por el Teorema de sustitución, tenemos:

$$(p \rightarrow q \vee p)[p, q := q, p] = q \rightarrow p \vee q.$$

Este último esquema tautológico nos sirve, pues lo que buscamos es que el rango derecho de la disyunción coincida con el rango izquierdo de la condicional.

A continuación observamos que el rango izquierdo de la disyunción es una subexpresión compuesta, no nada más una fórmula atómica, por lo que ahí también se llevó a cabo una sustitución textual, que si la “deshacemos” queda como sigue:

$$\begin{aligned} (q \rightarrow p \vee q)[q, p := R, P \vee Q] &= \\ &= (R) \rightarrow (P \vee Q) \vee (R) \\ &= R \rightarrow (P \vee Q) \vee R \end{aligned}$$

Reglas de inferencia

Una vez que se ha mostrado la correctud de un argumento lógico, éste se convierte en un esquema de argumento que sigue siendo correcto al sustituir algunos de sus identificadores por fórmulas arbitrarias, puesto que el esquema correspondiente a su fórmula asociada es una tautología, que se preserva bajo sustituciones como lo asegura el teorema 2.1. En tal caso hablamos ya no de un argumento correcto sino de una *regla de inferencia*.

Definición 2.9 (regla de inferencia) Una *regla de inferencia* es un esquema de argumento correcto.

Por ejemplo, dado que los argumentos de los ejemplos 2.4 y 2.6 –el primero conocido como *modus ponens*– son correctos podemos enunciarlos como esquemas:

$$\begin{array}{cc} A \rightarrow B & A \wedge B \rightarrow C \\ \hline A & A \\ \hline B & B \rightarrow C \end{array}$$

Obsérvese que una vez que un argumento correcto se transforma en regla de inferencia, al ser correcto, el símbolo \therefore desaparece en la conclusión.

Ejercicios

2.3.1.- Clasifique las proposiciones que aparecen en la siguiente página, en alguna de las categorías que se listan a continuación, justificando la respuesta mediante el uso de esquemas:

- | | |
|-------------------|-----------------|
| (a) negación | (b) disyunción |
| (c) conjunción | (d) condicional |
| (e) bicondicional | |

Fórmula	Categoría
$\neg P \rightarrow \neg Q$	
$P \leftrightarrow Q \leftrightarrow (P \rightarrow Q) \wedge (Q \rightarrow P)$	
$Q \wedge P \rightarrow Q \rightarrow P$	
$(P \rightarrow Q) \wedge (\neg P \rightarrow Q) \rightarrow Q$	
$P \rightarrow Q \leftrightarrow \neg Q \rightarrow \neg P$	

2.3.2.- Para las siguientes proposiciones, diga a cuál esquema básico corresponden, reha-
ciendo las sustituciones textuales que se hayan llevado a cabo. En caso de am-
bigüedad respecto a la asociatividad de dos operadores distintos con la misma prece-
dencia, se debe asociar desde la izquierda.

- | | |
|---|---|
| (a) $p \rightarrow q \wedge q \rightarrow p$ | (e) $\neg (p \vee \neg q \wedge p)$ |
| (b) $r \wedge \neg q \leftrightarrow \neg r \wedge q$ | (f) $\neg p \rightarrow q$ |
| (c) $p \rightarrow q \rightarrow r$ | (g) $\neg (p \wedge \neg q)$ |
| (d) $p \vee q \wedge \neg p \wedge q \rightarrow q \wedge \neg q$ | (h) $\neg p \wedge (\neg p \wedge q) \vee (p \wedge (p \wedge \neg q))$ |

2.3.3.- De los siguientes enunciados, defina el conectivo principal. Para cada operador: si el
operador es binario especifique su rango izquierdo y su rango derecho; si el operador
es unario, especifique su rango (derecho).

- | | |
|---|--|
| (a) $p \vee (\neg p \wedge q) \rightarrow p \vee q$ | (d) $(p \rightarrow q) \rightarrow p \vee q \rightarrow q$ |
| (b) $\neg (p \wedge q \rightarrow p \vee q)$ | (e) $\neg(p \vee q \rightarrow r)$ |
| (c) $\neg p \wedge (q \vee p) \wedge \neg q$ | |

2.3.4.- Dé el árbol de análisis sintáctico de cada una de los siguientes esquemas:

- | | |
|--|---|
| a) $P \wedge Q \wedge R \rightarrow P$ | c) $P \rightarrow Q \rightarrow R \vee S \vee P$ |
| b) $P \rightarrow Q \leftrightarrow \neg Q \rightarrow \neg P$ | d) $P \rightarrow Q \wedge R \rightarrow S \rightarrow P \rightarrow S$ |

2.3.5.- Construya el árbol de análisis sintáctico para cada una de las siguientes fórmulas

- a) $\neg\neg p \wedge \neg q \rightarrow s \leftrightarrow \neg s \rightarrow \neg p \vee q$
- b) $\neg p \vee q \rightarrow p \wedge \neg q \rightarrow \neg p \vee \neg q \rightarrow \neg p \wedge \neg q$
- c) $p \wedge q \rightarrow p \vee q \rightarrow \neg p \wedge q$

2.3.6.- Considere la siguiente fórmula A :

$$A =_{def} p \leftrightarrow q \rightarrow \neg(p \wedge r)$$

Realice lo siguiente:

- a) Determine el esquema básico de A mediante sustitución textual.
- b) Construya el árbol de análisis sintáctico de A .
- c) Realice la siguiente sustitución textual, eliminando en el resultado todos los paréntesis que sea posible.

$$A[p, q := p \vee q, s][p, q := \neg p, \neg s]$$

2.4. Equivalencia lógica

El concepto de expresiones equivalentes es imprescindible para todo tipo de razonamiento. Decimos que dos expresiones son equivalentes si y sólo si en todos y cada uno de sus posibles estados se evalúan a lo mismo.

Por ejemplo, podemos comprobar usando una tabla de verdad, que las expresiones $\neg\neg P$ y P son equivalentes:

P	$\neg P$	$\neg(\neg P)$
1	0	1
0	1	0

Lo que debemos observar es que, *renglón por renglón*, el valor correspondiente a P es el mismo que el valor correspondiente a $\neg\neg P$. No se interprete esta definición como que estamos exigiendo tener el mismo valor en todos los renglones, esto es, que todos los renglones valieran 0 o todos los renglones valieran 1.

En el caso de expresiones lógicas el concepto de equivalencia está relacionado con un tipo particular de tautología. Si tenemos una bicondicional ($A \leftrightarrow B$) que es una tautología, entonces decimos que tenemos una *equivalencia lógica* :

Definición 2.10 (equivalencia lógica) Sean A, B dos fórmulas. Si $A \leftrightarrow B$ es una tautología, entonces decimos que A y B son *lógicamente equivalentes* y lo denotamos por $A \equiv B$. Esto es lo mismo que decir

$$A \equiv B \quad \text{si y sólo si} \quad \models A \leftrightarrow B.$$

La tabla 2.6, a continuación, resume algunas equivalencias lógicas de importancia, las cuales pueden comprobarse mediante el uso de tablas de verdad.

Tabla 2.6. Leyes de equivalencia de la lógica proposicional

Asociatividad:	$(P \wedge Q) \wedge R \equiv P \wedge (Q \wedge R)$	(2.13)
	$(P \vee Q) \vee R \equiv P \vee (Q \vee R)$	(2.14)
Identidad:	$P \vee \text{false} \equiv P$	(2.15)
	$P \wedge \text{true} \equiv P$	(2.16)
Idempotencia:	$P \vee P \equiv P$	(2.17)
	$P \wedge P \equiv P$	(2.18)
Dominación	$P \vee \text{true} \equiv \text{true}$	(2.19)
(o elemento nulo):	$P \wedge \text{false} \equiv \text{false}$	(2.20)
Conmutatividad:	$P \vee Q \equiv Q \vee P$	(2.21)
	$P \wedge Q \equiv Q \wedge P$	(2.22)
Tercero excluido:	$P \vee \neg P \equiv \text{true}$	(2.23)
Contradicción:	$P \wedge \neg P \equiv \text{false}$	(2.24)
Doble negación:	$\neg \neg P \equiv P$	(2.25)
Distributividad:	$P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$	(2.26)
	$P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$	(2.27)
De Morgan:	$\neg (P \wedge Q) \equiv \neg P \vee \neg Q$	(2.28)
	$\neg (P \vee Q) \equiv \neg P \wedge \neg Q$	(2.29)
Eliminación de	$P \rightarrow Q \equiv \neg P \vee Q$	(2.30)
operadores:	$P \leftrightarrow Q \equiv (\neg P \vee Q) \wedge (P \vee \neg Q)$	(2.31)
	$P \leftrightarrow Q \equiv (P \wedge Q) \vee (\neg P \wedge \neg Q)$	(2.32)

A continuación mostraremos el uso de equivalencias lógicas en particular como herramienta auxiliar imprescindible en el análisis de un argumento lógico.

2.4.1. Razonamiento ecuacional

Consideremos la igualdad aritmética $x + y + x + z = y + 2x + z$. Probablemente ninguno de nosotros dudaría de su validez, debido a la experiencia con números que tenemos desde nuestra educación básica. Más aún, si se nos pidiera una demostración formal tal vez daríamos la siguiente:

$$x + y + x + z = y + x + x + z = y + 2x + z;$$

y si se nos pidiera nuevamente una justificación tal vez apelaríamos a las igualdades

$$x + y = y + x \quad \text{y} \quad x + x = 2x.$$

Este tipo de razonamiento se conoce como razonamiento ecuacional y será parte importante del proceso de análisis de un argumento lógico.

Por lo general las fases de razonamiento ecuacional nos son tan familiares que no se mencionan explícitamente dentro del análisis de un argumento; de hecho, nosotros respetaremos esta costumbre. Sin embargo, en nuestro curso nos conciernen no sólo los aspectos puramente matemáticos de un tema, sino también el proceso de implementación, el cual es esencialmente sintáctico dado que las computadoras no entienden de significados ni son capaces de razonar como nosotros. A continuación discutimos las propiedades de la igualdad, en particular la llamada regla de Leibniz que involucra a la sustitución textual; ésta nos brindará una manera posible de implementar el razonamiento ecuacional.

Si consideramos a la igualdad como un operador (cuyo resultado es 0 o 1), podemos observar que tiene las siguientes propiedades:

Reflexividad	$X = X$
Conmutatividad	$X = Y$
	$Y = X$
Transitividad	$X = Y$
	$Y = Z$
	$X = Z$

Las últimas dos propiedades las dimos como reglas de inferencia, puesto que corresponden a argumentos correctos. Finalmente, veamos una propiedad, conocida con el nombre de regla de Leibniz, que nos va a permitir sustituir expresiones iguales en expresiones que resultarán iguales nuevamente y proporciona una manera de implementar nuestro razonamiento ecuacional usual.

$$\frac{X = Y}{E[z := X] = E[z := Y]} \quad \text{Leibniz}$$

Lo que esta regla de inferencia nos dice es que si suponemos que $X = Y$, entonces es posible tomar dos copias de la expresión E (en la que tenemos presencias de una variable z); en una de ellas sustituir a la variable z por la expresión X ; en la otra copia sustituir a la misma variable z por la expresión Y , obteniendo que las expresiones $E[z := X]$ y $E[z := Y]$ son iguales nuevamente. Es decir, la sustitución de expresiones equivalentes en expresiones iguales (o en una misma expresión) genera expresiones equivalentes.

Es importante notar que en el caso de expresiones lógicas el concepto de igualdad que se utiliza es el de equivalencia lógica, es decir, si decimos que dos expresiones lógicas A y B son iguales, queremos decir que $A \equiv B$. De manera que en este caso podemos reescribir el argumento de Leibniz de la siguiente forma:

$$\frac{X \equiv Y}{E[z := X] \equiv E[z := Y]} \quad \text{Leibniz}$$

Veamos unos ejemplos de la aplicación de esta regla de inferencia.

Ejemplo 2.18. Supongamos que $b + 3 = c + 5$, y sea E la expresión aritmética $d + e$. Entonces, tenemos la siguiente instancia de la regla de Leibniz, que se muestra a continuación.

$$\frac{b + 3 = c + 5}{(d + e)[e := b + 3] = (d + e)[e := c + 5]},$$

lo que nos permite concluir que $d + (b + 3) = d + (c + 5)$ es verdadero en aquellos estados en los que $b + 3 = c + 5$ se evalúe a verdadero. Como la suma es asociativa y podemos eliminar paréntesis superfluos, esto es lo mismo que decir $d + b + 3 = d + c + 5$.

Las situaciones en las que usualmente se usa la regla de Leibniz se dan como sigue:

- Tenemos una expresión $E[z := X] = G$. Esto quiere decir que dada una expresión cualquiera G , localizamos en ella una subexpresión a la que denotamos con X . Esta subexpresión puede aparecer más de una vez, ya que la variable “original” z también puede ocurrir más de una vez en E .
- Buscamos una expresión Y que nos convenga, tal que $X = Y$.
- Podemos entonces obtener una nueva expresión $G' = E[z := Y]$.
- La regla de Leibniz nos permite concluir que $G = G'$.

A continuación discutimos el ejemplo introductorio de esta sección.

Ejemplo 2.19. Sabemos que

$$\bullet \quad x + y = y + x \quad (2.33)$$

$$\bullet \quad x + x = 2 \cdot x \quad (2.34)$$

Sea $E = x + y + x + z$. Si deseamos simplificar esta expresión, debemos poder aplicar los dos hechos que sabemos –equivalencias (2.33) y (2.34)–. Por lo pronto, únicamente podemos aplicar la equivalencia (2.33), con dos lugares (en la expresión que queremos manipular) donde podemos hacerlo, considerando que tratamos de localizar a cualquiera de los dos lados de la igualdad:

$$\bullet \quad \boxed{x+y} + x + z \quad (\text{primer acomodo})$$

$$\bullet \quad x + \boxed{y+x} + z \quad (\text{segundo acomodo})$$

Si utilizamos el primer acomodo, entonces $X = x + y$ e $Y = y + x$, con lo que podemos sustituir lo que está en la caja por la expresión equivalente, obteniendo:

$$\boxed{y+x} + x + z$$

Pero ahora tenemos la siguiente expresión, en la que, nuevamente, podemos localizar varias subexpresiones:

$$\bullet \boxed{y+x} + x + z \quad (\text{tercer acomodo})$$

$$\bullet y + \boxed{x+x} + z \quad (\text{cuarto acomodo})$$

Pero si elegimos el tercer acomodo, regresamos a donde estábamos, por lo que no nos conviene. Mejor elegimos el cuarto acomodo, utilizando la equivalencia (2.34), con lo que tenemos $X = x + x$, $Y = 2 \cdot x$, quedándonos nuestra expresión de la siguiente forma:

$$y + \boxed{2 \cdot x} + z$$

El lector puede comprobar que también eligiendo el segundo patrón que reconocimos en la expresión original hubiésemos podido llegar al mismo resultado.

Veamos otro ejemplo aritmético a detalle.

Ejemplo 2.20.

Supongamos que queremos demostrar $\boxed{(a + b) - b = a}$ y que conocemos las siguientes equivalencias:

$$(x + y) - z = x + (y - z) \quad (2.35)$$

$$y - y = 0 \quad (2.36)$$

$$x + 0 = x \quad (2.37)$$

Entonces, podemos pensar en la siguiente demostración, utilizando la propiedad de sustitución (teorema 2.1), la regla de Leibniz y lo que ya conocemos. Como queremos demostrar que $(a + b) - b = a$, y dado que el lado izquierdo de la igualdad presenta más estructura, lo indicado es “salir” de ese lado y tratar, mediante la aplicación de la propiedad de sustitución y la regla de Leibniz, llegar a a . Es obvio que cada vez que pasamos a una nueva instancia de una regla de inferencia cualquiera, estamos utilizando la propiedad de transitividad para “encadenar” las igualdades:

Paso 1: Aplicar el Teorema 2.1.

$$((x + y) - z = x + (y - z))[x, y, z := a, b, b] = ((a + b) - b = a + (b - b))$$

La propiedad que estamos utilizando es la de sustitución: sabemos que la premisa es una igualdad válida, una tautología, $((x + y) - z = x + (y - z))$ y elegimos las sustituciones que necesitamos para obtener la expresión con la que queremos trabajar $((a + b) - b)$. De la regla de inferencia tenemos lo siguiente:

$$E \quad \text{es} \quad (x + y) - z = x + (y - z)$$

$$E[x, y, z := a, b, b] \quad \text{es} \quad (a + b) - b = a + (b - b)$$

También utilizamos este mismo teorema de sustitución para pasar de la expresión que tenemos $(y - y = 0)$ a la forma que queremos $(b - b = 0)$.

Paso 2: Volver a aplicar el Teorema 2.1.

$$\frac{y - y = 0}{(y - y = 0)[y := b]}$$

Y como $(y - y = 0)[y := b]$ es $(b - b = 0)$, tenemos ya:

$$\begin{array}{ll} (a + b) - b = a + (b - b) & \text{(por la aplicación del paso 1)} \\ b - b = 0 & \text{(por la aplicación del paso 2)} \end{array}$$

Podemos ahora utilizar la regla de Leibniz de la siguiente manera:

Paso 3: Aplicar la regla de Leibniz.

$$\frac{b - b = 0}{a + (b - b) = a + 0}$$

donde:

$$\begin{array}{lll} X & \text{es} & b - b \\ Y & & 0 \\ E & & a + z \\ E[z := X] & (a + z)[z := b - b] & = (a + (b - b)) \\ E[z := 0] & (a + z)[z := 0] & = (a + (0)), \end{array}$$

que cuando quitamos los paréntesis superfluos nos dejan

$$a + (b - b) = a + 0.$$

También sabemos que $x + 0 = x$ es una igualdad válida. Entonces podemos aplicarle sustitución textual y seguir teniendo una igualdad válida:

$$\textbf{Paso 4:} \quad \frac{x + 0 = x}{(x + 0 = x)[x := a]} .$$

Pero como $(x + 0 = x)[x := a]$ es $a + 0 = a$, tenemos la siguiente sucesión de igualdades válidas:

$$\begin{array}{l} (a + b) - b = a + (b - b) \\ b - b = 0 \\ a + (b - b) = a + 0 \\ a + 0 = a \\ \hline (a + b) - b = a \end{array}$$

Decimos entonces que hemos demostrado que $(a + b) - b = a$ es una igualdad válida.

2.4.2. Álgebra de equivalencias lógicas

Análogamente al hecho de que el razonamiento aritmético ecuacional es la base del álgebra que conocemos desde hace tiempo, en el caso de las expresiones lógicas se genera

un álgebra que manipula variables y constantes que representan valores de verdad; en particular podemos emplear equivalencias lógicas para deducir o simplificar nuevas expresiones a partir de otras ya conocidas. Ilustremos esto mediante algunos ejemplos.

Ejemplo 2.21. Sabemos que

$$\bullet P \wedge P \equiv P \quad (2.38)$$

$$\bullet P \wedge Q \equiv Q \wedge P \quad (2.39)$$

Supongamos que queremos “simplificar” la expresión $q \wedge r \wedge q \wedge s$.

Para poder aplicar el argumento de Leibniz, hagamos primero sustitución textual sobre las variables, para tener los mismos términos:

$$(q \wedge r \wedge q \wedge s)[q, r, s := P, Q, R] = P \wedge Q \wedge P \wedge R.$$

Ahora tratemos de identificar alguno de los lados de las equivalencias dentro de la expresión que tenemos. Existen dos posiciones que podemos reconocer:

- $\boxed{P \wedge Q} \wedge P \wedge R$ – lado izquierdo de (2.39)
- $P \wedge \boxed{Q \wedge P} \wedge R$ – lado derecho de (2.39)

Si aplicamos a la primera elección la igualdad, $X = Y$ con $X = P \wedge Q$ y $Y = Q \wedge P$, la regla de Leibniz nos lleva a la expresión:

$$\frac{P \wedge Q \equiv Q \wedge P}{\boxed{P \wedge Q} \wedge P \wedge R \equiv \boxed{Q \wedge P} \wedge P \wedge R}.$$

Enseguida localizamos el otro esquema que corresponde a la equivalencia dada en (2.38), al principio de esta sección, donde $X = P \wedge P$ e $Y = P$. La sustitución se hace como sigue:

$$\frac{P \wedge P \equiv P}{Q \wedge \boxed{P \wedge P} \wedge R \equiv Q \wedge \boxed{P} \wedge R},$$

por lo que terminamos con la siguiente expresión:

$$Q \wedge P \wedge R \equiv P \wedge Q \wedge R;$$

de las dos aplicaciones de Leibniz y usando la regla de transitividad podemos concluir que

$$P \wedge Q \wedge P \wedge R \equiv P \wedge Q \wedge R.$$

Si nos quedamos con la expresión de la derecha y hacemos la sustitución de las variables de regreso a q, r y s , tenemos: $(P \wedge Q \wedge R)[P, Q, R := q, r, s] = q \wedge r \wedge s$; y esta última es la simplificación final de la original.

Ejemplo 2.22. Consideremos ahora la siguiente expresión lógica $(P \wedge Q) \wedge \neg Q$. El objetivo es simplificarla lo más posible. Tenemos que:

- | | | |
|----|---|--|
| 1. | $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$ | Propiedad asociativa de \wedge |
| 2. | $(P \wedge Q) \wedge \neg Q \equiv P \wedge (Q \wedge \neg Q)$ | Sustitución textual en 1. |
| 3. | $\frac{P \wedge \neg P \equiv \text{false}}{P \wedge (Q \wedge \neg Q) \equiv P \wedge \text{false}}$ | $X = P \wedge \neg P$ e $Y = \text{false}$ |
| 4. | | Leibniz |
- y como $P \wedge \text{false} \equiv \text{false}$ Elemento nulo

ya terminamos.

De esta manera hemos demostrado que $(P \wedge Q) \wedge \neg Q \equiv \text{false}$, con la siguiente sucesión de equivalencias, utilizando la propiedad de transitividad de la equivalencia lógica:

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

En la tabla 2.6 (página 56) mostramos la lista inicial de equivalencias que vamos a utilizar para nuestro razonamiento ecuacional. Sin embargo, existen muchas otras equivalencias que se pueden derivar de las anteriores y son de gran importancia. A continuación obtenemos algunas de ellas.

Leyes de absorción:

$$P \vee (P \wedge Q) \equiv P \quad (2.40)$$

$$P \wedge (P \vee Q) \equiv P \quad (2.41)$$

Leyes de simplificación:

$$(P \wedge Q) \vee (\neg P \wedge Q) \equiv Q \quad (2.42)$$

$$(P \vee Q) \wedge (\neg P \vee Q) \equiv Q \quad (2.43)$$

Estamos obligados a demostrar estas nuevas leyes, ya que no aparecen en nuestro conjunto inicial de equivalencias. Lo haremos con cuidado y detalle en uno de los casos, dejando el otro como ejercicio.

Ejemplo 2.23. Absorción frente a \vee : $P \vee (P \wedge Q) \equiv P$.

Utilizaremos el método de tomar a uno de los equivalentes y derivar, a partir de él, al otro. Como el de la izquierda tiene más estructura, es el que tomamos como punto de partida.

Punto de partida. Localizaremos este esquema en alguno de los axiomas o teoremas que ya hayamos demostrado. En este momento únicamente contamos con (2.22) a (2.32).

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

<p>Usando Identidad (2.16) y Leibniz.</p> $\frac{P \equiv P \wedge \text{true}}{\boxed{P} \vee (P \wedge Q) \equiv \boxed{P \wedge \text{true}} \vee (P \wedge Q)}$ <p>Distributividad de \wedge (2.27)</p> $(P \wedge Q) \vee (P \wedge R) \equiv P \wedge (Q \vee R)$ <p>Usando sustitución $[Q, R := \text{true}, Q]$ tenemos:</p>	$\equiv (P \wedge \text{true}) \vee (P \wedge Q)$
<p>Usando dominación (2.19) y Leibniz:</p> $\frac{Q \vee \text{true} \equiv \text{true}}{P \wedge (Q \vee \text{true}) \equiv P \wedge \boxed{\text{true}}}$	$\equiv P \wedge (\text{true} \vee Q)$ $\equiv P \wedge \text{true}$
<p>Usando identidad de \wedge (2.16)</p>	$\equiv P$

Ejemplo 2.24. Simplificación: $(P \vee Q) \wedge (\neg P \vee Q) \equiv Q$.

Nuevamente tenemos que demostrar una equivalencia lógica, por lo que trataremos de transformar a uno de los equivalentes en el otro. Como el equivalente de la izquierda tiene mayor estructura, partiremos de él. Dado que el número que le corresponde a este teorema es el (2.42), podemos utilizar en este caso las leyes ((2.22) a (2.41)).

<p>Punto de partida. Localizaremos este esquema en alguno de los axiomas o teoremas que ya hayamos demostrado. Vemos un esquema similar en el rango derecho de (2.27):</p>	$(P \vee Q) \wedge (\neg P \vee Q)$
<p>Usando Conmutatividad (2.21).</p> $(P \vee Q) \wedge (\neg P \vee Q) \equiv (Q \vee P) \wedge (Q \vee \neg P)$	$\equiv (Q \vee P) \wedge (Q \vee \neg P)$

<p>Instanciando (2.27)</p> $(P \vee Q) \wedge (P \vee R) \equiv P \vee (Q \wedge R)$ <hr/> $(Q \vee P) \wedge (Q \vee \neg P) \equiv Q \vee (P \wedge \neg P)$	
<p>Contradicción: (2.24)</p> $P \wedge \neg P \equiv \text{false}$	$\equiv Q \vee (P \wedge \neg P)$
<p>Identidad: (2.15)</p> $Q \vee \text{false} \equiv Q$	$\equiv Q \vee \text{false}$ $\equiv Q$

Se deja como ejercicio la demostración de (2.41).

En todos los ejemplos de esta sección marcamos e hicimos explícitos todos los usos de las reglas. Sin embargo, en la práctica muchas de estas reglas se usan de manera implícita. A continuación damos algunos atajos que se pueden tomar al hacer álgebra de equivalencias lógicas.

1. La Ley de Conmutatividad se aplica directamente, “sin avisar”.
2. La Ley de Asociatividad se aplica directamente, “sin avisar”.
3. Se puede desechar directamente lo siguiente:
 - a) Copias duplicadas de una subexpresión en una expresión que es una disyunción o una conjunción (Ley de Idempotencia).
 - b) La constante true en una conjunción (Ley de Identidad para \wedge).
 - c) La constante false en una disyunción (Ley de Identidad para \vee).
4. De igual manera, se puede simplificar haciendo lo siguiente:
 - a) Sustituir el esquema $A \wedge \neg A$ por false (Ley de Contradicción).
 - b) Sustituir el esquema $A \vee \neg A$ por true (Ley del Tercero Excluido).
 - c) Sustituir el esquema $\neg \neg A$ por A (Ley de Doble Negación).

En esta sección hemos mostrado cómo es posible justificar formalmente el razonamiento ecuacional usual. Esta justificación, que se hizo apelando al uso de la regla de Leibniz, además de proporcionar un fundamento matemático formal a un razonamiento al que estamos acostumbrados desde hace mucho, nos da una pauta para una posible automatización del proceso.

En adelante el uso de razonamiento ecuacional será, por lo general, intuitivo, sin requerir el uso explícito de la regla de Leibniz.

Para terminar probaremos la equivalencia lógica entre una implicación y su contrapositiva, usando algunos de los atajos anteriores. Esto justifica el método de demostración por contrapositivo, usual en matemáticas.

Ejemplo 2.25. Contrapositiva: $P \rightarrow Q \equiv \neg Q \rightarrow \neg P$. Usaremos la ley de eliminación de la implicación mediante disyunción, así como las leyes de De Morgan.

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

Ejercicios

2.4.1.- Para las siguientes expresiones E , dadas z , X e Y , obtener $E[z := X]$ y $E[z := Y]$.

	z	E	X	Y
a)	p	p	$p \wedge q$	$q \wedge p$
b)	p	$(p \vee q) \wedge (p \vee r)$	true	$p \leftrightarrow p$
c)	p	$p \wedge p \leftrightarrow p$	$p \vee q$	$p \vee \neg q \leftrightarrow p$
d)	q	$p \wedge (\neg p \wedge q)$	$p \vee (q \wedge r)$	$(p \vee q) \wedge (p \vee r)$

2.4.2.- La regla de Leibniz se refiere a cualquier combinación de expresiones E , X e Y y a cualquier variable z . A continuación damos varios razonamientos que siguen el patrón de Leibniz y que están incompletos. El orden no es forzosamente el dado por la expresión, esto es, abajo de X no forzosamente está $E[z := X]$. Llene las partes que faltan y escriba en qué consiste la expresión E . Los últimos dos ejercicios tienen tres respuestas. Dar todas.

$$a) \frac{p \leftrightarrow p \vee 0}{p \vee 0 \vee q \leftrightarrow ?}$$

$$c) \frac{p \rightarrow q \leftrightarrow \neg q \rightarrow \neg p}{p \rightarrow q \rightarrow p \leftrightarrow ?}$$

$$b) \frac{7 = y + 1}{7 \cdot x + 7 \cdot y = ?}$$

$$d) \frac{x = y}{x + x = ?}$$

$$e) \frac{x = b + c}{x + y + w = ?}$$

$$f) \frac{b \cdot c = y + w}{x + y + w = ?}$$

2.4.3.- El objetivo de este ejercicio es reforzar las habilidades en el uso del argumento de Leibniz para demostrar que dos expresiones son iguales. Vamos a dar las expresiones $E[z := X]$ y $E[z := Y]$ y se deberán localizar respectivamente a X y a Y .

	$E[z := X]$	$E[z := Y]$
(a)	$(x + y) \cdot (x + y)$	$(x + y) \cdot (y + x)$
(b)	$(x + y) \cdot (x + y)$	$(y + x) \cdot (y + x)$
(c)	$x + y + w + x$	$x + y \cdot w + x$
(d)	$x \cdot y \cdot x$	$(y + w) \cdot y \cdot x$
(e)	$x \cdot y \cdot x$	$y \cdot x \cdot x$

2.4.4.- Elimine los operadores \rightarrow y \leftrightarrow de cada una de las siguientes proposiciones:

- a) $(P \rightarrow Q \wedge R) \vee ((R \leftrightarrow S) \wedge (Q \vee S))$
- b) $(P \rightarrow Q) \wedge (Q \rightarrow R)$
- c) $\neg P \rightarrow \neg Q$
- d) $(P \rightarrow Q) \leftrightarrow ((P \wedge Q) \leftrightarrow Q)$

2.4.5.- Demuestre la siguiente equivalencia lógica mediante razonamiento ecuacional.

$$(A \rightarrow B) \wedge (B \vee A) \equiv B$$

2.4.6.- Justifique la siguiente equivalencia lógica dando las instancias explícitas de la regla de Leibniz utilizadas.

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

2.4.7.- Considere el siguiente razonamiento ecuacional

$$\begin{aligned} \neg p \vee (q \rightarrow \neg s) &\equiv \neg p \vee (s \rightarrow \neg q) \\ &\equiv (s \rightarrow \neg q) \vee \neg p \\ &\equiv \neg(s \rightarrow \neg q) \rightarrow \neg p \end{aligned}$$

Justifique cada paso mediante instancias particulares de la regla de Leibniz, llenando la siguiente tabla:

<i>Paso</i>	<i>X</i>	<i>Y</i>	<i>E</i>	$E[z := X]$	$E[z := Y]$
1				$\neg p \vee (q \rightarrow \neg s)$	
2					
3					$\neg(s \rightarrow \neg q) \rightarrow \neg p$

2.4.8.- Complete utilizando la regla de Leibniz. Decida cuáles son las expresiones E, X, Y . Dé todas las respuestas posibles.

$$\frac{2 * y + 1 = 5}{x + (2 * y + 1) * w = ?}$$

2.5. Conceptos semánticos importantes

Una vez que hemos estudiado el análisis sintáctico de una fórmula lógica pasamos a estudiar ciertos conceptos de importancia relacionados con su semántica.

2.5.1. Interpretaciones

La noción de interpretación presentada en esta sección será de gran importancia para evitar el uso de tablas de verdad en las pruebas de correctud.

Definición 2.11 Un *estado* de las variables proposicionales es una función \mathcal{I} que asigna a cada variable proposicional el valor de falso o verdadero:

$$\mathcal{I} : VProp \rightarrow \{0, 1\}$$

donde $VProp$ es el conjunto de variables proposicionales.

Cada estado genera una función de interpretación sobre todas las fórmulas, definida como se explica a continuación:

Definición 2.12 Cada estado \mathcal{I} determina una *interpretación* de las fórmulas –denotada también por \mathcal{I} – definida como se muestra en la siguiente página:

$\mathcal{I}(\text{true})$	$= 1$	$\mathcal{I}(\text{false}) = 0$
$\mathcal{I}(\neg P)$	$= 1$	si y sólo si $\mathcal{I}(P) = 0$
$\mathcal{I}(P \vee Q)$	$= 0$	si y sólo si $\mathcal{I}(P) = 0 = \mathcal{I}(Q)$
$\mathcal{I}(P \wedge Q)$	$= 1$	si y sólo si $\mathcal{I}(P) = 1 = \mathcal{I}(Q)$
$\mathcal{I}(P \rightarrow Q)$	$= 0$	si y sólo si $\mathcal{I}(P) = 1$ e $\mathcal{I}(Q) = 0$
$\mathcal{I}(P \leftrightarrow Q)$	$= 1$	si y sólo si $\mathcal{I}(P) = \mathcal{I}(Q)$

Si $\mathcal{I}(P) = 1$ entonces decimos que

- \mathcal{I} *satisface a* P , o bien
- P *es satisfacible en* \mathcal{I} , o bien
- P *se satisface en* \mathcal{I} , o bien
- \mathcal{I} es un *modelo* de P .

Ejemplo 2.26. Si tenemos la fórmula $A = p \rightarrow q \vee r$, la siguiente asignación de estado

$$\mathcal{I}_1(p) = 1, \mathcal{I}_1(q) = 0, \mathcal{I}_1(r) = 0,$$

hace $\mathcal{I}_1(p \rightarrow q \vee r) = 0$, por lo que \mathcal{I}_1 no es un modelo para la fórmula. Por otro lado, el estado

$$\mathcal{I}_2(p) = 1, \mathcal{I}_2(q) = 0, \mathcal{I}_2(r) = 1$$

hace que $\mathcal{I}_2(p \rightarrow q \vee r) = 1$, por lo que sí es un modelo para la fórmula.

Dada una fórmula P podemos preguntarnos ¿cuántas interpretaciones hacen verdadera a P ? Las posibles respuestas llevan a las siguientes definiciones.

Definición 2.13 Sea P una fórmula. Entonces

- Si $\mathcal{I}(P) = 1$ para toda interpretación \mathcal{I} , decimos que P es una tautología o fórmula válida y escribimos $\models P$.
- Si $\mathcal{I}(P) = 1$ para alguna interpretación \mathcal{I} , decimos que P es satisfacible, que P es verdadera en \mathcal{I} o que \mathcal{I} es modelo de P y escribimos $\mathcal{I} \models P$.
- Si $\mathcal{I}(P) = 0$ para alguna interpretación \mathcal{I} , decimos que P es falsa o insatisfacible en \mathcal{I} o que \mathcal{I} no es modelo de P y escribimos $\mathcal{I} \not\models P$.
- Si $\mathcal{I}(P) = 0$ para toda interpretación \mathcal{I} , decimos que P es una contradicción o fórmula no satisfacible.

Similarmente, si Γ es un conjunto de fórmulas decimos que:

- Γ es satisfacible si tiene un modelo, es decir, si existe una interpretación \mathcal{I} tal que $\mathcal{I}(P) = 1$ para toda $P \in \Gamma$, lo cual denotamos a veces, abusando de la notación, con $\mathcal{I}(\Gamma) = 1$.
- Γ es insatisfacible o no satisfacible si no tiene un modelo, es decir, si no existe una interpretación \mathcal{I} tal que $\mathcal{I}(P) = 1$ para toda $P \in \Gamma$.

Para el último ejemplo se cumple lo siguiente, de acuerdo a la definición anterior,

$$\mathcal{I}_1 \not\models A, \mathcal{I}_2 \models A, \not\models \mathcal{A}.$$

Veamos otro ejemplo.

Ejemplo 2.27. Sean $\Gamma_1 = \{p \rightarrow q, r \rightarrow s, \neg s\}$, $\Gamma_2 = \{p \rightarrow q, \neg(q \vee s), s \vee p\}$. Entonces

- Si $\mathcal{I}(s) = \mathcal{I}(r) = \mathcal{I}(p) = 0$, entonces $\mathcal{I}(\Gamma_1) = 1$ por lo que Γ_1 es satisfacible.
- Γ_2 resulta insatisfacible, pues supóngase que existe una interpretación \mathcal{I} tal que $\mathcal{I}(\Gamma_2) = 1$. Entonces se tiene que $\mathcal{I}(\neg(q \vee s)) = 1$, por lo que $\mathcal{I}(\neg q) = \mathcal{I}(\neg s) = 1$. Además, como $\mathcal{I}(p \rightarrow q) = 1$ entonces $\mathcal{I}(p) = 0$, puesto que el consecuente de la implicación es falso. De esto último se tiene $\mathcal{I}(s) = 1$, dado que $\mathcal{I}(s \vee p) = 1$. De manera que se tiene $\mathcal{I}(\neg s) = 1 = \mathcal{I}(s)$, lo cual es imposible. Por lo tanto, no puede existir una interpretación \mathcal{I} que satisfaga a Γ_2 .

Con respecto a las tablas de verdad tenemos las siguientes observaciones:

- Una fórmula P es satisfacible si en alguna línea de la tabla de verdad, P toma el valor 1. En caso contrario, es decir si en **todas** las líneas toma el valor 0, entonces es insatisfacible (contradicción).
- Un conjunto de fórmulas Γ es satisfacible si existe alguna línea de la tabla de verdad en la que **todas** las fórmulas de Γ toman el valor 1.

2.5.2. Consecuencia lógica

La definición matemática formal de argumento deductivo correcto se sirve del concepto de consecuencia o implicación lógica que discutimos en esta sección.

Definición 2.14 Sean $\Gamma = \{A_1, \dots, A_n\}$ un conjunto de fórmulas y B una fórmula. Decimos que B es *consecuencia lógica* de Γ si toda interpretación \mathcal{I} que satisface a Γ también satisface a B . Es decir, si todo modelo de Γ es modelo de B . En tal caso escribimos $\Gamma \models B$.

Nótese que la relación de consecuencia lógica está dada por una implicación de la forma si $\mathcal{I}(\Gamma) = 1$ entonces $\mathcal{I}(B) = 1$.

De manera que no se afirma nada acerca de la satisfacibilidad del conjunto Γ , sino que simplemente se supone que es satisfacible y, en tal caso, se prueba que la fórmula B también lo es con la misma interpretación.

Obsérvese la sobrecarga del símbolo \models que previamente utilizamos para denotar satisfacibilidad ($\mathcal{I} \models A$) y tautologías ($\models A$).

Ejemplo 2.28. Considérese el siguiente conjunto $\Gamma = \{q \rightarrow p, p \leftrightarrow t, t \rightarrow s, s \rightarrow r\}$. Muestre que $\Gamma \models q \rightarrow r$.

Sea \mathcal{I} un modelo de Γ . Tenemos que demostrar que $\mathcal{I}(q \rightarrow r) = 1$. Si $\mathcal{I}(q) = 0$ entonces $\mathcal{I}(q \rightarrow r) = 1$ y terminamos. En otro caso se tiene $\mathcal{I}(q) = 1$ de donde $\mathcal{I}(p) = 1$ pues $\mathcal{I}(q \rightarrow p) = 1$. Entonces se tiene $\mathcal{I}(t) = 1$, pues \mathcal{I} es modelo de $p \leftrightarrow t$, de donde $\mathcal{I}(s) = 1$ dado que \mathcal{I} también es modelo de $t \rightarrow s$. Finalmente, como $\mathcal{I}(s \rightarrow r) = 1$ e $\mathcal{I}(s) = 1$, entonces $\mathcal{I}(r) = 1$. Por lo tanto, $\mathcal{I}(q \rightarrow r) = 1$.

Para terminar la sección discutimos algunas propiedades importantes de la relación de consecuencia lógica.

Proposición 2.2 La relación de consecuencia lógica cumple las siguientes propiedades:

- (a) Si $A \in \Gamma$ entonces $\Gamma \models A$.
- (b) Principio de refutación: $\Gamma \models A$ si y sólo si $\Gamma \cup \{\neg A\}$ es insatisfacible.
- (c) $\Gamma \models A \rightarrow B$ si y sólo si $\Gamma \cup \{A\} \models B$.
- (d) Insatisfacibilidad implica trivialidad: Si Γ es insatisfacible entonces $\Gamma \models A$ para toda fórmula A .
- (e) Si $\Gamma \models \text{false}$ entonces Γ es insatisfacible.
- (f) $A \equiv B$ si y sólo si $A \models B$ y $B \models A$.
- (g) $\models A$ (es decir A es tautología) si y sólo si $\emptyset \models A$ (es decir, A es consecuencia lógica del conjunto vacío).

Demostración.

Procedemos a justificar algunos de los incisos:

- (a) Si $\mathcal{I}(\Gamma) = 1$ quiere decir que existe un modelo para Γ y, por lo tanto, para cada una de las fórmulas de Γ , en particular para A .
- (b) Supongamos que toda interpretación que satisface a Γ también satisface a A (definición de $\Gamma \models A$). Tenemos que demostrar que $\Gamma \cup \{\neg A\}$ es insatisfacible. Si una interpretación satisface a Γ , dado que satisfacía también a A , entonces no satisface a $\neg A$. Por lo tanto, es imposible satisfacer a Γ y a $\neg A$ al mismo tiempo, lo cual implica que $\Gamma \cup \{\neg A\}$ es insatisfacible.

En sentido contrario, supongamos que $\Gamma \cup \{\neg A\}$ es insatisfacible. Para mostrar que $\Gamma \models A$, consideremos \mathcal{I} , una interpretación cualquiera tal que $\mathcal{I}(\Gamma) = 1$. En tal caso, necesariamente tenemos que $\mathcal{I}(A) = 1$. De lo contrario, si $\mathcal{I}(A) = 0$ entonces $\mathcal{I}(\neg A) = 1$, con lo que $\Gamma \cup \{\neg A\}$ sería satisfacible mediante \mathcal{I} , lo cual, por hipótesis, no sucede.

- (c) Supongamos $\Gamma \models A \rightarrow B$ y tenemos que demostrar, entonces, que $\Gamma \cup \{A\} \models B$. Por la definición de consecuencia lógica, tenemos que si $\mathcal{I}(\Gamma) = 1$ entonces $\mathcal{I}(A \rightarrow B) = 1$. Para mostrar que $\Gamma \cup \{A\} \models B$, sea \mathcal{I} una interpretación tal que $\mathcal{I}(\Gamma \cup \{A\}) = 1$; en esta interpretación se tiene que $\mathcal{I}(A) = 1$; como además $\mathcal{I}(A \rightarrow B) = 1$ por hipótesis, porque estamos suponiendo $\mathcal{I}(\Gamma) = 1$, entonces, por

definición de la interpretación de una implicación y dado que para el antecedente A se tiene $\mathcal{I}(A) = 1$, necesariamente $\mathcal{I}(B) = 1$. Por lo tanto $\Gamma \cup \{A\} \models B$.

En sentido contrario, supongamos que $\Gamma \cup \{A\} \models B$. Esto es que si $\mathcal{I}(\Gamma \cup \{A\}) = 1$ entonces $\mathcal{I}(B) = 1$. Sea \mathcal{I} una interpretación tal que $\mathcal{I}(\Gamma) = 1$. Tenemos los siguientes casos:

- $\mathcal{I}(A) = 1$. Entonces $\mathcal{I}(B) = 1$, pues se cumple que $\mathcal{I}(\Gamma \cup \{A\}) = 1$; además, estamos suponiendo que $\Gamma \cup \{A\} \models B$, con lo que $\mathcal{I}(A \rightarrow B) = 1$. Por lo tanto, $\Gamma \models A \rightarrow B$.
- $\mathcal{I}(A) = 0$. En este caso, independientemente de cuál sea el valor de $\mathcal{I}(B)$, tenemos $\mathcal{I}(A \rightarrow B) = 1$, por lo que nuevamente $\Gamma \models A \rightarrow B$.

Hemos demostrado, entonces, $\Gamma \models A \rightarrow B$ si y sólo si $\Gamma \cup \{A\} \models B$.

- (d) Si Γ es insatisfacible quiere decir que para toda interpretación \mathcal{I} , se tiene $\mathcal{I}(\Gamma) = 0$. Si esto es así, se cumple trivialmente que si $\mathcal{I}(\Gamma) = 1$ entonces $\mathcal{I}(A) = 1$. Es decir $\Gamma \models A$.
- (e) Si $\Gamma \models \text{false}$, por la definición de consecuencia lógica tenemos que $\mathcal{I}(\Gamma) = 1$ implica $\mathcal{I}(\text{false}) = 1$. Sin embargo, $\mathcal{I}(\text{false}) = 0$ siempre sucede; por lo que, como $\Gamma \models \text{false}$ tenemos necesariamente que $\mathcal{I}(\Gamma) = 0$ para toda posible interpretación de Γ , es decir, Γ es insatisfacible.

Se deja la justificación de los incisos restantes al lector. □

Es importante disponer de métodos algorítmicos para decidir la consecuencia lógica, que nos permitirán, en particular, analizar argumentos del lenguaje natural y establecer su correctud formalmente. En las siguientes secciones presentaremos algunos de estos métodos.

Ejercicios

2.5.1.- Para cada una de las fórmulas que siguen, determine si son o no *satisfacibles*. Si lo son, muestre un *modelo* para cada una de ellas.

a) $p \wedge q \leftrightarrow \neg p \wedge q$

c) $p \wedge q \wedge \neg p$

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

d) $(p \rightarrow q) \wedge (q \rightarrow p)$

2.5.2.- Use interpretaciones para determinar si las siguientes fórmulas son tautologías, contradicciones o contingentes. Si son contingentes, dé una interpretación en la que la fórmula no se evalúa a verdadero.

a) $\left(((p \vee q) \vee r) \wedge (p \vee (q \vee r)) \right) \rightarrow p \vee q$

c) $p \vee q \rightarrow p \vee r$

b) $(p \wedge (q \wedge r)) \rightarrow (p \rightarrow (q \rightarrow r))$

d) $(p \rightarrow (p \rightarrow q)) \rightarrow p$

2.5.3.- Decida si los siguientes conjuntos son satisfacibles.

- a) $\Gamma = \{(\neg q \wedge r) \vee p \vee q, p \wedge r\}$
- b) $\Gamma = \{p \wedge \neg q, \neg(q \vee \neg p), (q \wedge p) \vee q \vee \neg p\}$
- c) $\Gamma = \{q \vee r \vee s, \neg(q \vee r), \neg(r \vee s), \neg(s \vee q)\}$
- d) $\Gamma = \{\neg(p \wedge q) \wedge \neg(p \wedge r), q \vee r, \neg(p \vee \neg r)\}$
- e) $\Gamma = \{p \leftrightarrow q, q \leftrightarrow s, p, \neg s\}$

2.5.4.- Demuestre la consecuencia lógica en cada caso:

- a) $\{p, q\} \models p \wedge q$
- b) $\{p, \neg q\} \models \neg(p \rightarrow q)$
- c) $\{r \wedge s \rightarrow t, \neg t\} \models t \rightarrow q$
- d) $\{\neg q \rightarrow \neg r, \neg r \rightarrow \neg p, \neg p \rightarrow \neg q\} \models q \leftrightarrow r$
- e) $\{p \vee q, p \rightarrow r, q \rightarrow r\} \models r$

2.5.5.- Determine, utilizando interpretaciones, si los siguientes conjuntos de fórmulas de la lógica proposicional son satisfacibles; en caso positivo exhiba un modelo.

- a) $\Gamma = \{p \rightarrow q, \neg q \vee r, p \wedge \neg r\}$
- b) $\Gamma = \{(p \vee q) \rightarrow r, \neg((\neg p \wedge \neg q) \vee r)\}$
- c) $\Gamma = \{(p \vee q) \rightarrow r, \neg r, \neg p\}$

2.6. Análisis de argumentos

En esta sección aplicamos todos los conocimientos previos de lógica matemática estudiados hasta ahora para cumplir con nuestro propósito fundamental: el análisis de correctud de un argumento lógico proposicional.

2.6.1. Tablas de verdad

Como ya discutimos antes, un argumento es correcto si y sólo si su fórmula asociada es una tautología; para decidir esta situación podemos construir la tabla de verdad correspondiente tal y como lo hicimos en la sección 2.1.6. Veamos un ejemplo más.

Ejemplo 2.29. El argumento $P \rightarrow Q, Q \rightarrow R / \therefore P \rightarrow R$ es correcto.

Basta ver que $\models (P \rightarrow Q) \wedge (Q \rightarrow R) \rightarrow (P \rightarrow R)$. La tabla de verdad se muestra en la tabla 2.9 en la siguiente página.

Como se observa de los valores en la quinta columna, en negritas, la fórmula es una tautología, por lo que este argumento, conocido como *silogismo hipotético*, es correcto.

Este ejemplo, junto con los de la sección 2.1.6, deja ver que la tabla de verdad se vuelve más complicada al aumentar el número de variables proposicionales involucradas. La construcción de la tabla de verdad completa, aunque plausible desde el punto de vista

teórico, es de “fuerza bruta” en la práctica, pues nos obliga, en los casos interesantes y no triviales, a evaluar un número muy grande de estados para determinar si tenemos o no una tautología (o una contradicción). Aun si lo hiciésemos con una computadora, y suponiendo que a la computadora le llevara un milisegundo evaluar cada estado, si la expresión es muy grande tenemos el crecimiento en el número de estados que vemos en la tabla 2.10.

Tabla 2.9. $P \rightarrow Q, Q \rightarrow R / \therefore P \rightarrow R$

P	Q	R	$(P \rightarrow Q)$	\wedge	$(Q \rightarrow R)$	\rightarrow	$(P \rightarrow R)$
1	1	1	1	1	1	1	1
1	1	0	1	0	0	1	0
1	0	1	0	0	1	1	1
1	0	0	0	0	1	1	0
0	1	1	1	1	1	1	1
0	1	0	1	0	0	1	1
0	0	1	1	1	1	1	1
0	0	0	1	1	1	1	1

Como se puede observar en la tabla 2.10, cada vez que se agrega una variable a la expresión, el tiempo que lleva calcular todos sus estados se duplica, siguiendo, como ya mencionamos, a la función 2^n , donde n es el número de variables⁸.

Tabla 2.10. Crecimiento en el número de estados con respecto a número de variables

Número de variables	Número de estados	Tiempo (segundos)
1	2	.002
2	4	.004
3	8	.008
⋮	⋮	⋮
10	1,024	1
11	2,048	2
⋮	⋮	⋮
20	1,048,576	1,048(= 17min)
⋮	⋮	⋮

⁸Cuando tenemos este tipo de cálculo, decimos que la función crece con 2^n , o que tiene un *crecimiento exponencial*. Este tipo de cálculos, en la práctica, no pueden ser evaluados en una computadora cuando la n no es pequeña.

Esta ineficiencia surge, por ejemplo, en problemas de calendarización o búsqueda de rutas, donde ciertas fórmulas lógicas involucradas tienen usualmente cientos de variables. Para estimar la ineficiencia considérese una fórmula con 500 variables, cuya tabla de verdad tendrá 2^{500} renglones, número aproximadamente igual a 10^{150} , los cuales, de acuerdo a nuestra suposición anterior respecto a la velocidad de la computadora, se calcularían en 10^{147} milisegundos. Dado que en un año hay 3.1536×10^{10} milisegundos, la tabla terminaría de calcularse en aproximadamente 3.2×10^{139} años; considerando que la edad de nuestro planeta es aproximadamente 10^9 años, podemos corroborar que el tiempo estimado del método es inadmisibile.

Dada esta situación, vamos a utilizar tablas de verdad únicamente para verificar expresiones pequeñas y cuando no podamos recurrir a otras técnicas.

Obsérvese que el método de tablas de verdad puede evitarse al usar esquemas: una vez que se prueba que un argumento es correcto, él mismo genera un esquema, llamado *regla de inferencia*, en el que cada instancia de esta regla será, a su vez, un argumento correcto.

Ejemplo 2.30. Mostrar la correctud del argumento

$$\begin{array}{l} r \rightarrow s \vee \neg t \\ \hline (r \rightarrow s \vee \neg t) \rightarrow \neg p \wedge (q \vee w) \\ \hline \therefore \neg p \wedge (q \vee w) \end{array}$$

La tabla de verdad para este análisis tendría $2^6 = 64$ renglones, dado que tenemos seis variables. Sin embargo, no es necesario el análisis puesto que el argumento corresponde al esquema del modus ponens que ya mostramos que es correcto. Formalmente tenemos que

$$\begin{aligned} (P \wedge (P \rightarrow Q) \rightarrow Q) [P, Q := r \rightarrow s \vee \neg t, \neg p \wedge (q \vee w)] &= \\ = ((r \rightarrow s \vee \neg t) \wedge ((r \rightarrow s \vee \neg t) \rightarrow (\neg p \wedge (q \vee w))) &\rightarrow (\neg p \wedge (q \vee w))) \end{aligned}$$

y como $\models P \wedge (P \rightarrow Q) \rightarrow Q$ podemos concluir que

$$\models ((r \rightarrow s \vee \neg t) \wedge ((r \rightarrow s \vee \neg t) \rightarrow (\neg p \wedge (q \vee w))) \rightarrow (\neg p \wedge (q \vee w))).$$

Este método es útil en algunos casos en los que ya se conoce de antemano un esquema de argumento correcto; sin embargo no es siempre efectivo ni fácil de implementar.

2.6.2. Uso de interpretaciones

Ya estamos convencidos de que el uso de una tabla de verdad para analizar la correctud de un argumento es, en general, una muy mala idea.

Construir la tabla de verdad para una fórmula de la forma $A_1 \wedge \dots \wedge A_n \rightarrow B$, en su totalidad, resulta, en la mayoría de los casos, innecesario. Por ejemplo, al observar nuevamente la tabla 2.9, podemos darnos cuenta que sólo nos interesa la mitad de ésta, a saber

los renglones donde la conjunción de las premisas es verdadera. El resto de la tabla puede desecharse, puesto que si la conjunción de las premisas no es verdadera, la implicación será verdadera automáticamente. El concepto de consecuencia lógica toma en cuenta esta observación, al suponer que las premisas son ciertas y bajo este supuesto mostrar que, bajo la misma interpretación, la conclusión también lo es.

Para mostrar la correctud del argumento lógico $A_1, \dots, A_n / \therefore B$ mediante el uso de interpretaciones, nos servimos de la siguiente proposición, cuya demostración dejamos como ejercicio.

Proposición 2.3 El argumento $A_1, \dots, A_n / \therefore B$ es lógicamente correcto si y sólo si

$$\{A_1, \dots, A_n\} \models B,$$

es decir, si la conclusión es consecuencia lógica de las premisas.

De acuerdo a las propiedades de la consecuencia lógica, existen básicamente dos formas para demostrar la correctud de un argumento, el método directo y el indirecto.

Método directo: Probar la consecuencia $A_1, \dots, A_n \models B$. Para esto se supone la existencia de una interpretación \mathcal{I} que sea modelo de todas las premisas y se argumenta, usando esta información y la definición de interpretación, que la conclusión B también se satisface con \mathcal{I} .

Método indirecto (refutación o contradicción): Probar que es insatisfacible el conjunto $\{A_1, \dots, A_n, \neg B\}$. Para esto se supone que hay una interpretación \mathcal{I} que hace verdaderas a todas las premisas y a la negación de la conclusión $\neg B$ o bien, equivalentemente, hace falsa a la conclusión B . Apelando a este supuesto y a la definición de interpretación, se trata de mostrar que tal interpretación no puede existir; esto se logra mostrando que cierta fórmula está forzada a ser verdadera y falsa al mismo tiempo.

Es de importancia observar que estos métodos son la base de los métodos usuales de demostración en matemáticas. En un curso cualquiera de matemáticas, cuando se dice que la demostración de un teorema de la forma $A \rightarrow B$ es directa es porque estamos probando la consecuencia $A \models B$ con el método directo. Similarmente si hablamos de una demostración indirecta o por contradicción o reducción al absurdo, es porque estamos probando $A \models B$ con el método indirecto.

Veamos algunos ejemplos.

Ejemplo 2.31. Mostrar la correctud del argumento $\{p, s \vee \neg s, \neg p \vee q, \neg q \leftrightarrow r\} / \therefore \neg r$.

Sean $\Gamma = \{p, s \vee \neg s, \neg p \vee q, \neg q \leftrightarrow r\}$; debemos mostrar que $\Gamma \models \neg r$, para lo cual tomamos una interpretación \mathcal{I} tal que \mathcal{I} es modelo de Γ . Debemos mostrar que $\mathcal{I}(\neg r) = 1$.

Como \mathcal{I} es modelo de Γ entonces $\mathcal{I}(p) = 1$ e $\mathcal{I}(\neg p \vee q) = 1$, de donde $\mathcal{I}(q) = 1$ puesto que $\mathcal{I}(\neg p) = 0$. Como $\mathcal{I}(q) = 1$ e $\mathcal{I}(\neg q \leftrightarrow r) = 1$ entonces $\mathcal{I}(r) = 0$, de donde finalmente se obtiene $\mathcal{I}(\neg r) = 1$. Obsérvese que la prueba no determina un valor para s ya que con esta

interpretación el argumento es correcto independientemente del valor de s . En particular, la única fórmula que involucra a s es la tautología $s \vee \neg s$.

Este método puede resultar tedioso o intrincado pero puede escribirse de manera más clara enunciando cada paso de razonamiento, como en el siguiente ejemplo.

Ejemplo 2.32. Mostrar la correctud del argumento $p \rightarrow q, \neg q / \therefore \neg p$, conocido como *Modus Tollens*, al que se hace referencia más adelante.

Para lograr esto mostramos la consecuencia lógica $p \rightarrow q, \neg q \models \neg p$.

- 1. $\mathcal{I}(p \rightarrow q) = 1$ Hipótesis
- 2. $\mathcal{I}(\neg q) = 1$ Hipótesis
- 3. $\mathcal{I}(q) = 0$ por 2, ya que $\mathcal{I}(\neg q) = 1$
- 4. $\mathcal{I}(p) = 0$ por 1 y 3, ya que si $\mathcal{I}(p \rightarrow q) = 1$ e $\mathcal{I}(q) = 0$,
 $\therefore \mathcal{I}(p)$ no puede ser 1.

De manera que el argumento es correcto. El razonamiento paso a paso permite una mayor claridad en el proceso de análisis. Por supuesto que cada paso debe tener una justificación exacta. El análisis terminó aquí al llegar a que la conclusión es verdadera, por lo que se probó la consecuencia lógica de manera directa.

Ejemplo 2.33. Si hoy tirila y Chubaka es kismi entonces Chubaka es borogrove y si hoy no tirila entonces hay fefos. Más aún sabemos que no hay fefos y que Chubaka es kismi, luego entonces Chubaka es borogrove.

La formalización es:

Variable Proposicional	Enunciado
t	hoy tirila
k	Chubaka es kismi
b	Chubaka es borogrove
f	hay fefos

y el argumento queda como sigue:

$t \wedge k \rightarrow b$	Si hoy tirila y Chubaka es kismi entonces Chubaka es borogrove
$\neg t \rightarrow f$	si hoy no tirila entonces hay fefos
$\neg f \wedge k$	sabemos que no hay fefos y que Chubaka es kismi
$\therefore b$	de donde Chubaka es borogrove

Queremos demostrar que $\{t \wedge k \rightarrow b, \neg t \rightarrow f, \neg f \wedge k\} \models b$.

1. $\mathcal{I}(t \wedge k \rightarrow b) = 1$ Hipótesis.
2. $\mathcal{I}(\neg t \rightarrow f) = 1$ Hipótesis.
3. $\mathcal{I}(\neg f \wedge k) = 1$ Hipótesis.
4. $\mathcal{I}(b) = 0$ Refutación.
5. $\mathcal{I}(k) = 1$ por 3, $\mathcal{I}(p \wedge q) = 1$ si y sólo si $\mathcal{I}(p) = 1$ e $\mathcal{I}(q) = 1$
6. $\mathcal{I}(t \wedge k) = 0$ por 4 y 1. Como $\mathcal{I}(b) = 0$ y la implicación en 1 es verdadera, entonces la única posibilidad para $t \wedge k$ es que valga 0.
7. $\mathcal{I}(t) = 0$ por 5 y 6. Por 5, $\mathcal{I}(k) = 1$; si $\mathcal{I}(t \wedge k) = 0$ (por 6) es porque $\mathcal{I}(t) = 0$
8. $\mathcal{I}(\neg t) = 1$ por 7.
9. $\mathcal{I}(f) = 1$ por 2 y 8. Como el antecedente es verdadero en (2), para que la implicación sea verdadera el consecuente tiene que serlo.
10. $\mathcal{I}(\neg f) = 1$ por 3, Tenemos que $\mathcal{I}(\neg f \wedge k) = 1$ y esta interpretación exige $\mathcal{I}(\neg f) = 1$ e $\mathcal{I}(k) = 1$.
11. $\mathcal{I}(f) = 0$ por 10, lo que nos lleva a una contradicción con 9.

Los pasos 9 y 11 generan una contradicción explícita, de manera que, por el principio de refutación, el conjunto $\Gamma \cup \{\neg b\}$ es insatisfacible y el argumento es correcto.

Ejemplo 2.34. Mostrar la correctud del siguiente argumento conocido como *dilema constructivo simple*: $p \rightarrow r, \neg p \rightarrow r / \therefore r$.

1. $\mathcal{I}(p \rightarrow r) = 1$ Hipótesis
2. $\mathcal{I}(\neg p \rightarrow r) = 1$ Hipótesis
3. $\mathcal{I}(r) = 0$ Refutación
4. $\mathcal{I}(p) = 0$ por 3 y 1. Como $\mathcal{I}(p \rightarrow r) = 1$ e $\mathcal{I}(r) = 0$, $\mathcal{I}(p)$ tiene que ser 0.
5. $\mathcal{I}(\neg p) = 0$ por 3 y 2, argumento similar a 4
6. $\mathcal{I}(p) = 1$ por 5, pero hay contradicción con 4

Por lo tanto el argumento es correcto.

Es importante observar lo siguiente acerca del uso del método de interpretaciones para analizar argumentos:

- Si se usa el método directo, el análisis termina una vez que se logra asignar a la conclusión el valor de verdadero.
- Si se usa el método indirecto, el análisis termina una vez que se logre forzar a que una fórmula tome los dos valores posibles de verdad. Esta fórmula es generalmente una variable proposicional, aunque esto no es la única opción.
- Forzar un valor v para una fórmula A significa que, de acuerdo a la definición de interpretación y a los valores previamente obtenidos de variables o fórmulas, el valor para A es *necesariamente y sin lugar a dudas* el valor v , que puede ser 1 o 0. Por ejemplo, si sabemos que $\mathcal{I}(p \rightarrow q) = 1$ e $\mathcal{I}(q) = 0$, entonces necesariamente $\mathcal{I}(p) = 0$, puesto que si tuviésemos $\mathcal{I}(p) = 1$, la definición de interpretación para la implicación nos llevaría a $\mathcal{I}(p \rightarrow q) = 0$, lo cual sabemos que no sucede. De esta manera, el valor de p *está forzado* a ser 0.

Es error común asignar valores que no están forzados; por ejemplo, si sólo sabemos que $\mathcal{I}(r \rightarrow s) = 1$, entonces es un error decir que el valor $\mathcal{I}(s) = 0$ está forzado puesto que no hay suficiente información para descartar la posibilidad de que $\mathcal{I}(r) = 0$, en cuyo caso s podría ser verdadero sin afectar el valor conocido de $r \rightarrow s$.

- Si al usar el método indirecto no es posible hallar una contradicción o si en el método directo no se forzó a que la conclusión sea verdadera, entonces el argumento resulta incorrecto y la interpretación asignada será un *contraejemplo* a la correctud del argumento, puesto que las premisas serán ciertas y la conclusión falsa.

Analizaremos ahora un par de argumentos incorrectos.

Ejemplo 2.35. Analizar el argumento $q \rightarrow p, r \vee s / \therefore r \rightarrow p$.

Procedemos directamente:

1. $\mathcal{I}(q \rightarrow p) = 1$ Hipótesis
2. $\mathcal{I}(r \vee s) = 1$ Hipótesis

En este momento no hay manera de forzar ningún valor puesto que tanto la implicación como la disyunción son verdaderas en tres estados. Esta libertad nos permite asignar valores que causen que la conclusión sea falsa, lo que sucede como sigue:

3. $\mathcal{I}(r) = 1$ Supuesto
4. $\mathcal{I}(p) = 0$ Supuesto

Aún no terminamos, puesto que debemos dar valores a q y s , los cuales pueden obtenerse como sigue:

5. $\mathcal{I}(q) = 0$ por 1 y 4
6. $\mathcal{I}(s) = 0$ Supuesto

De manera que la interpretación dada por $\mathcal{I}(p) = \mathcal{I}(q) = \mathcal{I}(s) = 0$ e $\mathcal{I}(r) = 1$ es un contraejemplo al argumento, pues con esta interpretación tenemos $\mathcal{I}(r \rightarrow p) = 0$, ya que $1 \rightarrow 0$ es 0. Esto es, en el estado $\{p = 0, q = 0, s = 0, r = 1\}$, tenemos que

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

se evalúa a 0.

				(2)	(3)	(1)	(5)	(4)
p	q	r	s	$q \rightarrow p$	\wedge	$r \vee s$	\rightarrow	$r \rightarrow p$
0	0	1	0	1	1	1	0	0

Obsérvese que s también pudo haber sido verdadero, lo cual habría generado otro contraejemplo.

El método indirecto puede ser de más ayuda en algunos casos, pues obliga, desde el principio, a forzar algunos valores, como en el siguiente ejemplo.

Ejemplo 2.36. Analizar el argumento $q \rightarrow p, r \rightarrow p / \therefore r \vee s$.

Procedemos indirectamente:

1. $\mathcal{I}(q \rightarrow p) = 1$ Hipótesis
2. $\mathcal{I}(r \rightarrow p) = 1$ Hipótesis
3. $\mathcal{I}(r \vee s) = 0$ Refutación
4. $\mathcal{I}(r) = 0$ por 3
5. $\mathcal{I}(s) = 0$ por 3

Obsérvese que falta asignar los valores de p y q . Puede ser que con la asignación $\mathcal{I}(r) = 0$ ya aseguramos que la segunda premisa se mantiene cierta, por lo que el valor de p está libre. Asimismo, el valor de q sólo afecta a la primera premisa y puede elegirse libremente. Un contraejemplo es entonces $\mathcal{I}(r) = \mathcal{I}(s) = \mathcal{I}(q) = \mathcal{I}(p) = 0$. Con estos valores aseguramos que las premisas son verdaderas pero que la conclusión es falsa, por lo que el argumento no es correcto. Otro contraejemplo es $\mathcal{I}(r) = \mathcal{I}(s) = \mathcal{I}(q) = 0, \mathcal{I}(p) = 1$, como se puede verificar de manera muy sencilla.

Algunas observaciones son pertinentes.

- Al usar valores supuestos –no forzados– no es posible afirmar la correctud del argumento al llegar al valor verdadero para la conclusión o al llegar a una contradicción. En este caso esto sólo indica que el valor supuesto debe reconsiderarse. Si se llega al mismo resultado para todos los posibles valores supuestos entonces podremos afirmar la correctud del argumento, pero sólo hasta ese momento.
- En el caso de llegar a un contraejemplo con un valor supuesto, con éste basta. No es necesario reconsiderar valores supuestos pues el contraejemplo ya está construido.

El método de interpretaciones, si bien es más eficiente en general que el uso de tablas de verdad, requiere de una gran interacción con el usuario, por lo que se antoja difícil de automatizar; es un método muy cercano al razonamiento humano. Más aún, los pasos de razonamiento no siempre son únicos, por ejemplo al usar supuestos, lo cual añade una dificultad más, la elección o no determinismo.

La noción de consecuencia lógica es un concepto semántico de gran importancia que permite analizar argumentos lógicos y, además, puede generalizarse a otros sistemas lógicos, en contraste con las tablas de verdad. Más aún, el uso de interpretaciones proporciona la base para la búsqueda de contraejemplos a argumentos incorrectos. Sin embargo, no es un método eficiente para encontrar consecuencias dado un conjunto de premisas. Para este propósito es más conveniente construir pruebas o derivaciones de manera sintáctica, es decir, sin apelar al concepto de interpretaciones. Haremos esto en la siguiente sección.

2.6.3. Derivaciones

Muchos argumentos lógicos correctos pueden obtenerse mediante composición de otros argumentos, cuya correctud ya fue verificada previamente, en el sentido de que la conclusión de un argumento particular puede servir como una de las premisas para un siguiente argumento, y así sucesivamente, hasta llegar a una conclusión deseada. Obsérvese que esta composición de argumentos es un mecanismo puramente sintáctico, al no apelar directamente a la noción de verdad o interpretación. Veamos un par de ejemplos.

Ejemplo 2.37. Queremos demostrar que el siguiente fragmento de programa deja el valor de la variable x de tal forma que después de la ejecución es imposible que $x > \text{Max}$, esto es $(x > \text{Max}) \equiv \text{false}$.

if $x > \text{Max}$ **then** $x := \text{Max}$;

Formalizamos con las siguientes variables proposicionales:

p	:	$x > \text{Max}$	antes de la ejecución
q	:	$x = \text{Max}$	después de la ejecución
r	:	$x > \text{Max}$	después de la ejecución

Tenemos que distinguir entre $x > \text{Max}$ antes y después de la ejecución, pues la asignación modifica el valor de la variable x , es decir, x tiene un valor distinto antes y después de la ejecución del programa.

Vamos a hacer primero un análisis intuitivo del problema: hay dos casos, correspondientes a p y $\neg p$. Si p sucede entonces la asignación se lleva a cabo y q se vuelve válida, es decir la implicación $p \rightarrow q$ se cumple. Además, si q es válida entonces $\neg r$ también, pues si los dos números x y Max son iguales entonces $x > \text{Max}$ es falso, así que la implicación $q \rightarrow \neg r$ es válida. Por otro lado, si $\neg p$ es válida, entonces la asignación no se lleva a cabo

y claramente $\neg r$ es cierta, pues en este caso p es equivalente a r , por lo que la implicación $\neg p \rightarrow \neg r$ es válida. Formalmente queremos concluir $\neg r$, lo cual es posible usando como hipótesis las implicaciones anteriores y aplicando los esquemas de silogismo hipotético (SH) y dilema constructivo simple (DCS), (ver ejemplos 2.5 y 2.34). Procedemos paso a paso como sigue:

Fórmula	Justificación	Comentario
1. $p \rightarrow q$	Hipótesis	Si $x > \text{Max}$ antes de la ejecución entonces $x = \text{Max}$ después de la ejecución
2. $q \rightarrow \neg r$	Hipótesis	Si $x = \text{Max}$ después de la ejecución entonces $x > \text{Max}$ no es cierta después de la ejecución.
3. $\neg p \rightarrow \neg r$	Hipótesis	Si $x > \text{Max}$ no es cierta antes de la ejecución entonces tampoco después de la ejecución
4. $p \rightarrow \neg r$	SH 1,2	Si $x > \text{Max}$ antes de la ejecución entonces $x > \text{Max}$ no es cierta después de la ejecución.
5. $\neg r$	DCS 3,4	Por lo tanto, sin importar si $x > \text{Max}$ es cierta o falsa antes de la ejecución, después de la ejecución $x > \text{Max}$ es falsa.

Se observa que el paso 4, que es la conclusión de una instancia del silogismo hipotético, fue usado además como premisa para lograr una instancia del dilema constructivo simple. Más aún, en ningún momento se apela a la noción de interpretación.

Ejemplo 2.38. Uno de los más reconocidos pensadores “lógicos” es Sherlock Holmes, el detective creado por Arthur Conan Doyle. Veamos una de sus argumentaciones más famosas, que aparece en el libro “Estudio en Escarlata”:

Y ahora llegamos a la gran pregunta del motivo. El robo no fue la razón del asesinato, ya que nada fue sustraído. Entonces, ¿fue la política o fue una mujer? Esta es la pregunta a la que me enfrenté. Me incliné desde un principio a la segunda suposición. Los asesinos políticos hacen su trabajo lo más rápido posible y huyen en cuanto terminan. Este asesinato, en cambio, fue hecho de manera deliberada y el asesino dejó sus huellas en todo el cuarto, mostrando que permaneció ahí mucho tiempo.

Para expresar esta cita, utilizaremos las siguientes variables proposicionales:

r : fue un robo

p : fue la política (motivos políticos)

s : algo fue sustraído

h : el asesino huyó inmediatamente

m : fue una mujer

c : el asesino dejó sus huellas en todo el cuarto

Veamos la derivación que llevó a cabo Sherlock Holmes en la tabla 2.11 y que lo llevó a concluir que fue una mujer.

Tabla 2.11. Análisis dado por Sherlock Holmes

Derivación	Regla	Comentario
1. $r \rightarrow s$	Premisa	Si fue un robo entonces algo debió ser sustraído
2. $\neg s$	Premisa	Nada fue sustraído
3. $\neg r$	Modus Tollens 1, 2	No fue un robo
4. $\neg r \rightarrow p \vee m$	Premisa	Si no fue un robo, debió ser motivo político o una mujer
5. $p \vee m$	Modus Ponens 3, 4	Fue motivo político o una mujer
6. $p \rightarrow h$	Premisa	Si fue motivo político, el asesino debió huir inmediatamente
7. $c \rightarrow \neg h$	Premisa	Si el asesino dejó huellas en todo el cuarto, no huyó inmediatamente
8. c	Premisa	El asesino dejó huellas en todo el cuarto
9. $\neg h$	Modus Ponens 7, 8	El asesino no huyó inmediatamente
10. $\neg p$	Modus Tollens 6, 9	El motivo no fue político
11. m	Silogismo Disyuntivo 5, 10	Por lo tanto debió ser una mujer

La secuencia de argumentos utilizados se muestra en la lista a continuación. En ella se puede observar claramente como las conclusiones que se van obteniendo de los argumentos, se pueden utilizar como premisas en argumentos sucesivos.

1. $r \rightarrow s$	
2. $\neg s$	
3. $\neg r$	Modus Tollens
3. $\neg r$	
4. $\neg r \rightarrow p \vee m$	
5. $p \vee m$	Modus Ponens
7. $c \rightarrow \neg h$	
8. c	
9. $\neg h$	Modus Ponens

6.	$p \rightarrow h$	
9.	$\neg h$	Modus Tollens
<hr/>		
10.	$\neg p$	
5.	$p \vee m$	
10.	$\neg p$	Silogismo Disyuntivo
<hr/>		
11.	m	

Las secuencias de composición de argumentos que acabamos de mostrar en los ejemplos anteriores se llaman *derivaciones*, *pruebas* o *deducciones formales*. A continuación las estudiamos de manera formal.

Sistemas para derivaciones

Los aspectos de la lógica relacionados con el estudio de las derivaciones conforman lo que se llama *teoría de la demostración* en contraste con los aspectos semánticos cuyo estudio se conoce como *teoría de modelos*. En esta sección describimos formalismos para desarrollar pruebas o derivaciones en lógica proposicional de manera sistemática, los cuales se conocen como cálculos deductivos o sistemas para derivaciones.

Aunque existen diversos sistemas para desarrollar derivaciones, todos tienen las siguientes características en común:

1. Hay un conjunto de argumentos lógicos admisibles, que definimos ya como reglas de inferencia. Nos referiremos a este conjunto con \mathcal{R} . Formalmente cada elemento de \mathcal{R} es en realidad un esquema de argumento, el cual debe ser un argumento correcto. En algunos casos se aceptan argumentos sin premisas los cuales se llaman *axiomas*.
2. La derivación es en sí misma una lista de expresiones lógicas. Originalmente, la lista está vacía y una expresión puede agregarse a la lista si es una premisa, o si se puede obtener como conclusión de alguna de las reglas de inferencia de \mathcal{R} a partir de expresiones que se encuentran previamente en la lista. Este proceso continúa hasta que se llega a la fórmula B que se desea obtener como conclusión. En tal caso decimos que la lista completa es una derivación de B .

Estas características describen el conocido método axiomático introducido por Euclides en sus “*Elementos*” donde están las bases de la geometría euclidiana.

La siguiente definición es de importancia.

Definición 2.15 Sean $\Gamma = \{A_1, \dots, A_n\}$ un conjunto de fórmulas. Si existe una derivación de B a partir de Γ , es decir, donde las premisas son fórmulas del conjunto Γ , entonces decimos que B es derivable a partir de Γ y escribimos $\Gamma \vdash_{\mathcal{R}} B$, o simplemente $\Gamma \vdash B$ si el conjunto de reglas de inferencia válidas ya es conocido.

Por lo general el conjunto de reglas de inferencia \mathcal{R} está fijo desde un principio, de manera que únicamente pueden usarse reglas de inferencia que figuran en él. En nuestro caso no seremos tan estrictos y permitiremos usar cualquier regla previamente derivada, aunque esencialmente usaremos las siguientes:

Tabla 2.13. Principales reglas de inferencia

Regla	Nombre	Notación
$A \quad B / A \wedge B$	Introducción de \wedge	$I \wedge$
$A \wedge B / B$	Eliminación de \wedge	$E \wedge$
$A \wedge B / A$	Eliminación de \wedge	$E \wedge$
$A / A \vee B$	Introducción de \vee	$I \vee$
$B / A \vee B$	Introducción de \vee	$I \vee$
$A \quad A \rightarrow B / B$	Modus Ponens	MP
$\neg B \quad A \rightarrow B / \neg A$	Modus Tollens	MT
$A \rightarrow B \quad B \rightarrow C / A \rightarrow C$	Silogismo Hipotético	SH
$A \vee B \quad \neg A / B$	Silogismo Disyuntivo	SD
$A \vee B \quad \neg B / A$		SD
$A \rightarrow B \quad \neg A \rightarrow B / B$	Dilema Constructivo simple	DCS
$A \leftrightarrow B / A \rightarrow B$	Eliminación de equivalencia	$E \leftrightarrow$
$A \leftrightarrow B / B \rightarrow A$		$E \leftrightarrow$
$A \rightarrow B \quad B \rightarrow A / A \leftrightarrow B$	Introducción de Equivalencia	$I \leftrightarrow$
$A, \neg A / B$	Inconsistencia	Inc

Es momento de desarrollar algunos ejemplos.

Ejemplo 2.39. Mostrar la correctud del siguiente argumento:

$$p \rightarrow r, r \rightarrow s, t \vee \neg s, \neg t \vee u, \neg u / \therefore \neg p.$$

Desarrollaremos una derivación de $\neg p$ con premisas $\Gamma = \{p \rightarrow r, r \rightarrow s, t \vee \neg s, \neg t \vee u, \neg u\}$.

Derivación:

1. $p \rightarrow r$ Premisa
 2. $r \rightarrow s$ Premisa
 3. $t \vee \neg s$ Premisa
 4. $\neg t \vee u$ Premisa
 5. $\neg u$ Premisa
 6. $p \rightarrow s$ (SH) Silogismo hipotético con 1, 2
 7. $\neg t$ (SD) Silogismo disyuntivo con 4, 5
 8. $\neg s$ (SD) Silogismo disyuntivo con 7, 3
 9. $\neg r$ (MT) Modus tollens con 2, 8
 10. $\neg p$ (MT) Modus tollens con 9, 1
-

Ejemplo 2.40. Mostrar la correctud del siguiente argumento

$$p \rightarrow q, q \rightarrow r \wedge s, \neg r \vee \neg t \vee u, p \wedge t / \therefore u.$$

Derivación:

1. $p \rightarrow q$ Premisa
 2. $q \rightarrow r \wedge s$ Premisa
 3. $\neg r \vee \neg t \vee u$ Premisa
 4. $p \wedge t$ Premisa
 5. $p \rightarrow r \wedge s$ SH 1,2
 6. p E \wedge 4
 7. $r \wedge s$ MP 5,6
 8. r E \wedge 7
 9. $\neg t \vee u$ SD 8,3
 10. t E \wedge 4
 11. u SD 9,10
-

Ejemplo 2.41. Mostrar la correctud del siguiente argumento:

Si la banda no puede tocar cumbia o las cervezas no llegan temprano, entonces la fiesta de fin de semestre se cancelaría y Menelao montaría en cólera. Si la fiesta se cancela, hay que devolver las entradas. No se devolvieron las entradas. Luego entonces la banda pudo tocar cumbia.

Se asignan las siguientes variables proposicionales:

- | | |
|------------------------------------|-----------------------------------|
| b : La banda pudo tocar cumbia | m : Menelao monta en cólera |
| c : Las cervezas llegan temprano | d : Hubo que devolver el dinero |
| f : La fiesta se cancela | |

El argumento a verificar es: $\neg b \vee \neg c \rightarrow f \wedge m, f \rightarrow d, \neg d / \therefore b$.

Derivación:

- | | | |
|-----|---|--------------|
| 1. | $\neg b \vee \neg c \rightarrow f \wedge m$ | Premisa |
| 2. | $f \rightarrow d$ | Premisa |
| 3. | $\neg d$ | Premisa |
| 4. | $\neg f$ | MT 2, 3 |
| 5. | $\neg f \vee \neg m$ | I \vee 4 |
| 6. | $\neg(f \wedge m)$ | RE 5 |
| 7. | $\neg(\neg b \vee \neg c)$ | MT 6,1 |
| 8. | $\neg\neg b \wedge \neg\neg c$ | RE 7 |
| 9. | $b \wedge c$ | RE 8 |
| 10. | b | E \wedge 9 |

Se observa en los pasos 6, 8 y 9 el uso de razonamiento ecuacional (RE); muchas veces éste se da por sobreentendido y no se menciona, por lo que podríamos haber pasado del paso 5 al 7 o del paso 7 al 9 directamente.

Estrategias para la construcción de derivaciones

En esta sección presentamos algunas estrategias o métodos para la derivación de argumentos correctos. La meta es construir una derivación $\Gamma \vdash B$. De acuerdo al conectivo principal de la conclusión B de un argumento, podemos simplificar la derivación del mismo.

Conjunción

Para derivar una conjunción $\Gamma \vdash P \wedge Q$ basta derivar ambos operandos por separado. Es decir,

- Si $\Gamma \vdash P$ y $\Gamma \vdash Q$, entonces $\Gamma \vdash P \wedge Q$.

Esta propiedad es inmediata de la regla de inferencia ($\wedge I$). Obsérvese que la afirmación recíproca también es cierta en nuestro sistema de derivación aunque podría fallar en otros sistemas.

Disyunción

De acuerdo a la regla de introducción de la disyunción ($\vee I$), para mostrar $\Gamma \vdash P \vee Q$ basta mostrar alguno de los dos operandos. Es decir,

- Si $\Gamma \vdash P$ o bien $\Gamma \vdash Q$, entonces $\Gamma \vdash P \vee Q$.

En este caso la afirmación recíproca no es necesariamente cierta; por ejemplo, tenemos $p \vee q \vdash p \vee q$ pero no es posible derivar $p \vee q \vdash p$ ni $p \vee q \vdash q$.

Implicación

Cuando tratamos de derivar una implicación basta suponer como premisa adicional el antecedente y derivar a partir de ello el consecuente. Esto se debe a que para mostrar la verdad de una implicación, basta examinar aquellos casos en que el antecedente es verdadero y corroborar que de ese antecedente se infiere el consecuente; si el antecedente es falso, la implicación es verdadera no importando el valor del consecuente. Esto se expresa en la siguiente propiedad conocida como el *meta teorema de la deducción*:

- Si $\Gamma, P \vdash Q$ entonces $\Gamma \vdash P \rightarrow Q$.

Obsérvese que esta regla se usa prácticamente siempre en las demostraciones matemáticas en general.

Ejemplo 2.42. Supongamos que deseamos demostrar

$$\vdash P \rightarrow P \vee Q$$

Utilizando la propiedad anterior basta encontrar una derivación

$$P \vdash P \vee Q$$

la cual es inmediata de la regla de introducción de la disyunción ($\vee I$).

Equivalencia

Para derivar una equivalencia $P \leftrightarrow Q$ basta probar ambas implicaciones.

- Si $\Gamma \vdash P \rightarrow Q$ y $\Gamma \vdash Q \rightarrow P$, entonces $\Gamma \vdash P \leftrightarrow Q$.

Nuevamente esta propiedad es muy común en demostraciones matemáticas.

Negación

Para derivar una negación no hay estrategia general. En algunos casos podemos usar equivalencias lógicas, por ejemplo si deseamos $\Gamma \vdash \neg(P \wedge Q)$ entonces basta mostrar $\Gamma \vdash \neg P \vee \neg Q$; para demostrar esto último podemos usar la estrategia para la disyunción y probar alguna de $\Gamma \vdash \neg P$ o bien $\Gamma \vdash \neg Q$.

Un sistema de derivación \mathcal{S} debe ser tal que no se puedan derivar resultados que no son sólidos. Esto es, \mathcal{S} no debe contener ninguna *falacia*, una regla de inferencia que permite concluir algo que no está implicado por las premisas y que por lo tanto no es válido.

También es deseable que un sistema de derivación sea *completo*, esto es, que sea posible derivar absolutamente a todas las conclusiones que sean consecuencia lógica de las premisas. Por ejemplo, la tabla 2.13 no nos da un sistema completo, pues hay leyes (tautologías), como la del *Tercero excluido*, que no se puede derivar de ellas. Como no hay forma de derivar esta ley a partir de las que se dan en la tabla, el sistema no es completo; si se desea validar esta propiedad, debemos agregarla como regla básica o axioma:

$$/ P \vee \neg P \quad (TE)$$

Ejercicios

2.6.1.- Use los identificadores P y Q para formalizar los siguientes argumentos. Además indique de cuál de las reglas de inferencia son instancia.

- a) Si 10 es primo, 10 no puede ser igual a 2 veces 5. 10 es 2 veces 5. Por lo tanto, 10 no puede ser primo.
- b) Si llueve frecuentemente, los agricultores se quejan; si no llueve frecuentemente, los agricultores se quejan. En conclusión, los agricultores se quejan.

2.6.2.- Para los siguientes argumentos decida si son correctos y en caso de no serlo dé un interpretación que haga verdaderas a las premisas y falsa a la conclusión.

- a) $(p \rightarrow q) \wedge (p \rightarrow r) / \therefore q \rightarrow r$
- b) $p \vee q \rightarrow r, s \rightarrow p, s / \therefore r$
- c) $p \vee q, \neg (p \wedge r), \neg q / \therefore r \rightarrow s$
- d) $p \rightarrow q, p \vee r, \neg (r \wedge s) / \therefore (p \rightarrow q) \rightarrow (q \vee \neg s)$

2.6.3.- Dé un ejemplo, en español, para cada uno de las siguientes reglas de inferencia

- a) Silogismo hipotético.
- b) Silogismo disyuntivo.
- c) Eliminación de \wedge .
- d) Introducción de \vee .
- e) Inconsistencia.

2.6.4.- Identifique qué regla de inferencia corresponde a los siguientes argumentos en español.

- a) Si vamos al cine, nos desvelamos. No me quiero desvelar. Entonces no vamos al cine.
- b) ¡Me pagas la deuda o te quito la televisión! No me pagaste la deuda. Entonces te quito la televisión.
- c) Si el número de visitas es a lo más 15, estarán todos en la sala. Hay visitas en la recámara. Es porque vinieron más de 15.
- d) Ese muchacho se llama Juan o Pedro. No se llama Juan. Entonces se llama Pedro.

2.6.5.- Utilice el método de interpretaciones para analizar la validez del siguiente argumento:

Si la gorila es atractiva, el gorila sonreirá abiertamente o será infeliz. Si no es feliz, no procreará en cautiverio. Por consiguiente, si la gorila es atractiva, entonces, si el gorila no sonríe abiertamente, no procreará en cautiverio.

2.6.6.- Decida si el siguiente argumento es correcto usando interpretaciones:

$$\frac{\begin{array}{l} A \vee B \\ C \vee \neg A \\ B \rightarrow C \end{array}}{\therefore C}$$

2.6.7.- Traduzca y decida la validez del siguiente argumento mediante interpretaciones, definiendo previamente el significado de cada variable utilizada:

Si Chubaka no es perro entonces no es cierto que sea alado o borogrove. Si Chubaka es quelite entonces es alado. Sabemos que Chubaka no es perro. Luego entonces, Chubaka no es quelite.

2.6.8.- Decida la validez del siguiente argumento proposicional usando interpretaciones:

Si Polo es un animal entonces come queso y es ratón; si es ratón o no come queso entonces es selenita y troglodita. Polo es troglodita si y sólo si es unicornio. Luego entonces, si Polo es un animal entonces es un unicornio.

2.6.9.- Construya las siguientes derivaciones

- a) $p \rightarrow q, r \rightarrow s, \neg q \vee \neg s \vdash \neg p \vee \neg r$
- b) $\vdash p \vee (p \wedge \neg q \rightarrow r)$
- c) $\vdash p \vee (\neg p \wedge q) \rightarrow p \vee q$
- d) $\vdash (p \rightarrow q) \rightarrow (p \vee q \rightarrow q)$
- e) $\vdash (\neg p \wedge (\neg p \wedge q)) \vee (p \wedge (p \wedge \neg q)) \leftrightarrow (\neg p \wedge q) \vee (p \wedge \neg q)$

2.7. Tableaux semánticos para el cálculo proposicional

Una de las preocupaciones de la lógica proposicional (y de la de predicados que veremos más adelante) es la de determinar si una fórmula es o no *razonable*. Esto último quiere decir que deseamos determinar si existe algún estado en el que la fórmula se evalúe a verdadero; o dicho de otra manera, si hay alguna asignación posible a las variables proposicionales que participan en la fórmula de tal manera que ésta se evalúa a verdadera (dicho de una tercera forma, si la fórmula tiene modelo).

Uno de los mecanismos que podemos utilizar para determinar si una fórmula es razonable es la de elaborar la tabla de verdad de la misma. Sin embargo, como ya hemos mencionado, la tarea de elaborar tablas de verdad cuando estamos hablando de fórmulas de más de tres o cuatro variables se vuelve un problema intratable, ya que tendremos que examinar 2^n posibles estados.

Un mecanismo que permite de manera eficiente y segura determinar si una fórmula es tautología, contradicción o contingencia, y encontrar un estado para el cual la fórmula se evalúa a verdadera, es el de las tablas semánticas o *tableaux*.

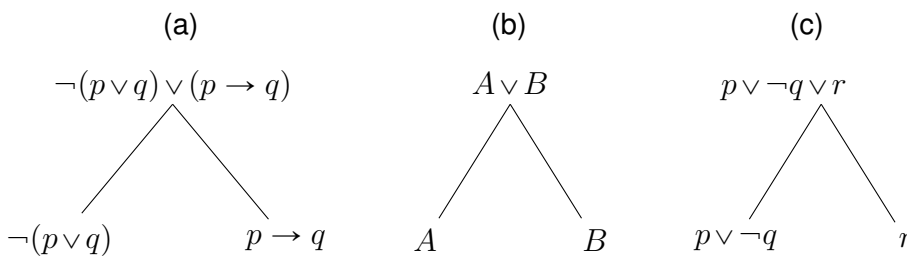
2.7.1. El concepto de tableau

Un *tableau* corresponde a un árbol cuya función es buscar una interpretación para determinada fórmula. Los tableaux toman la forma de un árbol, parecido a los árboles de derivación. Las fórmulas que van a ser representadas en un tableau deben consistir únicamente de conjunciones y disyunciones de literales o constantes, que son fórmulas atómicas (true, false, p, q, r, \dots) o negaciones de literales o constantes (\neg true, \neg false, $\neg p, \neg q, \neg r, \dots$). Estas fórmulas no pueden tener ningún otro operador, pero esto no nos debe preocupar ya que vimos que es posible eliminar la implicación y la bicondicional sustituyéndolas por disyunciones y conjunciones. También podemos eliminar la negación de una fórmula disyuntiva o conjuntiva ($\neg(p \wedge q)$) utilizando las leyes de De Morgan ($\neg(p \wedge q) \equiv \neg p \vee \neg q$). Es importante, sin embargo, mantener la asociatividad de los operadores dada por la fórmula original (preservar la precedencia original o sea trabajar con fórmulas donde todos los paréntesis que indican precedencia son explícitos).

La construcción de tableaux tiene realmente muy pocas reglas. Veámoslas:

1. La fórmula para la que deseamos construir el tableau aparece como raíz del árbol.
2. Si el esquema de la fórmula es una disyunción ($A \vee B$), de la raíz del subárbol se abren dos ramas, una para la fórmula A y otra para la fórmula B , como podemos ver en la figura 2.4.

Figura 2.4. Construcción de tableau para la disyunción

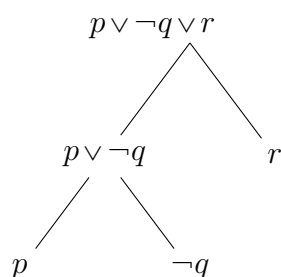


Como el operador \vee es conmutativo y asociativo, se puede intercambiar el orden de las ramas de los árboles. También utilizamos la propiedad asociativa de la disyunción en el caso de la fórmula del tableau 2.4(c) y decidimos “abrir” primero la segunda disyunción.

Por lo pronto, dejamos a los tableaux desarrollados únicamente en el primer nivel, lo que deja ramas que deben ser expandidas en el primer y tercer caso. Más adelante

veremos cuándo y cómo conviene extender un tableau. Conforme se avanza en la fórmula, se va “componiendo” con el árbol que se tiene hasta ese momento. Lo que debe quedar claro es que en la fórmula 2.4(a) no podemos extender, tal como están, a ninguna de las fórmulas en el segundo nivel del árbol, ya que no corresponden a esquemas de disyunción o conjunción; en esta expresión, la fórmula de la izquierda corresponde a un esquema de negación, mientras que la segunda es una condicional; así que por lo pronto posponemos su extensión hasta que demos las reglas de transformación para tableaux. En cambio, en la fórmula 2.4(c) sí tenemos en la rama izquierda un esquema de disyunción, por lo que ya podemos expandirla, quedando el tableau como se muestra en la figura 2.5.

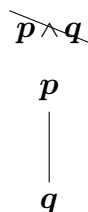
Figura 2.5. Desarrollo completo del tableau de la fórmula 2.4(c)



3. Si el esquema de la fórmula es una conjunción ($A \wedge B$) se pone a uno de los operandos como hijo del otro (como el operador \wedge es conmutativo, el orden no importa). Podemos ver tres ejemplos en las figuras 2.6 a 2.8.

En la fórmula de la figura 2.6 tenemos un esquema de conjunción, donde cada uno de los operandos es una variable proposicional.

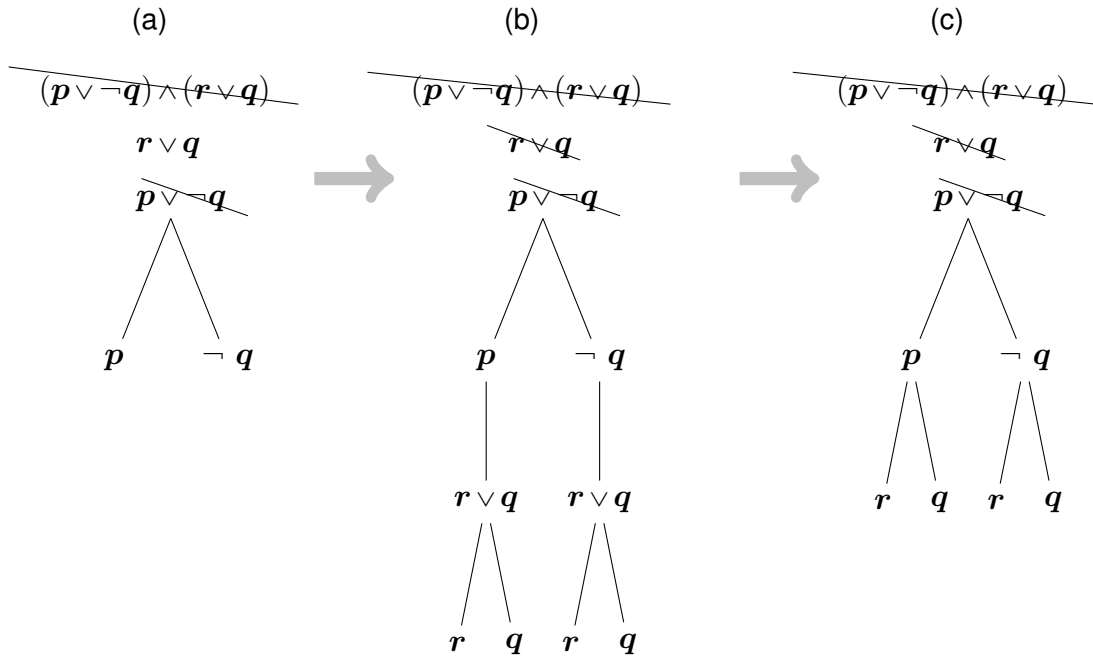
Figura 2.6. Primer ejemplo de tableau para representar conjunciones



En la fórmula de la figura 2.7 en la siguiente página, tenemos un esquema de conjunción donde cada operando es, a su vez, una disyunción. Entonces, listamos los dos operandos, uno abajo del otro (el orden no importa) y procedemos a construir el tableau para uno de ellos, en este caso el primero. Una vez que tenemos en el tableau como hojas únicamente variables proposicionales que ya no pueden descomponerse más, colgamos de cada una de las ramas al otro operando y procedemos a abrirlo. Mostramos en el tableau de la figura 2.7(b) el nivel intermedio para la fórmula $r \vee q$,

aunque esto no es necesario, sino que podríamos haber colgado directamente la conjunción, como se ve en el tercer tableau de esta fórmula.

Figura 2.7. Segundo ejemplo de tableau para representar conjunciones

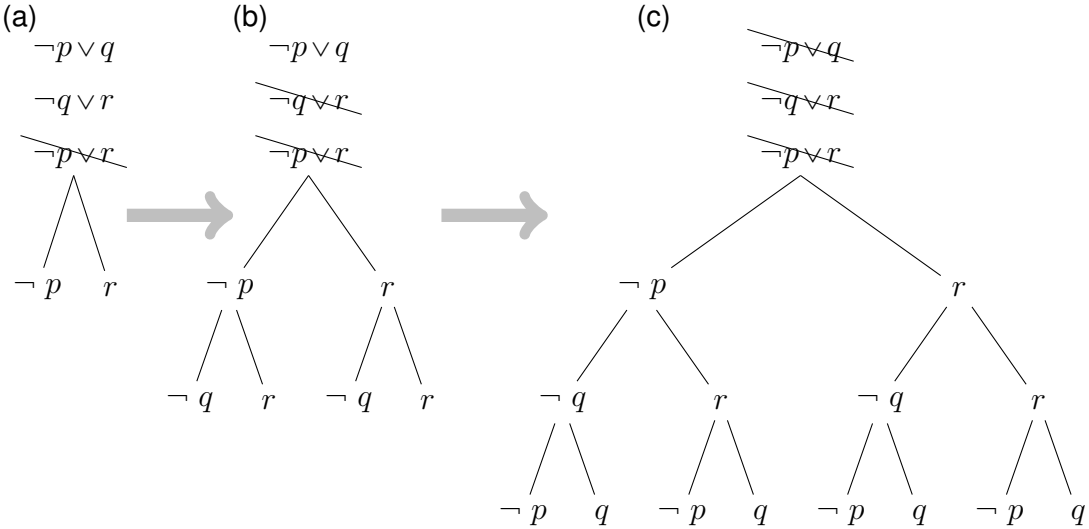


Para la tercera fórmula tenemos también un esquema de conjunción. Como la conjunción es asociativa, podemos asociar $((\neg p \vee q) \wedge (\neg q \vee r)) \wedge (\neg p \vee r)$, que es como lo hicimos, o pudiéramos usar también la conmutatividad de este operador. Listamos los tres operandos uno abajo del otro y desarrollamos el tableau del último $(\neg p \vee r)$ como primer paso. A continuación colgamos de todas las ramas de este tableau al segundo operando $(\neg q \vee r)$ y lo tachamos –ya no pusimos la subfórmula original explícitamente en el árbol–. Una vez que tenemos únicamente variables proposicionales como hojas del tableau, como tercer paso colgamos de cada una de las ramas a la primera fórmula $(\neg p \vee q)$. El tableau construido de esta manera es el último en la figura 2.8 de la siguiente página.

Este caso es, claramente, un poco más complicado que el caso de la bifurcación. La intención con la que se construyeron los árboles (tableaux) es la de que, como se trata de una conjunción, cualquier “camino” en el árbol debe contemplar a todos los operandos de la conjunción. En el primer ejemplo, simplemente tenemos dos variables proposicionales, por lo que las ponemos en el árbol a una de ellas como descendiente de la otra. En el segundo ejemplo, desarrollamos uno de los operandos de la disyunción y de cada hoja, en la que hay únicamente variables proposicionales, “colgamos” a la otra proposición desarrollada como tableau. Como el tableau para $r \vee q$ es un tableau con dos ramas, éste se cuelga tanto de p como de $\neg q$.

Figura 2.8. Tercer ejemplo de tableaux con disyunción

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



El tercer ejemplo consiste de dos operadores \wedge (tres operandos). En el primer nivel colocamos (es arbitraria esta elección) al tercer operando. Una vez que lo desarrollamos completo, colgamos de cada una de las ramas el segundo operando, a su vez desarrollado ya en un tableau; por último, tenemos que colocar el operando que nos falta, $\neg p \vee q$, colgándolo de cada una de las ramas que llevamos hasta el momento. El orden no es importante, siempre y cuando hayamos incluido para desarrollar a todas las subfórmulas en la manera en que indicamos.

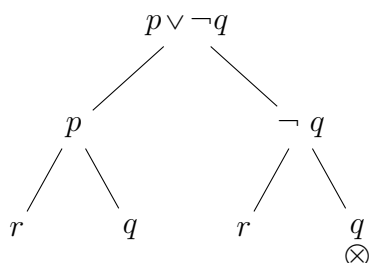
Dado que únicamente tenemos reglas de construcción para la disyunción y la conjunción, debemos decidir qué hacer con aquellas fórmulas que involucren otros operadores. Tenemos dos opciones: transformar la fórmula antes de construir el tableau, usando propiedades de los operadores, asociatividad, conmutatividad y las Leyes de De Morgan, y proceder después a desarrollar el tableau de la fórmula resultante. Otra opción es ir transformando las subfórmulas durante la construcción del tableau. Esta estrategia nos puede ahorrar trabajo por razones que no tardaremos en explicar. Además, es la que más se beneficia del uso de tableaux.

2.7.2. Eliminación de ramas del tableau

Como dijimos antes, vamos a utilizar los tableaux para determinar si una fórmula es satisfacible o no. Por como están contruidos, si seguimos un camino dentro del tableau vamos a tener la conjunción de variables proposicionales. Por ejemplo, en el caso del tableau de la figura 2.6 simplemente tenemos que el único camino en el árbol es salir de p y llegar a q . Pero en el caso del tableau de la figura 2.7, el camino $(p \vee \neg q) \wedge (r \vee q)$ nos indica

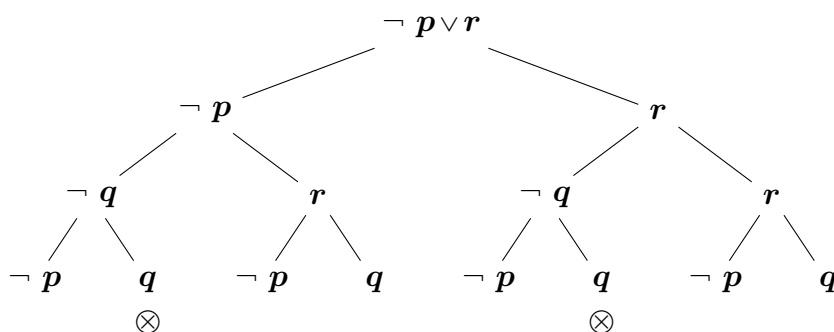
la subfórmula $\neg q \wedge q$, que es una contradicción, por lo que “siguiendo” esa rama ya no va a satisfacer a la fórmula (la conjunción será evaluada a falso). Cada vez que encontramos, en un camino (una rama) dentro del árbol, una literal y su literal complementaria⁹, podemos *cerrar* esa rama y ya no extenderla más, pues no importa qué fórmulas le colguemos a ese camino tendremos una conjunción con falso, lo que hace a la fórmula representada por ese camino falsa. Denotamos que un camino está cerrado colocando el símbolo \otimes . En la figura 2.7 se habría eliminado una rama, como se puede observar en la figura 2.9.

Figura 2.9. Cierre de ramas en la figura 2.7



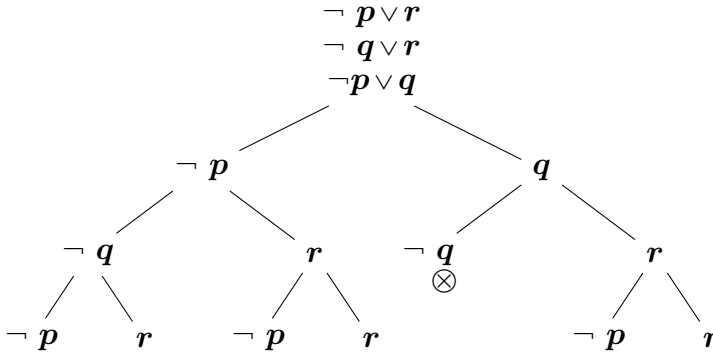
Por ejemplo, el tableau de la figura 2.8 no presenta ninguna rama cerrada que nos ahorre trabajo –ver figura 2.10–.

Figura 2.10. Cierre de ramas en un tableau para la fórmula $(\neg p \vee r) \wedge (\neg q \vee r) \wedge (\neg p \vee q)$



Sin embargo, hasta ahora únicamente hemos podido cerrar ramas que ya están totalmente desarrolladas, pero lo que queremos es *ahorrar* trabajo, esto es, cerrar ramas lo antes posible. Si hubiésemos seguido otro orden en la doble conjunción en la fórmula de la figura 2.8, buscando que aparezcan lo antes posible una literal y su complementaria, habríamos podido llevar a cabo menos trabajo. Por ejemplo, si el orden en que colgamos del tableau es $(\neg p \vee q)$, $(\neg q \vee r)$ y por último $(\neg p \vee r)$, tenemos lo antes posible la contradicción, como se muestra en la figura 2.11 en la siguiente página.

⁹La literal complementaria de una literal dada L se define como A si L es $\neg A$ y como $\neg A$, si L es A , siendo A una fórmula atómica.

Figura 2.11. Orden de armado del tableau para cerrar lo más pronto posible

Las ramas que cerremos ya no tiene sentido seguir expandiéndolas y eso nos va a ahorrar trabajo. Si todas las ramas quedan cerradas, la fórmula es una contradicción; sin embargo, el que todas las ramas queden abiertas **no** significa que tenemos una tautología. Como ejemplo veamos la fórmula de la figura 2.6, donde todas sus ramas (exactamente una) quedaron abiertas y, sin embargo, esta fórmula sólo será verdadera en el caso en que $\mathcal{I}(p) = \mathcal{I}(q) = 1$; en caso de que algunas ramas queden abiertas y otras cerradas se trata de una fórmula contingente. Para determinar si una fórmula es tautología tenemos que construir el tableau para su negación; si en este tableau se cierran todas las ramas, tenemos que la negación de la fórmula es contradicción y por lo tanto la original es tautología.

2.7.3. Reglas para los tableaux

Vamos a optar por ir “abriendo” las fórmulas conforme las vamos incluyendo en el tableau; la razón para ello es que si nos encontramos con ramas cerradas antes de agregar alguna conjunción, nos ahorramos el trabajo de transformar la regla. Como en el caso del razonamiento ecuacional y de la sustitución textual, es muy importante determinar cuál es el esquema principal que estamos procesando: cuál es el operador que domina. Las reglas que podemos usar para transformar las fórmulas y poderlas agregar al tableau en desarrollo se encuentran a continuación.

- α -reglas:

1. De $A \wedge B$ se deduce A y B . $\alpha(1)$
2. De $\neg(A \vee B)$ se deduce $\neg A$ y $\neg B$. $\alpha(2)$
3. De $\neg(A \rightarrow B)$ se deduce A y $\neg B$. $\alpha(3)$

- β -reglas:

1. De $A \vee B$ se deduce A y, en una rama separada, B . $\beta(1)$
2. De $\neg(A \wedge B)$ se deduce $\neg A$ y, en una rama separada, $\neg B$. $\beta(2)$
3. De $A \rightarrow B$ se deduce $\neg A$ y, en una rama separada, B . $\beta(3)$

- σ -reglas:

1. De $\neg\neg A$ se deduce A . $\sigma(1)$
2. De $\neg\text{false}$ se deduce true . $\sigma(2)$
3. De $\neg\text{true}$ se deduce false . $\sigma(3)$

Las reglas σ son auxiliares y pueden evitarse usando razonamiento ecuacional.

- Reglas de cierre:

1. Cerrar cualquier rama que tenga A y $\neg A$ (para cualquier A),
o bien tenga $\neg \text{true}$, o false . (cierre)

Obsérvese que no hemos dados reglas para la equivalencia. Dependiendo del caso una fórmula de la forma $A \leftrightarrow B$ puede transformarse con alguna equivalencia lógica conveniente, de manera que se pueden usar las reglas α o β . Usualmente

$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A) \text{ o bien } A \leftrightarrow B \equiv (A \wedge B) \vee (\neg A \wedge \neg B).$$

Veamos algunos ejemplos de construcción de tableaux.

Ejemplo 2.43. [Ley de Peirce] Demostrar $\vdash \left(((p \rightarrow q) \rightarrow p) \rightarrow p \right)$, construyendo el tableau correspondiente a su negación.

De lo anterior, usando $\alpha(3)$ tenemos:

$$\neg \left(((p \rightarrow q) \rightarrow p) \rightarrow p \right) = \left(((p \rightarrow q) \rightarrow p) \wedge \neg p \right),$$

por lo que pasamos a desarrollar el tableau de esta conjunción en la figura 2.12 en la siguiente página.

Vemos que no queda ninguna rama abierta, lo que denota a una contradicción. Como el tableaux se armó para la negación de la fórmula original y tenemos una contradicción para esta fórmula, podemos deducir que la fórmula original es una tautología.

Ejemplo 2.44. Demostrar que el silogismo hipotético es una tautología:

$$(P \rightarrow Q) \wedge (Q \rightarrow R) \rightarrow (P \rightarrow R)$$

Para demostrar que esta fórmula es tautología trabajamos con su negación:

$$\neg \left(((P \rightarrow Q) \wedge (Q \rightarrow R)) \rightarrow (P \rightarrow R) \right).$$

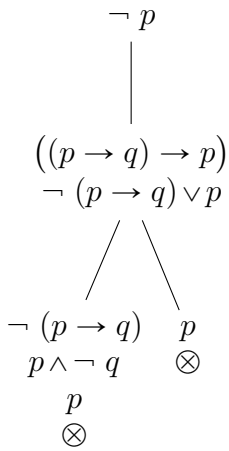
Ver figura 2.13 en la siguiente página para desarrollo del tableau correspondiente.

Como todas las ramas están cerradas, la fórmula es una contradicción y, por lo tanto, la fórmula original es tautología (lo que ya sabíamos).

Figura 2.12. Construcción del tableau para el ejemplo 2.43.**Fórmula:**

$$\neg \left(((p \rightarrow q) \rightarrow p) \rightarrow p \right) \equiv \left(((p \rightarrow q) \rightarrow p) \wedge \neg p \right)$$

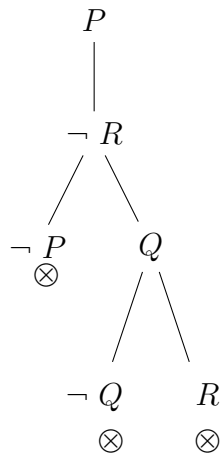
$$((p \rightarrow q) \rightarrow p)$$

**Regla usada:** $\alpha(3)$ $\alpha(1)$ $\beta(3)$ $\alpha(3)$
*cierre**cierre***Figura 2.13.** Construcción del tableau para el ejemplo 2.44**Fórmula:**

$$((P \rightarrow Q) \wedge (Q \rightarrow R)) \wedge \neg (P \rightarrow R)$$

$$\neg (P \rightarrow R)$$

$$P \wedge \neg R \quad \neg (P \rightarrow R) \quad a \quad P \wedge \neg R$$

**Regla usada:** $\alpha(3)$ $\alpha(3)$ $\beta(3)$ $\beta(3)$

2.7.4. Modelo de una fórmula

Al desarrollar un tableau para una fórmula dada, trataremos de trabajar lo menos posible, esto es, abrir el menor número de ramas posibles. Ya vimos que una rama cerrada no tiene sentido seguirla extendiendo; las estrategias usadas deberán ir en la dirección de cerrar lo antes posible una rama. Estas estrategias las podemos resumir de la manera que se muestra en la siguiente página:

1. Descomponer primero las fórmulas que no abran ramas; es decir, usar las α -reglas y las σ -reglas antes que las β -reglas.
2. Dar prioridad a la descomposición de fórmulas que cierren ramas.
3. Parar cuando el problema esté resuelto (para demostrar satisfacibilidad basta con encontrar una rama abierta completa).
4. Cuando no sirvan las estrategias anteriores, empezar por las fórmulas más complejas (así más tarde habrá menos ramas en las que se tenga que desarrollar la fórmula compleja).

El tableau de una fórmula también nos proporciona una interpretación para la fórmula que es modelo de la misma. De hecho, cada rama completa que queda abierta corresponde a una interpretación de la fórmula. Por lo tanto, para encontrar un modelo de una fórmula basta encontrar una rama abierta completa. La interpretación que corresponde a esa rama es como sigue:

1. A las variables que aparecen negadas en esa rama se les asigna el valor 0.
2. A las variables que aparecen sin negar en esa rama se les asigna el valor 1.
3. Aquellas variables que aparecen en la fórmula pero no en esa rama pueden tener cualquier asignación.

Nótese que no puede haber ninguna variable a la que se tuviera que asignar 0 y 1 en una misma rama, porque esto querría decir que aparece negada y sin negar, en cuyo caso la rama se habría cerrado.

2.7.5. Algoritmos para la lógica proposicional

Los tableaux son muy útiles en lógica proposicional pues proporcionan diversos algoritmos de decisión, algunos de los cuales enunciamos a continuación.

- ¿Es A una tautología?

Objetivo: Definir si A es tautología.

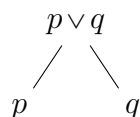
Entrada: A .

Salida: La decisión de si A es o no tautología.

Método:

- Construir el tableau \mathcal{T} para $\neg A$.
- Si \mathcal{T} se cierra entonces A es tautología.
- En otro caso, existe una rama abierta y completa en \mathcal{T} , la cual genera un modelo de $\neg A$, por lo que A no es tautología.

Para demostrar si una fórmula es tautología, construimos el tableau de la negación de la fórmula; si este tableau corresponde a una contradicción, entonces la fórmula es tautología. Este algoritmo se puede adaptar fácilmente para obtener uno que decida si A es una contradicción, observando que una fórmula es una contradicción si y sólo si todas las ramas de su tableau se cierran. Sin embargo, no podemos decir que una fórmula es tautología si todas sus ramas quedan abiertas, porque pudiera haber interpretaciones que no fueran modelo. Por ejemplo, la fórmula $p \vee q$, cuyo tableau aparece a continuación



es un tableau (muy simple) en el que todas las ramas quedaron abiertas y, sin embargo, no es tautología. Lo único que podemos concluir, respecto a interpretaciones, es que $\mathcal{I}_1(p) = 1$ e $\mathcal{I}_2(q) = 1$ son modelos de esta fórmula (la variable que no se menciona en cada una de las interpretaciones puede tomar cualquier valor).

- Si se desea clasificar una fórmula en tautología, contradicción o contingencia se usa el siguiente algoritmo.

Objetivo: Clasificar una fórmula A .

Entrada: Una fórmula A que deseamos clasificar como tautología, contradicción o contingencia.

Salida: El dictamen de si la fórmula es tautología, contradicción o contingencia.

Método:

- Construir el tableau \mathcal{T} de A .
- Si \mathcal{T} se cierra entonces A es contradicción.
- En otro caso, existe una rama abierta y completa en \mathcal{T} que proporciona un modelo \mathcal{I} de A .
 - Construir el tableau \mathcal{T}' para $\neg A$.
 - Si \mathcal{T}' se cierra entonces A es tautología.
 - En otro caso \mathcal{T}' tiene una rama abierta y completa que proporciona un modelo \mathcal{I}' de $\neg A$.
- Las interpretaciones \mathcal{I} e \mathcal{I}' muestran que A es contingente.

Con respecto a conjuntos de fórmulas tenemos los siguientes algoritmos.

- Satisfacibilidad de un conjunto de fórmulas Γ .

Objetivo: Decidir la satisfacibilidad de un conjunto de fórmulas Γ

Entrada: Un conjunto de fórmulas $\Gamma = \{A_1, \dots, A_n\}$.

Salida: La decisión de si Γ es o no satisfacible.

Método:

- Construir el tableau \mathcal{T} para $A_1 \wedge A_2 \wedge \dots \wedge A_n$.
- Si \mathcal{T} se cierra entonces Γ es insatisfacible.
- En otro caso existe una rama abierta y completa en \mathcal{T} , la cual genera un modelo de Γ , por lo que este conjunto es satisfacible.

- ¿Es A consecuencia lógica de Γ ?

Objetivo: Decidir la consecuencia lógica $\Gamma \models A$.

Entrada: Un conjunto de fórmulas Γ y una fórmula A .

Salida: La decisión de si A es consecuencia lógica de Γ (sí o no).

Método:

- Construir el tablero \mathcal{T} para el conjunto $\Gamma \cup \{\neg A\}$.
- Si \mathcal{T} se cierra entonces la consecuencia lógica $\Gamma \models A$ se da y el argumento que representa es correcto.
- En otro caso, existe una rama abierta y completa en \mathcal{T} por lo que la consecuencia es inválida y se genera un modelo de las premisas Γ donde la conclusión A es falsa.

Como ya se ha visto, los tableaux son un mecanismo para derivación de fórmulas y demostración de teoremas que resulta mucho más económico, en términos de trabajo, que el razonamiento ecuacional, por interpretaciones o mediante derivaciones; adicionalmente, es algorítmico, ya que siempre termina y no hay que ser creativos en el orden de abrir los tableaux; en el peor de los casos, haremos un poco más de trabajo, pero está garantizado que terminamos con la respuesta correcta.

Ejercicios

2.7.1.- Construya el tableau correspondiente a cada una de las fórmulas, sin cerrar ramas.

Para poder hacerlo, primero transforme la fórmula para que tenga únicamente conjunciones y disyunciones de literales.

- $((p \vee q) \wedge (r \rightarrow \neg p)) \rightarrow (r \rightarrow q)$
- $((p \rightarrow q) \rightarrow p) \rightarrow (q \rightarrow p)$
- $((p \vee q) \wedge (p \vee r)) \rightarrow p$

- d) $((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (\neg r \rightarrow \neg p)$
 e) $((r \vee \neg s) \vee t) \wedge \neg ((p \vee \neg q) \wedge (\neg q \vee \neg p))$
 f) $\neg ((p \rightarrow q) \wedge (q \rightarrow p))$
 g) $(p \rightarrow q) \vee r$

2.7.2.- Usando tableaux, determine cuál de las siguientes fórmulas es tautología, contradicción o contingente.

- a) $((p \vee q) \vee r) \wedge (p \vee (q \vee r)) \rightarrow p \vee q$ c) $p \vee q \rightarrow p \vee r$
 b) $(p \wedge (q \wedge r)) \rightarrow (p \rightarrow (q \rightarrow r))$ d) $(p \rightarrow (p \rightarrow q)) \rightarrow p$

2.7.3.- Demuestre que las siguientes fórmulas son tautologías usando tableaux:

- a) $(p \rightarrow q) \wedge (r \rightarrow s) \wedge (\neg q \vee \neg s) \rightarrow \neg p \vee \neg r$
 b) $p \vee (p \wedge \neg q \rightarrow r)$
 c) $p \vee (\neg p \wedge q) \rightarrow p \vee q$
 d) $(p \rightarrow q) \rightarrow (p \vee q \rightarrow q)$
 e) $(\neg p \wedge (\neg p \wedge q)) \vee (p \wedge (p \wedge \neg q)) \leftrightarrow (\neg p \wedge q) \vee (p \wedge \neg q)$

2.7.4.- Para los ejercicios 2.7.2 y 2.7.3, usando los tableaux construidos, dé un modelo para las fórmulas y para sus negaciones en el caso de que sean contingentes, así como un modelo contraejemplo para los argumentos incorrectos.

2.7.5.- Determine si los siguientes conjuntos de fórmulas Γ son satisfacibles, en cuyo caso dé un modelo.

- a) $\Gamma = \{\neg p \wedge q, (r \rightarrow p \leftrightarrow \neg q) \vee \neg r, \neg(r \vee \neg p)\}$
 b) $\Gamma = \{r \rightarrow \neg(p \wedge \neg q), ((p \rightarrow r) \rightarrow (\neg q \leftrightarrow r)) \wedge \neg r, \neg(q \wedge q)\}$
 c) $\Gamma = \{(p \wedge r) \vee (\neg r \rightarrow q), (q \leftrightarrow r) \rightarrow (\neg q \rightarrow r), \neg p \wedge q \wedge \neg r\}$
 d) $\Gamma = \{p \wedge \neg q \rightarrow \neg r, (\neg p \rightarrow \neg q) \wedge \neg r, r \wedge q \wedge \neg r\}$
 e) $\Gamma = \{\neg q \rightarrow \neg r, p, (\neg q \rightarrow p) \rightarrow q, \neg r \rightarrow p, s \rightarrow q, r \vee p\}$

2.7.6.- Usando tableaux, determine la correctud de los siguientes argumentos.

- a) $(p \rightarrow q) \wedge (p \rightarrow r) / \therefore q \rightarrow r$
 b) $p \vee q \rightarrow r, s \rightarrow p, s / \therefore r$
 c) $p \vee q, \neg(p \wedge r), \neg q / \therefore r \rightarrow s$
 d) $p \rightarrow q, p \vee r, \neg(r \wedge s) / \therefore (p \rightarrow q) \rightarrow (q \vee \neg s)$

2.7.7.- ¿Por qué es que se pueden cerrar ramas si es que aparece una literal y la literal complementaria en un camino dentro del tableau?

2.7.8.- Decida la validez del siguiente argumento mediante tableaux. Si es el caso, muestre un modelo contraejemplo.

$$\begin{array}{l}
 \neg(p \rightarrow q \vee s) \\
 t \rightarrow r \wedge q \\
 s \rightarrow \neg(t \wedge q) \\
 \hline
 \neg q \\
 \hline
 \therefore s
 \end{array}$$

2.7.9.- Determine, usando un tableau, si el siguiente argumento es o no válido.

Si un triángulo tiene tres ángulos, un cuadrado tiene cuatro ángulos rectos. Un triángulo tiene tres ángulos y su suma vale dos ángulos rectos. Si los rombos tienen cuatro ángulos rectos, los cuadrados no tienen cuatro ángulos rectos. Por lo tanto, los rombos no tienen cuatro ángulos rectos.

2.7.10.- Utiliza tableaux para determinar la validez del siguiente argumento.

$$p \rightarrow \neg r \rightarrow q, \neg r \vee \neg s \rightarrow \neg t, t \wedge \neg s \therefore \neg p$$

Lógica aplicada: | 3

Circuitos digitales

Una de las aplicaciones más importantes de la lógica matemática a la computación tiene que ver con el álgebra booleana –otra manera de interpretar a la lógica proposicional– y en particular con la construcción de circuitos digitales, que son los componentes primitivos de los dispositivos de cómputo. Por ello dedicamos este capítulo a revisar esta relación importantísima para las ciencias de la computación.

Esta aplicación es imprescindible para entender organización y arquitectura de computadoras, compiladores, sistemas operativos y redes, temas esenciales en las ciencias de la computación. Los prerrequisitos para estudiarlos se centran únicamente en la lógica proposicional, por lo que está incluido en este material.

En los circuitos digitales hay dos criterios que resultan ser importantes: el primero de ellos tiene que ver con el *costo* del circuito, que depende directamente del número de variables, el tamaño de una fórmula –en la lógica proposicional– circuitos y, hoy en día, en el número de circuitos que se pueden integrar en una tableta –*chip*–. En este último aspecto, la diferencia de una sola variable o subfórmula puede duplicar el costo total del circuito, pues exige el uso de dos tabletas en lugar de una.

En este capítulo, después de introducir la implementación de fórmulas proposicionales mediante circuitos digitales, insistiremos en el tema del costo, mediante la minimización de los circuitos digitales.

3.1. Fórmulas proposicionales y circuitos digitales

Hoy en día nos encontramos en todos lados dispositivos electrónicos llamados *microprocesadores* que controlan automóviles, lavadoras, refrigeradores, teléfonos celulares, relojes digitales y un sinnúmero de aparatos eléctricos y electrónicos que usamos ya casi sin pensar. El cómo funcionan está determinado por los circuitos que contienen y que están modelados con lo que se conoce como *álgebra booleana*¹, la cual responde a la lógica proposicional.

Como en la lógica proposicional, en el álgebra booleana tenemos variables que pueden tomar únicamente dos valores, 0 y 1 –¿recuerdan el uso de interpretaciones?–, y que se manipulan mediante tres funciones básicas:

- \bar{v} que corresponde a $\neg v$.
- $v_1 + v_2$ que corresponde a $v_1 \vee v_2$.
- $v_1 \cdot v_2$ (o simplemente $v_1 v_2$) que corresponde a $v_1 \wedge v_2$.

Al terminar el proceso de sustituir en una fórmula las implicaciones y bicondicionales por sus equivalentes lógicos, con conjunciones y disyunciones, obtenemos lo que se conoce como una *forma normal*. Si la fórmula resultante es una disyunción de conjunciones de literales, entonces tenemos lo que se conoce como *forma normal disyuntiva* –por ejemplo, $xyz + \bar{x}\bar{w} + w$ –, mientras que si lo que tenemos es una conjunción de disyunciones tenemos la *forma normal conjuntiva* –como sería la fórmula $(x + y + z)(\bar{x} + w)(\bar{y} + w)$ –. Esto no resta expresividad, pues recordemos que toda fórmula es equivalente a una que no utiliza ni implicaciones ni equivalencias (como vimos al construir tableaux).

En lo que sigue revisaremos varios ejemplos de circuitos digitales, maneras de combinarlos y encontrar circuitos que de alguna manera pueden ser considerados mínimos. Pero antes, y para acostumbrarnos a la nueva notación, enunciaremos algunas equivalencias lógicas de importancia en el álgebra booleana, además de algunas definiciones importantes.

- Leyes conmutativas:

$$x + y = y + x$$

$$xy = yx$$

- Leyes asociativas:

$$x + (y + z) = (x + y) + z$$

$$x(yz) = (xy)z$$

- Leyes distributivas:

$$x(y + z) = xy + xz$$

$$x + (yz) = (x + y)(x + z)$$

- Leyes de complementación:

$$x + \bar{x} = 1$$

$$x\bar{x} = 0$$

¹Usamos este término dado que es lo común en ciencias de la computación, aunque debemos mencionar que las estructuras matemáticas conocidas como álgebras booleanas son más generales, como se verá en el capítulo 6. Lo que aquí llamamos álgebra booleana es un ejemplo particular de aquellas estructuras.

Como ya dijimos, nos interesa expresar cualquier fórmula del álgebra booleana en una forma particular, llamada forma normal disyuntiva. Para ello, presentamos las definiciones que siguen:

Definición 1 Una literal ℓ es una variable x o una variable negada \bar{x} .

Definición 2 Un mintérmino m es un producto de literales $m = \ell_1 \ell_2 \dots \ell_k$, tal que una variable x figura a lo más una vez en m .

Por ejemplo, x , zy , $\bar{y}xz\bar{w}$ son mintérminos, pero los productos $xw\bar{x}$, $yzw\bar{z}$ no. Obsérvese que un mintérmino es verdadero exactamente en un estado posible de sus variables y que cada mintérmino corresponde a un renglón de la tabla de verdad. Por ejemplo, el mintérmino $x\bar{y}\bar{z}$ corresponde al estado donde x es verdadera, con tanto y como z falsas. Por esta misma razón, para n variables existen 2^n mintérminos distintos (igual que existen 2^n posibles estados).

Proposición 3.1 Toda fórmula proposicional A es equivalente a una suma de mintérminos:

$$A \equiv m_1 + m_2 + \dots + m_i + \dots + m_k$$

La representación de una fórmula como suma de mintérminos se conoce, como ya mencionamos, como forma normal disyuntiva o también como suma de productos, puesto que cada mintérmino es en realidad un producto (conjunción) de literales.

A continuación vemos algunos ejemplos sencillos de diseño de circuitos lógicos e introducimos la notación de compuertas lógicas.

Ejemplo 3.1. Construyamos un controlador para el aire acondicionado de una oficina. Queremos que si la temperatura ambiental sube a más de 28 grados el aire acondicionado se active; lo mismo si la humedad está por arriba del 50 %. El dispositivo tiene dos *entradas*, el estado de la temperatura (en realidad, sólo si la temperatura excede 28 grados) y si la humedad está por arriba o por abajo de 50 %. Como se ve, podemos representar ambas entradas con variables booleanas (lógicas) y cómo debe reaccionar el aire acondicionado con una tabla de verdad:

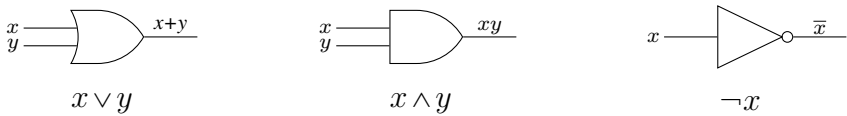
¿Temp > 28°?	¿Humedad > 50 %?	¿Aire encendido?
no	no	no
no	sí	sí
sí	no	sí
sí	sí	sí

Podemos observar que esta tabla de verdad corresponde –si asociamos 1 con “sí” y 0 con “no”– a la tabla de verdad del conectivo lógico \vee , por lo que si asignamos la variable c a que la temperatura sea mayor a 28°, h a que la humedad sea mayor a 50 % y p a si el aire acondicionado está o no encendido, podemos pensar que la expresión booleana que modela nuestro controlador está dada por $p \equiv c \vee h$, o bien, en términos de operaciones en el álgebra booleana $p = c + h$.

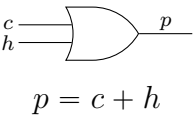
Para convertir la fórmula del ejemplo anterior en un circuito lógico real se utiliza la notación de compuertas, de las que hablaremos a continuación.

A las tres operaciones básicas del álgebra booleana les corresponden *compuertas lógicas*, que tienen, en general, una o más entradas y una única salida². Las tres compuertas básicas se representan como se muestra en la figura 3.1 a continuación.

Figura 3.1. Compuertas para las operaciones básicas del álgebra booleana



Para el ejemplo anterior, todo lo que necesitamos es una compuerta *or*, que queda como sigue:



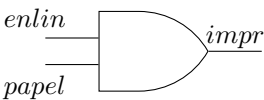
Veamos otros ejemplos sencillos.

Ejemplo 3.2. Una impresora va a imprimir sólo si la impresora está en línea y el sensor de papel dice que hay papel.

$\text{en-línea} \wedge \text{papel} = \text{imprimir}$

La tabla que corresponde a este proceso: Una compuerta *and* nos resuelve fácilmente el problema:

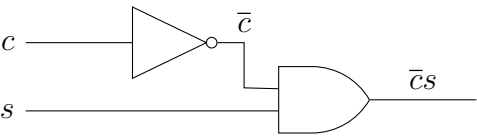
en línea	papel	imprimir
sí	sí	sí
sí	no	no
no	sí	no
no	no	no



Ejemplo 3.3. El calentador de gas de una casa está conectado a dos termostatos, uno en la sala y el otro donde está el calentador. Si la temperatura de la sala baja a 17° o menos, el calentador se prende. Pero si la temperatura en el cuarto del calentador sube a más de 40°, el calentador se apaga. Este proceso se representa aquel estado en el que la columna de *encender* en la tabla sea 1, como se muestra en la siguiente página.

²Muchos autores utilizan compuertas con k entradas, $k \geq 2$, y una salida, aprovechando la propiedad de asociatividad de \wedge y \vee .

cuarto $>40^\circ$	sala $<17^\circ$	encender
0	0	0
0	1	1
1	0	0
1	1	0



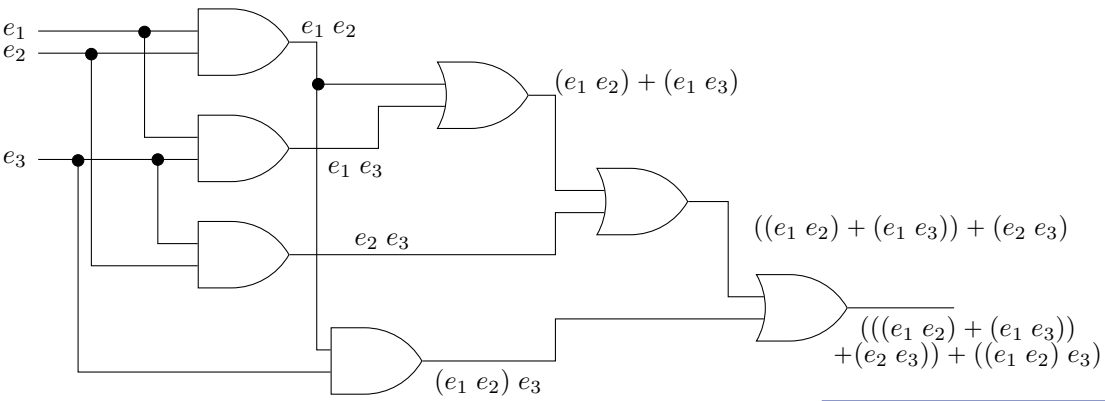
Ejemplo 3.4. Entre tres electores, la ley se aprueba si hay dos o más votando a favor. Mostramos en la tabla 3.1 a continuación los mintérminos correspondientes.

Tabla 3.1. Tabla correspondiente al ejemplo 3.4

elector 1	elector 2	elector 3	se aprueba
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

La fórmula representada en esta tabla es $(e_1e_2) + (e_1e_3) + (e_2e_3) + (e_1e_2e_3)$, cualquiera de los renglones 4, 6, 7 u 8. El el circuito correspondiente se encuentra en la figura 3.2.

Figura 3.2. Circuito correspondiente al problema de los electores



Cuando una línea muestra un punto quiere decir que se saca la señal directamente del cable después de alimentarla (o de la salida de alguna compuerta), para garantizar que todas son la misma señal, aunque al construir físicamente estos circuitos muchas veces se requiere un amplificador de la señal.

A continuación veremos dos ejemplos donde primero se construye un circuito de bloque, para después usar ese bloque construido para extender el alcance del circuito original. En este caso primero construiremos un circuito sumador de dos bits, para usar este circuito en la construcción de un sumador de dos números de tres bits.

Ejemplo 3.5. Queremos construir un sumador de dos números de un bit cada uno (dos bits). El sumador produce el valor de la suma y, en su caso, un acarreo. La suma de dos bits responde a la siguiente tabla, donde el valor de los bits se representa con b_1 y b_2 , como podemos observar en la tabla 3.2.

Tabla 3.2. Suma y acarreo de dos bits

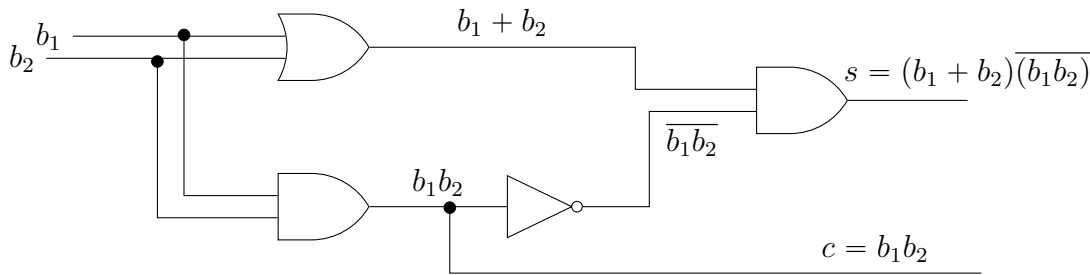
b_1	b_2	suma (s)	acarreo (c)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Es fácil construir un circuito lógico que produzca la suma y el acarreo de dos números de un solo bit, para implementar la tabla que acabamos de mostrar.

Estamos frente a un dispositivo con dos entradas y dos salidas. Las entradas las constituyen los bits a sumar (b_1 y b_2), mientras que las salidas son la suma (s) y el acarreo (c). En general, cuando estamos haciendo la suma de dos cantidades de k bits podemos esperar una salida con $k + 1$ bits, k de ellos las sumas correspondientes y el último bit que corresponderá al acarreo.

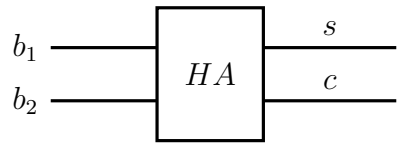
Si vemos la tabla notaremos que la suma es 1 cuando algunos de los dos bits es 1, pero no ambos (a esto se le conoce como el *o exclusivo* y se representa con \oplus). De lo anterior, la columna de la suma corresponde a la fórmula $s = (b_1 \wedge \neg b_2) \vee (\neg b_1 \wedge b_2)$, que corresponde a los estados segundo o tercero de la tabla. En la notación que estamos siguiendo para los circuitos digitales, con $+$ en lugar de \vee , la multiplicación implícita en lugar de \wedge y \bar{x} en lugar de $\neg x$, queda como $s = (b_1 \bar{b}_2) + (\bar{b}_1 b_2)$ –que es equivalente a $(b_1 + b_2)(\bar{b}_1 \bar{b}_2)$. En cuanto al acarreo, vemos que lo hay cuando ambos bits están en 1, o sea $b_1 \wedge b_2$ que corresponde a la expresión $c = b_1 b_2$. El circuito correspondiente se encuentra en la figura 3.3.

Figura 3.3. Circuito correspondiente a la suma de dos bits



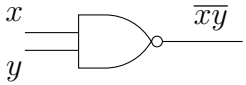
A este circuito se le conoce como semi sumador (*half adder*) y lo podemos representar en forma compacta como una caja que recibe dos entradas y produce dos salidas, como se muestra en la figura 3.4.

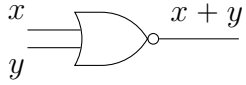
Figura 3.4. Diagrama de bloque de un semi sumador



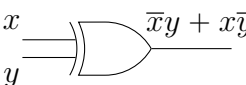
Vale la pena mencionar que muchos circuitos digitales se construyen directamente con compuertas *nand*, *nor*, *xor* o *xnor*, cuyas tablas y símbolos para las compuertas se encuentran a continuación.

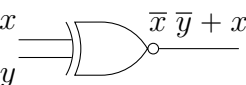
<i>nand</i>						<i>nor</i>		
<i>x</i>	<i>y</i>	\overline{xy}				<i>x</i>	<i>y</i>	$\overline{x + y}$
0	0	1				0	0	1
0	1	1				0	1	0
1	0	1				1	0	0
1	1	0				1	1	0





<i>xor</i>						<i>xnor</i>		
<i>x</i>	<i>y</i>	$\overline{xy} + x\overline{y}$				<i>x</i>	<i>y</i>	$\overline{\overline{xy} + x\overline{y}}$
0	0	0				0	0	1
0	1	1				0	1	0
1	0	1				1	0	0
1	1	0				1	1	1





Ejemplo 3.6. Queremos ahora construir un sumador de dos números de tres bits cada uno, usando lo que construimos para la suma de dos números de un bit cada uno. Sin embargo, hay que considerar que cuando se suman números de dos o más bits, se producen acarreo parciales en las sumas de cada dos bits, por lo que hay que tomar en cuenta este acarreo, que si bien es producido por un semi sumador, éste tiene sólo dos entradas y requerimos tres, cuando menos a partir de la segunda suma. Diseñaremos un circuito que tome tres entradas (dos bits y el acarreo) y produzca dos salidas, la suma y el acarreo. Un diseño inicial se encuentra en la tabla 3.3 en la siguiente página.

Como ya mencionamos, como nuestros semi sumadores sólo reciben dos entradas (no tres, como las que tenemos), tenemos que asociar y sumar los dos bits y obtener una suma parcial y un acarreo parcial; usamos la suma parcial para sumarla al acarreo anterior y obtener la suma final.

Tabla 3.3. Diseño inicial para un sumador con tres entradas y dos salidas

b_1	b_2	c_1	s_2	c_2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Para el acarreo, si hubo acarreo en la suma de los dos bits (1+1 provoca acarreo), ya no puede haber más acarreo al sumarle el acarreo parcial (1+1+1 acarrea sólo 1, igual que 1+1+0) y viceversa: si no hubo acarreo en la suma de los dos bits (0+0, 0+1, 1+0), podrá haber acarreo al sumar el acarreo parcial (0+1+1 y 1+0+1 provocan acarreo). La tabla para este sumador queda como sigue:

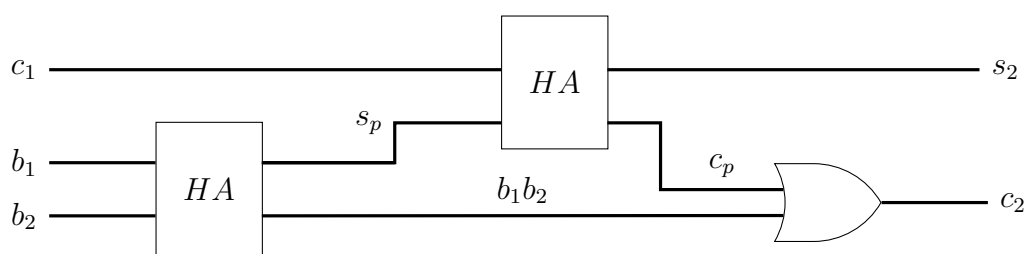
Tabla 3.4. Diseño final para un sumador con tres entradas y dos salidas

b_{12}	b_{22}	s_p	c_1	s_2	c_2
0	0	0	0	0	0
0	0	0	1	1	0
0	1	1	0	1	0
0	1	1	1	0	1
1	0	1	0	1	0
1	0	1	1	0	1
1	1	0	0	0	1
1	1	0	1	1	1

Los únicos renglones un poco más complicados son los dos últimos, pues tienen acarreo provocado por los dos bits, pero que no se refleja en la suma parcial, sino que hay que mandarlo al acarreo total de la suma. Hay que notar que cuando los dos bits a sumar están en 1, el acarreo se va a producir independientemente del valor del acarreo anterior; asimismo, el acarreo anterior ya no puede provocar un segundo acarreo, por lo que la suma de los dos bits produce directamente el segundo acarreo en este caso. El segundo acarreo también puede ser provocado, como se ve en la tabla, si la suma parcial es 1 y el acarreo anterior es 1, ($s_p c_1$). Este último resultado lo obtenemos si a un semi sumador le alimentamos

la suma parcial y el acarreo anterior. De esta discusión podemos diseñar nuestro circuito usando semi sumadores y una compuerta *or*, como se ve en la figura 3.5 en la página que sigue.

Figura 3.5. Implementación de un sumador completo usando semi sumadores

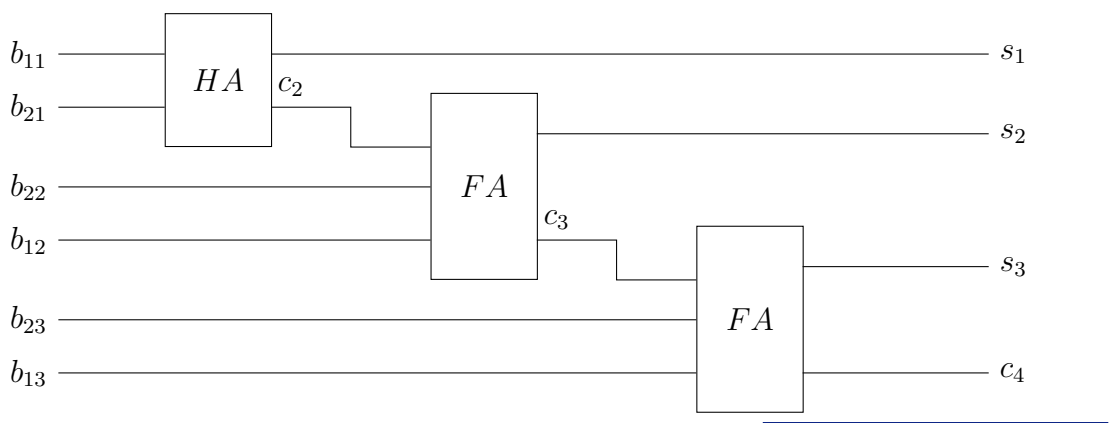


A la figura anterior es a lo que se conoce como un sumador completo (*full adder*) que recibe tres entradas y produce dos salidas, la suma y el acarreo.

Para cumplir con nuestro objetivo original, donde queremos sumar cantidades de tres bits, vamos a usar un semi sumador, ya que la posición 0 no tiene acarreo, y dos sumadores completos para las otras dos posiciones, quedando nuestro circuito como se muestra en la figura 3.6 y que corresponde a la siguiente suma:

$$\begin{array}{r} c_3 \quad c_2 \quad 0 \\ b_{13} \quad b_{12} \quad b_{11} \\ b_{23} \quad b_{22} \quad b_{21} \\ \hline c_4 \quad s_3 \quad s_2 \quad s_1 \end{array}$$

Figura 3.6. Sumador para cantidades de tres bits



3.1.1. Costo de los circuitos lógicos

En la construcción de circuitos lógicos el costo juega un papel importante, pues el dispositivo a construir, si no se hace con cuidado, puede resultar con un costo muy elevado.

El costo de un circuito lógico está relacionado con la cantidad de compuertas que se usan y el número de entradas para cada compuerta. Además, con la integración de circuitos y compuertas, existe además un umbral en el costo que en ocasiones se puede duplicar por el exceso de unas cuantas compuertas.

Se han implementado fórmulas usando compuertas de dos entradas, por lo que hablaremos ahora del costo de una fórmula. Si la fórmula se presenta en forma normal disyuntiva, podemos hablar del costo de cada mintérmino (conjunción) en la fórmula y el número de mintérminos en la misma. El costo de una fórmula f se obtiene mediante la siguiente relación:

$$C(f) = \sum_{j=0}^{k-1} P_j(f) + O(f)$$

donde

- k es el número de mintérminos en f ,
- $P_j(f)$ está relacionado con el número de literales en el j -ésimo mintérmino de f ,
- $O(f)$ está relacionado con el número de mintérminos en f .

Los costos de $P_j(f)$ y $O(f)$ están dados como sigue:

$$P_j(f) = \begin{cases} m & \text{si el } j\text{-ésimo mintérmino de } f \text{ tiene } m \text{ literales} \\ 0 & \text{si el } j\text{-ésimo mintérmino de } f \text{ tiene 1 literal} \end{cases}$$

$$O(f) = \begin{cases} m & \text{si } f \text{ tiene } m \text{ mintérminos} \\ 0 & \text{si } f \text{ tiene 1 mintérmino} \end{cases}$$

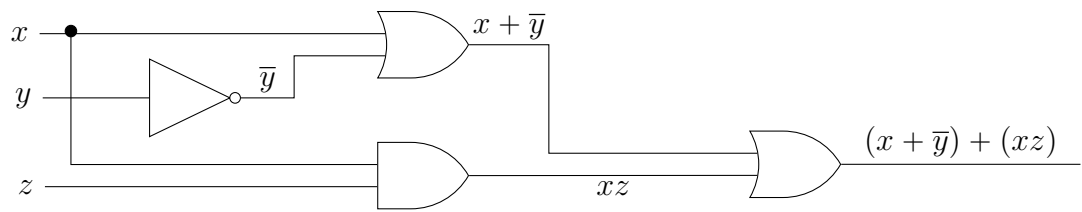
Por lo tanto, en el contexto de las ciencias de la computación es importante obtener una fórmula de costo mínimo, eliminando, en la medida de lo posible, conjunciones y literales de las fórmulas que se van a implementar con circuitos digitales.

3.1.2. Minimización de circuitos con equivalencias lógicas

En ocasiones la fórmula que obtenemos directamente de las tablas de verdad resultan ser muy grandes, pues explícitamente mencionan cada uno de los estados en el que la fórmula es verdadera. Desafortunadamente, si bien en lógica booleana esto únicamente se calificaría como “poco elegante”, en el caso de los circuitos lógicos en el que cada fórmula requiere de implementación física, esta redundancia cuesta dinero y espacio en una tableta. Si además consideramos que la diferencia entre una compuerta más para nuestro circuito puede significar la duplicación de los recursos requeridos por ya no caber en una única tableta, se entiende perfectamente la necesidad de denotar a la tabla de verdad con expresiones “mínimas” para reducir el número de compuertas y entradas requeridas.

Una manera natural de reducir el tamaño de las fórmulas es utilizando equivalencias lógicas. Veamos algunos ejemplos al respecto.

Ejemplo 3.7. Tenemos el siguiente circuito lógico para la fórmula $(x + \bar{y}) + (xz)$.



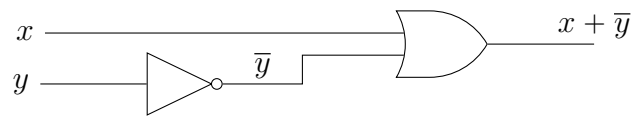
A este circuito le corresponde la tabla que se encuentra a continuación.

x	y	z	\bar{y}	$x + \bar{y}$	xz	$(x + \bar{y}) + (xz)$
0	0	0	1	1	0	1
0	0	1	1	1	0	1
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	1	0	1
1	0	1	1	1	1	1
1	1	0	0	1	0	1
1	1	1	0	1	1	1

Nos preguntamos si hay alguna expresión más sencilla para este circuito. Observando la tabla de verdad vemos que la fórmula se evalúa a 1 cuando \bar{y} es verdadera o cuando x es verdadera. Por lo tanto, podríamos “simplificar” esta fórmula para tener $x + \bar{y}$, que es mucho más sencilla y nos daría el mismo resultado –véase figura 3.7–.

Figura 3.7. Fórmula simplificada $x + \bar{y} = (x + \bar{y}) + xz$

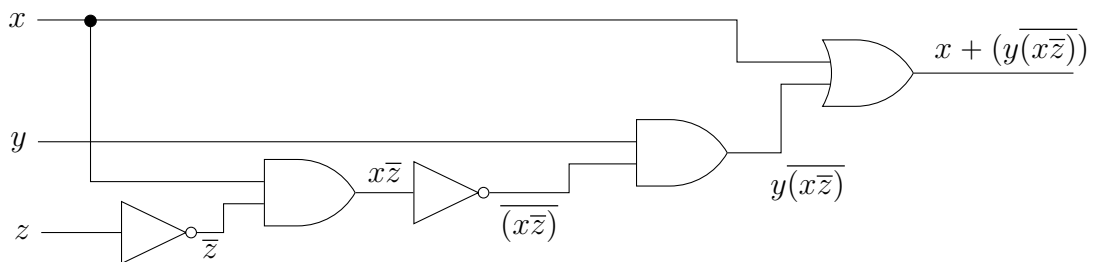
x	y	\bar{y}	$x + \bar{y}$
0	0	1	1
0	1	0	0
1	0	1	1
1	1	0	1



Podemos encontrar esta equivalencia utilizando derivaciones o lógica ecuacional:

$$\begin{aligned}
 (x + \bar{y}) + (xz) &\equiv (\bar{y} + x) + (xz) && \text{Conmutatividad} \\
 &\equiv \bar{y} + (x + (xz)) && \text{Asociatividad} \\
 &\equiv \bar{y} + x && \text{Eliminación de } \wedge \\
 &\equiv x + \bar{y}
 \end{aligned}$$

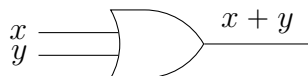
Ejemplo 3.8. Veamos el circuito que corresponde a la fórmula $x + (y(\overline{xz}))$.



Simplifiquemos esta fórmula:

$$\begin{aligned}
 x + (y(\overline{xz})) &\equiv x + (y(\bar{x} + z)) && \text{De Morgan} \\
 &\equiv (x + y)(x + (\bar{x} + z)) && \text{Distributividad} \\
 &\equiv (x + y)((x + \bar{x}) + z) && \text{Asociatividad} \\
 &\equiv (x + y)(\text{true} + z) && \text{Tercero excluido} \\
 &\equiv (x + y)(\text{true}) && \text{Dominancia de la suma} \\
 &\equiv (x + y) && \text{Identidad del producto}
 \end{aligned}$$

lo que deja un circuito mucho más económico y sencillo que el anterior:



El problema con este método es que tenemos que tener una idea de a dónde queremos llegar (los dos lados de la equivalencia), lo que cuando queremos simplificar fórmulas no siempre es claro. Para ello tenemos otros mecanismos para simplificar circuitos digitales que veremos a continuación.

Ejercicios

3.1.1.- Usando las propiedades del álgebra booleana, simplifique las siguientes fórmulas:

a) $xyz + x\bar{y}z$

c) $xy\bar{z} + x\bar{y}\bar{z}$

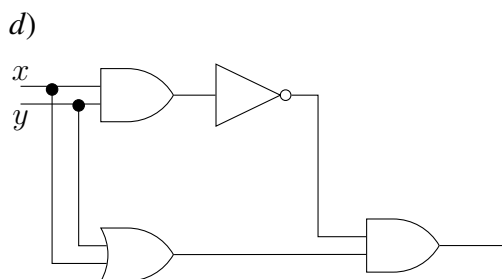
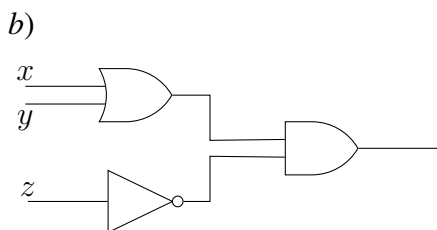
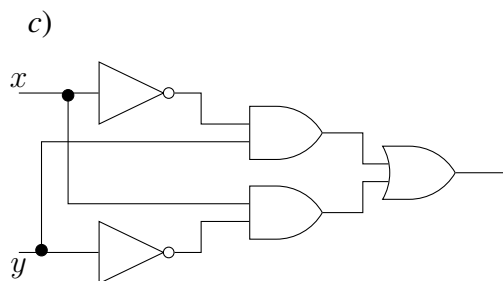
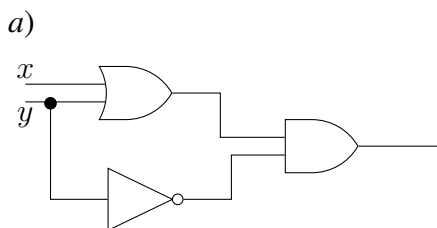
b) $xy + x\bar{y}$

d) $\bar{x} + \bar{y} + xyz$

3.1.2.- Dada la siguiente descripción, construya la tabla de verdad que representa a esa función y el circuito digital correspondiente.

- Un circuito que implemente votación mayoritaria de cinco individuos.
- Un circuito que compare dos números binarios de dos bits cada uno $-(x_1, x_0), (y_1, y_0)-$ y regrese 0 si el segundo es mayor que el primero y 1 si el primero es mayor que el segundo.
- Un circuito que diga que una torta de jamón puede tener queso pero una torta de huevo no.
- Los refrescos que hay son de tres sabores posibles: cola, fresa o naranja. Tienen gas si son de cola o fresa y no tienen gas si son de fresa o naranja. Un refresco no puede ser más que de un sabor.

3.1.3.- Describa la fórmula que corresponde a los siguientes circuitos:



3.1.4.- Construya los circuitos digitales con compuertas *AND*, *NOT* y *OR* que correspondan a las siguientes fórmulas:

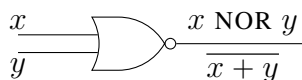
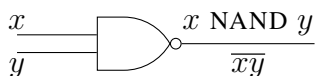
a) $xyz + \bar{x}\bar{y}\bar{z}$

c) $(x + \bar{y} + z)(\bar{x} + y + \bar{z})$

b) $(x + y)\bar{x}\bar{y}$

d) $((xy) + (\bar{x}\bar{y}))\bar{z}$

3.1.5.- Otras dos compuertas disponibles son la compuerta *NAND* y la compuerta *NOR* representadas por los siguientes diagramas:



Para las siguientes fórmulas:

a) \bar{x} b) $x + y$ c) xy d) $x \oplus y$

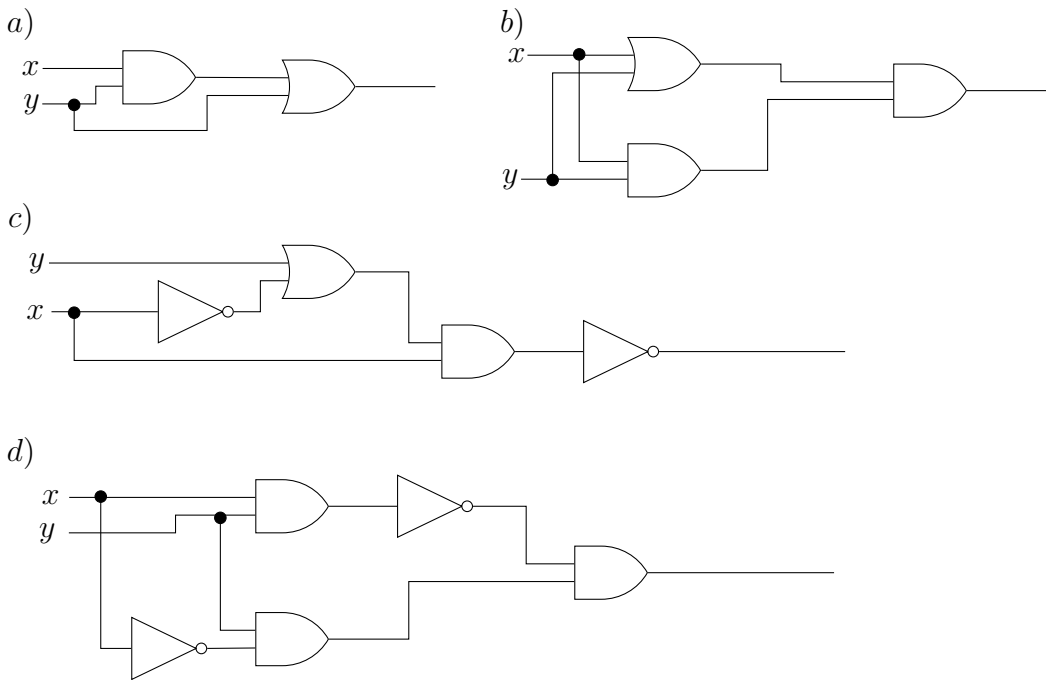
construya los circuitos utilizando únicamente

- i. Compuertas NAND.
- ii. Compuertas NOR.

3.1.6.- Construya un semi restador (*half subtracter*). Un semi restador tiene como entrada dos bits y produce como salida un bit para la diferencia y un bit para el acarreo.

3.1.7.- Construya un circuito para un restador completo (*full subtracter*). Un restador completo tiene como entrada dos bits y un bit para el acarreo y produce como salida un bit para la diferencia y un bit para el acarreo.

3.1.8.- Escriba las expresiones booleanas asociadas a los siguientes circuitos:



3.1.9.- Dibuje un circuito que represente a cada una de las siguientes expresiones booleanas.

a) $(xy) + (\bar{x} + y)$

c) $(\bar{y}z) + \overline{((w\bar{x})\bar{y})}$

b) $\bar{x}y + (x(yz))$

d) $(x(yz))((\bar{x}\bar{y}) + (z\bar{w}))$

3.1.10.- Para expresiones booleanas en la siguiente página, dé el resultado que se presenta en el estado dado:

- a) $(x + y)(\bar{x} + z)$; estado: $(x = 1, y = 1, z = 0)$.
 b) $((xy) + z)(x + (\bar{y}z))$; estado: $(x = 0, y = 1, z = 1)$.
 c) $\overline{x(yz)}$; estado: $(x = 0, y = 1, z = 0)$.
 d) $(x(y + \bar{z})(\bar{x} + \bar{z}))$; estado: $(x = 0, y = 1, z = 1)$.

3.1.11.- Construya una tabla de verdad para las siguientes expresiones booleanas:

- a) $x(y + \bar{x})$ c) $(x + \bar{y}) + (x\bar{z})$
 b) $\overline{x + \bar{y}} + x$ d) $((xy)z) + (x(y\bar{z}))$

3.1.12.- La luz de una escalera se controla con tres apagadores, uno fuera de la casa, uno al pie de la escalera y otro al final de la escalera. Debe ser posible prender o apagar la luz desde cualquiera de estos apagadores, sin importar cuál es la posición de cualquiera de los ellos. Diseñe un circuito que haga esto posible. (Pista: siempre que el número de apagadores prendidos sea impar, la luz estará prendida.)

3.1.13.- Sea $A =_{def} x \cdot y + \overline{x \cdot z} \cdot \bar{y}$.

- a) Dibuje el circuito lógico correspondiente a A .
 b) Transforme A a la forma normal disyuntiva.

3.1.14.- El sistema de alarma de una casa funciona de la siguiente manera: si la alarma está encendida entonces sonará si alguna de las dos puertas de la casa, la de enfrente o la trasera, se abren (esto incluye, por supuesto, al caso en que las dos se abren). Por otro lado, si la alarma está apagada entonces no sonará.

- a) Escriba una especificación formal del funcionamiento de la alarma mediante una fórmula lógica.
 b) Diseñe un circuito lógico que corresponda al sistema de alarma.

3.1.15.- El club de Tobi tiene cuatro integrantes y se rige, para tomar sus decisiones, como sigue: cada miembro puede dar un voto, excepto Tobi cuyo voto cuenta doble. Se propone que el dinero ganado por la venta de limonada se ocupe en comprar deliciosos chocolates. Esta propuesta se aprueba por mayoría simple, es decir, si hay al menos tres votos a favor. Se desea diseñar un circuito que implemente el mecanismo de votación, encendiendo el farol de la casa del árbol cuando la propuesta es aprobada.

- a) Elabore la tabla de verdad correspondiente al circuito.
 b) A partir de la tabla, construya la fórmula correspondiente mediante la forma normal disyuntiva.
 c) Dibuje el circuito correspondiente.

3.1.16.- El restaurante *El Pollo Farsante* está de aniversario y ofrece los siguientes platillos especiales: ensalada de salicornia, venado en salsa de ciruela, pato al Oporto y strudel de frutos del bosque. Adicionalmente, se regala una botella reserva especial de vino *Santa Clota* en cualquier orden que incluya la ensalada, el pato y el strudel; o bien si se pide venado. Se desea implementar un circuito que haga sonar un timbre en la cava cuando una orden recibe el vino de cortesía.

- a) Elabore la tabla de verdad correspondiente al circuito.
- b) Construya la forma normal disyuntiva correspondiente.

3.1.17.- Se desea implementar un circuito lógico que reconozca cuándo un número natural entre 0 y 15 es primo.

- a) Elabore la tabla de verdad correspondiente al circuito.
- b) Construya la forma normal disyuntiva correspondiente.

Sugerencia: los números entre 0 y 15 en su representación binaria corresponden a los estados de una función booleana de 4 variables. Por ejemplo, el 0 corresponde al estado 0000, el 15 al estado 1111 y el 6 al estado 0110.

3.2. Mapas de Karnaugh

La manera como hemos diseñado los circuitos lógicos hasta ahora ha sido, fundamentalmente, partiendo de la tabla de verdad. También hemos usado algunas equivalencias lógicas para simplificar estos circuitos. Sin embargo, esta manera de reducir las fórmulas no es mecanizable ni algorítmica, por lo que no nos garantiza que lleguemos a una buena solución. En 1953 Maurice Karnaugh introdujo un método gráfico para minimizar funciones booleanas (circuitos) con hasta seis variables de entrada, expresadas en forma normal disyuntiva, que hoy se conoce como el método de *mapas de Karnaugh*. Empezaremos a trabajar con fórmulas de dos variables, para extender después a tres y cuatro variables. Los casos para cinco y seis variables son más complicados y en su lugar suelen usarse otros métodos, por lo que no los discutimos aquí.

3.2.1. Mapas de Karnaugh de dos variables

Si tenemos dos variables, digamos x e y , tenemos, como ya vimos, cuatro posibles *mintérminos*: xy , $\bar{x}y$, $x\bar{y}$ y $\bar{x}\bar{y}$. La fórmula para representar un circuito que tenga dos variables utilizará aquellos *mintérminos* que sean 1 en la tabla de verdad. El mapa de Karnaugh de dos variables consiste de una cuadrícula de dos por dos, donde cada columna está etiquetada con una de las variables y su negación, y cada renglón con la otra variable y su

negación; la celda que corresponde al renglón x columna y –posición (x, y) – va a contener un 1 si el mintermino xy aparece en la fórmula.

	y	\bar{y}
x	xy	$x\bar{y}$
\bar{x}	$\bar{x}y$	$\bar{x}\bar{y}$

En cada una de las celdas va a aparecer un 1 si ese mintermino aparece en la expresión y nada si no aparece. Veamos algunos ejemplos en lo que sigue.

	y	\bar{y}
x	1	
\bar{x}	1	

$xy + \bar{x}y$

	y	\bar{y}
x		1
\bar{x}	1	

$\bar{x}y + x\bar{y}$

	y	\bar{y}
x		1
\bar{x}	1	1

$x\bar{y} + \bar{x}y + \bar{x}\bar{y}$

Siempre que se encuentran dos minterminos podemos combinar y eliminar a la variable que aparece también negada:

$$xy + \bar{x}y = (x + \bar{x})y = (\text{true})y = y$$

Gráficamente, esto se hace enmarcando aquellas celdas contiguas (vertical u horizontalmente). Un recuadro en una columna o renglón indica la eliminación de la variable que aparece tal cual y negada en esa columna o renglón. En los tres ejemplos que acabamos de dar, la simplificación se logra de la siguiente manera:

a)

	y	\bar{y}
x	1	
\bar{x}	1	

y

b)

	y	\bar{y}
x		1
\bar{x}	1	

$\bar{x}y + x\bar{y}$

c)

	y	\bar{y}
x		1
\bar{x}	1	1

$\bar{x} + \bar{y}$

Si procedemos a simplificar usando equivalencias lógicas, obtenemos las siguientes secuencias:

a) Para este mapa tenemos la siguiente sucesión de pasos:

$$\begin{aligned}
 xy + \bar{x}y &= (x + \bar{x})y && \text{Distributividad} \\
 &= (\text{true})y && \text{Tercero excluido} \\
 &= y && \text{Identidad del producto}
 \end{aligned}$$

b) Para el segundo mapa no podemos hacer nada gráficamente y tampoco algebraicamente.

c) Para el tercer mapa tenemos los siguientes pasos:

$$\begin{aligned}
 x\bar{y} + \bar{x}y + \bar{x}\bar{y} &= x\bar{y} + \bar{x}\bar{y} + \bar{x}y + \bar{x}\bar{y} && \text{Usamos idempotencia para repetir } \bar{x}\bar{y} \\
 &&& \text{y después conmutatividad y asociatividad} \\
 &= (\bar{x} + x)\bar{y} + \bar{x}(y + \bar{y}) && \text{Distributividad} \\
 &= (\text{true})\bar{y} + \bar{x}(\text{true}) && \text{Tercero excluido} \\
 &= \bar{y} + \bar{x} && \text{Identidad del producto} \\
 &= \bar{x} + \bar{y} && \text{Conmutatividad}
 \end{aligned}$$

Como se ve, el proceso de minimización consiste en armar bloques de unos (1) mediante sectores adyacentes, donde cada bloque tiene un número de celdas que es potencia de 2. La fórmula minimizada se obtiene a partir de dichos bloques, puesto que un bloque genera fórmulas donde se elimina una variable. Con respecto a los ejemplos anteriores, en el primer caso hay un solo bloque que corresponde a la primera columna, mientras que en el tercero hay dos bloques, uno de ellos formado por la segunda columna y el otro por el segundo renglón (por esto la fórmula mínima tiene dos sumandos). En el segundo caso no hay bloques, pues no hay unos adyacentes, por lo que la fórmula ya era mínima. Más adelante describiremos cómo construir los bloques y cómo, a partir de ellos, obtener la fórmula mínima y por lo tanto el circuito digital mínimo. Pero antes describimos el método de mapas para tres y cuatro variables.

3.2.2. Mapas de Karnaugh con tres y cuatro variables

Como pudieron observar, cada celda en un mapa de Karnaugh representa a un estado para evaluar la fórmula. En el caso de dos variables el número de celdas es de $2^2 = 4$; si tenemos tres variables tenemos que acomodar $2^3 = 8$ celdas, $2^4 = 16$ celdas para cuatro variables; y así sucesivamente. La regla para armar los mapas de Karnaugh es que el cambio entre dos celdas consecutivas, ya sea horizontales o verticales, no puede ser más que de una variable.

Veamos el caso de un mapa de Karnaugh para tres variables, x , y y z . Tenemos que representar en las celdas del mapa todos los minterminos de estas tres variables: $xyz, xy\bar{z}, x\bar{y}z$, etcétera. Si vemos a las variables negadas como 0 y a las no negadas como 1, tendríamos que tener todos los estados que van del 000 al 111. Podemos acomodarlos en una cuadrícula de dos renglones por cuatro columnas como se muestra en la figura 3.8 de la siguiente página.

Si observamos el mapa de Karnaugh anterior, en el que se colocó momentáneamente en cada celda el mintermino que le corresponde, podemos verificar que dos celdas contiguas, ya sea vertical u horizontalmente, sólo tienen un dígito de diferencia. Por ejemplo, a la izquierda de $xy\bar{z}$ (110) se encuentra la celda xyz (111) que únicamente tiene distinto el

Figura 3.8. Mapa de Karnaugh con tres variables

	yz	$y\bar{z}$	$\bar{y}z$	$\bar{y}\bar{z}$	
x	111 xyz	110 $xy\bar{z}$	100 $x\bar{y}\bar{z}$	101 $x\bar{y}z$	(1)
\bar{x}	011 $\bar{x}yz$	010 $\bar{x}y\bar{z}$	000 $\bar{x}\bar{y}\bar{z}$	001 $\bar{x}\bar{y}z$	(0)
	(11)	(10)	(00)	(01)	

último dígito, mientras que abajo de ésta se encuentra el mintérmino $\bar{x}y\bar{z}$ (010) que se distingue de la anterior sólo por la primera posición. Esta distribución de los mintérminos es importante, pues cuando queramos cubrir celdas contiguas, tenemos que garantizar que una de ellas tiene a alguna de las variables en 0 mientras que la otra tiene a esa misma variable en 1, siendo ésa la única diferencia.

Ejemplo 3.9. Por ejemplo, si quisiéramos expresar la fórmula $xyz + \bar{x}yz + x\bar{y}z + \bar{x}\bar{y}z$ colocaríamos un 1 en la celda correspondiente a la expresión, por lo que el Mapa de Karnaugh correspondiente a esta fórmula quedaría como se ve en la figura 3.9.

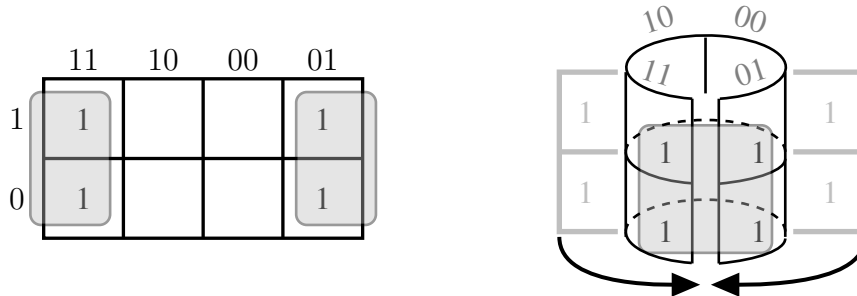
Figura 3.9. Mapa de Karnaugh para $xyz + \bar{x}yz + x\bar{y}z + \bar{x}\bar{y}z$

	yz	$y\bar{z}$	$\bar{y}z$	$\bar{y}\bar{z}$	
x	1 xyz			1 $x\bar{y}z$	(1)
\bar{x}	1 $\bar{x}yz$			1 $\bar{x}\bar{y}z$	(0)
	(11)	(10)	(00)	(01)	

	yz	$y\bar{z}$	$\bar{y}z$	$\bar{y}\bar{z}$	
x	1			1	(1)
\bar{x}	1			1	(0)
	(11)	(10)	(00)	(01)	

Obsérvese que en esta figura las celdas de los extremos en cada renglón también son adyacentes, puesto que sólo cambia el valor de una de sus variables. Para recordar esta adyacencia puede pensarse que el mapa de Karnaugh se enrolla por los extremos de los renglones como si fuera un cilindro, como se puede observar en la figura 3.10.

Figura 3.10. Adyacencia de última y primera columna para $xyz + \bar{x}yz + x\bar{y}z + \bar{x}\bar{y}z$



Al tomar como un solo bloque a la primera y última columna, como aparecen x y \bar{x} , así como y y \bar{y} , la única variable que aparece únicamente en positivo es z , por lo que la fórmula se minimiza y tenemos: $xyz + \bar{x}yz + x\bar{y}z + \bar{x}\bar{y}z = z$.

Ejemplo 3.10. El mapa de Karnaugh para la fórmula $xyz + xy\bar{z} + x\bar{y}\bar{z} + x\bar{y}z + \bar{x}yz + \bar{x}y\bar{z}$, queda como se ve a continuación (observemos que ahora puede haber bloques más grandes, por ejemplo formados por cuatro sectores):

	yz	$y\bar{z}$	$\bar{y}\bar{z}$	$\bar{y}z$
x	1	1	1	1
\bar{x}	1	1		

$= x + y$

El término x resulta de todo el renglón superior, ya que va a estar multiplicando a $(y + \bar{y})$ y $(z + \bar{z})$ que al distribuirlo queda sólo el término x .

$$\begin{aligned}
 xyz + xy\bar{z} + x\bar{y}\bar{z} + x\bar{y}z &= x(yz + y\bar{z} + \bar{y}\bar{z} + \bar{y}z) && \text{Distributividad} \\
 &= x(y(z + \bar{z}) + \bar{y}(z + \bar{z})) && \text{Distributividad} \\
 &= x(y(\text{true}) + \bar{y}(\text{true})) && \text{Tercero excluido} \\
 &= x(y + \bar{y}) && \text{Identidad del producto} \\
 &= x(\text{true}) && \text{Tercero excluido} \\
 &= x && \text{Identidad del producto}
 \end{aligned}$$

El término y resulta del recuadro de la izquierda. Se observa que estamos usando dos mintérminos ya usados para obtener el término x , a saber, xyz y $xy\bar{z}$. Estos mintérminos pueden usarse nuevamente debido a la ley de idempotencia de la lógica que nos dice que $w + w = w$ para cualquier expresión w .

$$\begin{aligned}
 xyz + xy\bar{z} + \bar{x}yz + \bar{x}y\bar{z} &= x(yz + y\bar{z}) + \bar{x}(yz + y\bar{z}) && \text{Distributividad} \\
 &= x(y(z + \bar{z})) + \bar{x}(y(z + \bar{z})) && \text{Distributividad} \\
 &= x(y(\text{true})) + \bar{x}(y(\text{true})) && \text{Tercero excluido} \\
 &= xy + \bar{x}y && \text{Identidad del producto} \\
 &= (x + \bar{x})y && \text{Distributividad} \\
 &= (\text{true})y && \text{Tercero excluido} \\
 &= y && \text{Identidad del producto}
 \end{aligned}$$

Como se puede ver, gráficamente se minimizó la expresión con mucho menos trabajo. Una expresión que tenía seis mintérminos, cada uno de ellos involucrando a tres variables (dos compuertas por mintérmino) se redujo a dos variables y una sola compuerta.

Mapas de Karnaugh de cuatro variables

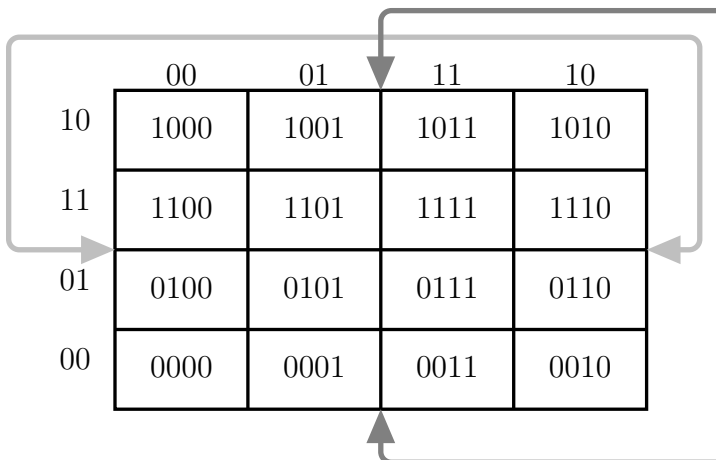
En el caso de cuatro variables tenemos 16 posibles estados ($2^4 = 16$), por lo que acomodaremos el mapa de Karnaugh en una cuadrícula de 4×4 , cuidando que entre celdas contiguas, ya sean verticales u horizontales, cambie exactamente una variable de positiva a negada o viceversa. El modelo para mapas de 4×4 se da en la figura 3.11.

Figura 3.11. Mapa de Karnaugh para cuatro variables

	$\bar{y}\bar{z} (00)$	$\bar{y}z (01)$	$yz (11)$	$y\bar{z} (10)$
$w\bar{x} (10)$	$w\bar{x}\bar{y}\bar{z}$	$w\bar{x}\bar{y}z$	$w\bar{x}yz$	$w\bar{x}y\bar{z}$
$wx (11)$	$wx\bar{y}\bar{z}$	$wx\bar{y}z$	$wxyz$	$wxy\bar{z}$
$\bar{w}x (01)$	$\bar{w}x\bar{y}\bar{z}$	$\bar{w}x\bar{y}z$	$\bar{w}xyz$	$\bar{w}xy\bar{z}$
$\bar{w}\bar{x} (00)$	$\bar{w}\bar{x}\bar{y}\bar{z}$	$\bar{w}\bar{x}\bar{y}z$	$\bar{w}\bar{x}yz$	$\bar{w}\bar{x}y\bar{z}$

Como en el caso de tres variables, los extremos de cada renglón son adyacentes, pero también lo son los extremos de cada columna. Para recordar esto puede pensarse que los lados izquierdo y derecho del mapa están pegados, formando un cilindro nuevamente, pero además las tapas de dicho cilindro también están pegadas, formando una dona (a este objeto geométrico se le llama toro o toroide). En la figura 3.12 podemos ver la dirección en que se doblan y juntan los extremos.

Figura 3.12. Mapas de Karnaugh de cuatro variables y su adyacencia



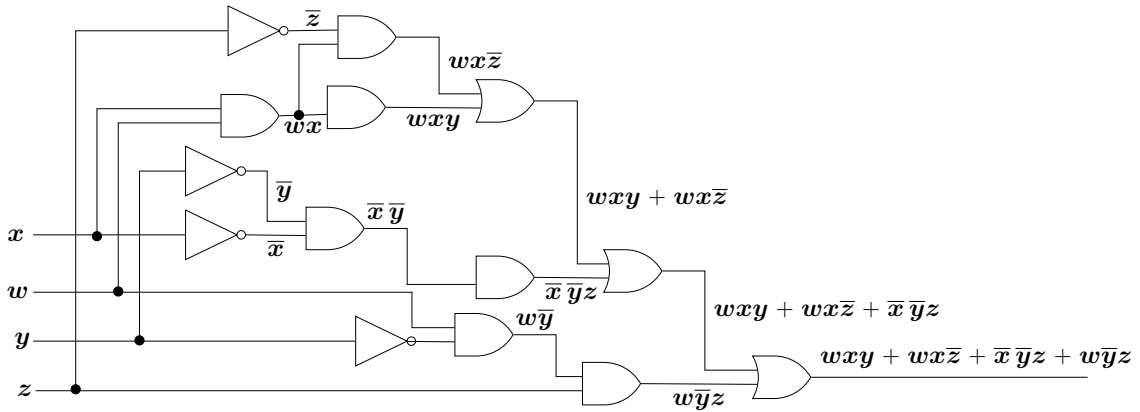
Como se puede observar en la figura, si tomamos el último y primer renglón, quedando como se muestra en la siguiente página,

00	0000	0001	0011	0010
10	1000	1001	1011	1010

y comparamos columna por columna ambos renglones, del último renglón al primero, en cada columna, únicamente cambia el valor de una variable. Por ejemplo, en el último renglón del mapa y segunda columna tenemos el valor 0 0 0 1, mientras que en el primer renglón del mapa y segunda columna tenemos 1 0 0 1, con lo que únicamente cambia la variable w (la primera) de 1 a 0.

Ejemplo 3.11. Supongamos que queremos simplificar la expresión $wxy + wx\bar{z} + \bar{x}\bar{y}z + w\bar{y}z$, cuyo circuito digital se encuentra en la figura 3.13.

Figura 3.13. Circuito correspondiente a $wxy + wx\bar{z} + \bar{x}\bar{y}z + w\bar{y}z$



Puesto que los sectores del mapa de Karnaugh involucran a las cuatro variables, aquellas variables que no aparecen en un mintermino las agregamos multiplicando por una expresión de la forma $v + \bar{v}$, lo cual no afecta al resultado por la propiedad de identidad del producto. Por ejemplo el mintermino $wx\bar{z}$ se transforma en la expresión $wx(y + \bar{y})\bar{z} = wxyz + wx\bar{y}z$. Con esta idea, procedemos a construir el mapa de Karnaugh correspondiente al circuito anterior:

$$\begin{aligned}
 wxy + wx\bar{z} + \bar{x}\bar{y} + w\bar{y}z &= wxy(z + \bar{z}) + wx(y + \bar{y})\bar{z} + (w + \bar{w})\bar{x}\bar{y}z + w(x + \bar{x})\bar{y}z \\
 &= wxyz + wxy\bar{z} + wxy\bar{z} + wx\bar{y}\bar{z} + w\bar{x}\bar{y}z + \bar{w}\bar{x}\bar{y}z + w\bar{x}\bar{y}z + w\bar{x}\bar{y}z \\
 &= wxyz + wxy\bar{z} + wxy\bar{z} + w\bar{x}\bar{y}z + \bar{w}\bar{x}\bar{y}z + w\bar{x}\bar{y}z \quad (3.1)
 \end{aligned}$$

El mapa de Karnaugh que corresponde a esta fórmula se encuentra en la figura 3.14, en la siguiente página.

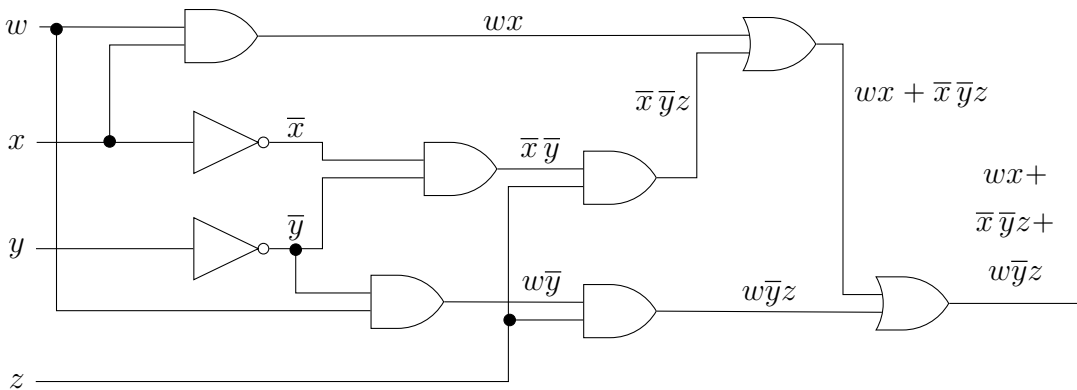
El circuito digital queda como se muestra en la siguiente página, en la figura 3.15. Vale la pena notar que en lugar de 14 compuertas se utilizaron 9, que representa un 36 % de reducción en el número de compuertas.

Figura 3.14. Mapa de Karnaugh correspondiente a la fórmula (3.1)

	$\bar{y}\bar{z}$	$\bar{y}z$	yz	$y\bar{z}$
$w\bar{x}$		1		
wx	1	1	1	1
$\bar{w}x$				
$\bar{w}\bar{x}$		1		

$$\begin{aligned}
 wxyz + wxy\bar{z} + wx\bar{y}\bar{z} + w\bar{x}yz + \bar{w}\bar{x}yz + wx\bar{y}z = \\
 = wx + \bar{x}\bar{y}z + w\bar{y}z
 \end{aligned}$$

Figura 3.15. Circuito digital correspondiente al Mapa de Karnaugh en la figura 3.14



El ejemplo anterior deja ver un caso de cómo obtener la fórmula y el circuito mínimo a partir de la formación de bloques. Sin embargo, no hemos descrito un proceso general para la minimización, cosa que hacemos a continuación.

3.2.3. Construcción del circuito digital mínimo

Para obtener un circuito digital mínimo a partir de una fórmula o circuito dados se debe resolver el siguiente problema:

Dada una fórmula A , construir una fórmula lógicamente equivalente a A , cuya forma normal disyuntiva utilice el menor número de variables y mintérminos posible. Dicha fórmula permitirá construir un circuito lógico con el menor número de compuertas y variables, el que se denotará con A^{Min} .

La solución a este problema se hace mediante los mapas de Karnaugh como sigue: Partiendo de la forma normal disyuntiva de la fórmula A se construye el mapa de Karnaugh, colocando un 1 en los sectores correspondientes a los mintérminos de A . Para obtener la minimización vamos a agrupar en bloques a los sectores marcados con 1 que además sean adyacentes en el mapa.

Para la construcción de bloques es importante atender las siguientes observaciones:

- Los bloques sólo constan de sectores marcados con 1 y cada 1 debe figurar en al menos un bloque.
- No es válido agrupar en diagonal, puesto que dos casillas en diagonal nunca son adyacentes (cambian el valor de más de una variable).
- Los bloques sólo pueden ser de tamaño 2^k , en nuestro caso un bloque puede ser de tamaño 1, 2, 4, 8 o 16.
- Un bloque debe cubrir, sin ambigüedad, una región determinada de renglones y columnas por lo que sólo puede ser rectangular (el cuadrado claramente es un rectángulo). En particular no puede haber bloques en forma de L .
- Los bloques pueden traslaparse (esto equivale a usar idempotencia para utilizar el mismo mintérmino más de una vez).
- Se deben agrupar todos los unos en el menor número posible de bloques. Para esto es útil seguir la siguiente heurística:
 1. Formar los bloques aislados de tamaño 1.
 2. Formar los bloques de tamaño 2, de manera única si es posible.
 3. Formar los bloques de tamaño 4, de manera única si es posible.
 4. Formar los bloques de tamaño 8, de manera única si es posible.
 5. Si aún quedan sectores marcados con 1 fuera de los bloques, formar nuevos bloques traslapados de la manera más eficiente posible.

Una vez contruidos los bloques, se procede a construir la fórmula minimizada A^{Min} de la siguiente manera:

- Dado un bloque B_i en el mapa de Karnaugh definimos un mintérmino β_i a partir de él.
- Si un bloque B_i es de tamaño 2^k entonces en el mintérmino correspondiente β_i se eliminarán k de las n variables de la fórmula, donde $n = 2, 3, 4$. Es decir, en β_i sólo aparecerán $n - k$ variables.
- La fórmula minimizada A^{Min} será la suma de los mintérminos generados por los bloques, es decir, si hay m bloques B_1, \dots, B_m en el mapa, entonces

$$A^{Min} =_{def} \beta_1 + \dots + \beta_m$$

- El mintérmino β asociado al bloque B se define como $\beta = rc$, donde c es una expresión obtenida de las columnas del bloque y r es una expresión obtenida de los renglones del bloque. Dichas expresiones se determinan como sigue:

- r es la expresión que permanece constante en todos los renglones de B , en particular si B está contenido totalmente en un renglón, r es la etiqueta de dicho renglón. Además si B cubre todos los renglones del mapa entonces $r = 1$.
- c es la expresión que permanece constante en todas las columnas de B , en particular si B está contenido totalmente en una columna, c es la etiqueta de dicha columna. Además si B cubre todas las columnas del mapa entonces $c = 1$.

Para ejemplificar este proceso tomemos nuevamente el mapa correspondiente a la fórmula $A = wxy + wx\bar{z} + \bar{x}\bar{y}z + w\bar{y}z$, que se encuentra en la figura 3.16.

Tenemos tres bloques B_1, B_2 y B_3 , cuyos mintérminos asociados se construyen como sigue:

- El bloque B_1 está formado por los sectores $w\bar{x}\bar{y}z$ y $wx\bar{y}z$. Como la expresión que permanece constante en los renglones de B_1 es w , se tiene $r_1 = w$. Dado que B_1 está contenido en la columna $\bar{y}z$, entonces $c_1 = \bar{y}z$. Por lo tanto $\beta_1 = r_1c_1 = w\bar{y}z$.

Figura 3.16. Mapa de Karnaugh para $A = wxy + wx\bar{z} + \bar{x}\bar{y}z + w\bar{y}z$

	$\bar{y}\bar{z}$	$\bar{y}z$	yz	$y\bar{z}$
$w\bar{x}$		1		
wx	1	1	1	1
$\bar{w}\bar{x}$				
$\bar{w}x$		1		

- El bloque B_2 está formado por el renglón wx . Dado que B_2 está contenido en el renglón wx , entonces $r_2 = wx$. Además, como B_2 abarca todas las columnas del mapa, $c_2 = 1$. Por lo tanto $\beta_2 = r_2c_2 = wx1 = wx$.
- El bloque B_3 está formado por los sectores $w\bar{x}\bar{y}z$ y $\bar{w}\bar{x}\bar{y}z$. Puesto que la expresión que permanece constante en los renglones de B_3 es \bar{x} , entonces $r_3 = \bar{x}$. Además, como B_3 está contenido en la columna $\bar{y}z$, se tiene $c_3 = \bar{y}z$. Por lo tanto

$$\beta_3 = r_3c_3 = \bar{x}\bar{y}z.$$

De manera que el circuito mínimo corresponde a la fórmula

$$A^{Min} = \beta_1 + \beta_2 + \beta_3 = w\bar{y}z + wx + \bar{x}\bar{y}z$$

que por supuesto coincide con el resultado obtenido anteriormente.

Ejercicios

3.2.1.- Encuentre las sumas de productos representados por los siguientes mapas de Karnaugh de dos variables (sin minimizar):

a)	y	\bar{y}	b)	y	\bar{y}	c)	y	\bar{y}	d)	y	\bar{y}
x	1		x	1	1	x	1		x	1	1
\bar{x}	1	1	\bar{x}			\bar{x}		1	\bar{x}	1	1

3.2.2.- Encuentre las sumas de productos representados por los siguientes Mapas de Karnaugh de tres variables (sin minimizar):

a)	yz	$y\bar{z}$	$\bar{y}z$	$\bar{y}\bar{z}$	b)	yz	$y\bar{z}$	$\bar{y}z$	$\bar{y}\bar{z}$
x	1		1		x	1	1		
\bar{x}		1		1	\bar{x}	1		1	1

c)	yz	$y\bar{z}$	$\bar{y}z$	$\bar{y}\bar{z}$	d)	yz	$y\bar{z}$	$\bar{y}z$	$\bar{y}\bar{z}$
x	1	1	1	1	x		1	1	
\bar{x}	1			1	\bar{x}			1	1

3.2.3.- Minimice los Mapas de Karnaugh de los ejercicios (3.2.1) y (3.2.2).

3.2.4.- Use mapas de Karnaugh para minimizar las siguientes funciones de dos variables:

- $\bar{x}y + \bar{x}\bar{y}$
- $xy + x\bar{y}$
- $x\bar{y} + xy + \bar{x}y$
- $\bar{x}\bar{y} + \bar{x}y + xy$

3.2.5.- Usando mapas de Karnaugh minimice las siguientes funciones de tres variables. Dibuje los circuitos correspondientes a las funciones originales y a las funciones minimizadas.

- $xyz + \bar{x}yz$
- $xy\bar{z} + x\bar{y}\bar{z} + \bar{x}y\bar{z} + \bar{x}\bar{y}\bar{z}$
- $\bar{x}yz((x + \bar{z}) + (\bar{y} + \bar{z}))$
- $\bar{x}yz + \bar{x}\bar{y}\bar{z}$
- $wx\bar{y}z + wx(y\bar{z} + \bar{y}\bar{z}) + w\bar{x}(yz + y\bar{z}) + \bar{w}\bar{x}\bar{y}z + \bar{w}x\bar{y}\bar{z}$

3.2.6.- Use mapas de Karnaugh para simplificar la siguiente expresión booleana:

$$(wxy) + (wx\bar{z}) + (w\bar{y}z) + (\bar{x}\bar{y}z)$$

3.2.7.- Use mapas de Karnaugh para simplificar el circuito correspondiente a las siguientes tablas de verdad.

a)

x	y	z	$x + (y(x\bar{z}))$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

b)

x	y	z	$\bar{x}yz + x(\bar{y}z + y\bar{z} + yz)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

c)

x	y	z	$x + (y(\bar{x} + \bar{z}))$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

d)

x	y	$x(x + y)$
0	0	0
0	1	0
1	0	1
1	1	1

3.2.8.- Escriba las expresiones que corresponden a los óvalos marcados en los mapas de Karnaugh que se encuentran a continuación.

a)

	y	\bar{y}
x	1	1
\bar{x}	1	

b)

	yz	$y\bar{z}$	$\bar{y}\bar{z}$	$\bar{y}z$
x	1	1	1	
\bar{x}	1	1		

(c)

	yz	$y\bar{z}$	$\bar{y}\bar{z}$	$\bar{y}z$
wx			1	
$w\bar{x}$	1	1	1	1
$\bar{w}\bar{x}$				
$\bar{w}x$			1	

(d)

	yz	$y\bar{z}$	$\bar{y}\bar{z}$	$\bar{y}z$
wx				1
$w\bar{x}$				1
$\bar{w}\bar{x}$	1		1	1
$\bar{w}x$				1

- 3.2.9.- Obtenga las fórmulas y circuitos mínimos a partir de los mapas del ejercicio anterior.
- 3.2.10.- Muestre a detalle el proceso de minimización mediante mapas de Karnaugh para los circuitos obtenidos en los ejercicios 14, 15, 16 y 17 de la sección 3.1.

3.3. Método de Quine-McCluskey para minimizar

El problema con los mapas de Karnaugh es que trabajar con más de seis variables de manera gráfica es imposible. De hecho, en la práctica, para cinco o seis variables ya es demasiado complicado pues tenemos que manejar cuadrículas con 32 o 64 celdas

El método de Quine-McCluskey fue diseñado en la década de 1950. Consiste en manejar los mintérminos como cadenas de bits o números binarios (ceros y unos). Identificaremos al j -ésimo mintérmino como P_j , como lo hicimos al calcular el costo de la implementación de una fórmula con un circuito digital. Todas las cadenas tienen el mismo número de posiciones y a cada variable se le asigna la misma posición en cada cadena. Si la literal aparece tal cual se le asigna un 1 en esa posición y si aparece negada se le asigna un 0. Cuando la variable no aparece, que significa que aparecía tanto en positivo como negada, se marca la posición con un guión (que indica que se puede usar cualquiera de los valores³). El índice j de los mintérminos corresponde al valor decimal del número binario representado por el mintérmino, vistos estos ordenados desde el 00...0 hasta el 11...1.

3.3.1. Tabla de mintérminos

Un *mintérmino esencial* es aquél en el que la función toma el valor 1. El primer paso consiste, entonces, en determinar los mintérminos esenciales de la función y asignarles valores booleanos utilizando ceros y unos. Veamos la codificación de un ejemplo con cuatro variables que corresponde a la tabla de verdad 3.5, en la que los mintérminos esenciales son los de los renglones P_0, P_2, P_4 a P_8, P_{10} a P_{12} , y P_{14} .

Tabla 3.5. Tabla de verdad con cuatro variables

						(continúa...)					
mintérm	w	x	y	z	$f(w, x, y, z)$	mintérm	w	x	y	z	$f(w, x, y, z)$
P_0	0	0	0	0	1	P_8	1	0	0	0	1
P_1	0	0	0	1	0	P_9	1	0	0	1	0
P_2	0	0	1	0	1	P_{10}	1	0	1	0	1
P_3	0	0	1	1	0	P_{11}	1	0	1	1	1
P_4	0	1	0	0	1	P_{12}	1	1	0	0	1
P_5	0	1	0	1	1	P_{13}	1	1	0	1	0
P_6	0	1	1	0	1	P_{14}	1	1	1	0	1
P_7	0	1	1	1	1	P_{15}	1	1	1	1	0

³En inglés a estos valores se les conoce como *don't care conditions*.

Una vez hecha la tabla de la función de la manera antes mencionada, se expresa la función fácilmente como la suma de los mintérminos esenciales que, como vimos al principio del capítulo, representan a la función (aquellos renglones que tienen 1 en el valor de la función). En el método de Quine-McCluskey a esta suma se le conoce como un *implicante*. Decimos que P es un implicante de la función F si siempre que P es verdadero, también lo es F (viene de $P \rightarrow F$). De la tabla anterior tenemos

$$f(w, x, y, z) = \bar{w}\bar{x}\bar{y}\bar{z} + \bar{w}\bar{x}y\bar{z} + \bar{w}x\bar{y}\bar{z} + \bar{w}xy\bar{z} + \bar{w}xyz + w\bar{x}\bar{y}\bar{z} \\ + w\bar{x}\bar{y}z + w\bar{x}y\bar{z} + w\bar{x}yz + wx\bar{y}\bar{z} + wxy\bar{z}$$

Para el método de Quine-McCluskey podemos observar que la codificación de los mintérminos consiste, precisamente, de las cadenas de ceros bajo las variables que participan en la función, mientras que el índice asignado al mintérmino corresponde al valor decimal del número binario en el renglón en que se encuentran.

3.3.2. Eliminación de variables combinando mintérminos

A continuación lo que deseamos es encontrar aquellos mintérminos esenciales que difieren en una sola posición, lo que indica que se puede factorizar pues esa posición refleja la existencia de la disyunción de una literal y su negación. Por ejemplo, si tuviésemos los mintérminos 0100 y 1100, tenemos

$$\bar{w}x\bar{y}\bar{z} + wx\bar{y}\bar{z} = (\bar{w} + w)(x\bar{y}\bar{z}) = (\text{true})(x\bar{y}\bar{z}) = x\bar{y}\bar{z}$$

Para que dos mintérminos difieran en una posición, una de ellas tiene que tener un 1 más (o menos) que la otra, por lo que se van a comparar aquellos términos que difieren en uno en el número de dígitos 1 (literales no negadas). Para ello, clasificamos las cadenas que correspondan a un mintérmino esencial (donde $f(w, x, y, z) = 1$) por el número de dígitos 1 que tengan, quedando en la misma clase aquellas cadenas que tengan el mismo número de unos. En la tabla 3.6 de la siguiente página hacemos esta clasificación y los listamos en orden descendente por el número de unos.

A continuación procedemos a la minimización. La presencia de una variable y su negación quiere decir que hay dos mintérminos en los cuales la posición correspondiente a la variable aparece con 0 en una y con 1 en la otra. Aquella en la que aparece esa posición con 1 tiene un 1 más que la otra, por lo que debemos comparar parejas de mintérminos que tienen un 1 de diferencia. Por lo tanto comparamos cada uno de los que tienen tres unos con cada uno de los que tienen dos unos, los de dos unos con los de un uno, y los de un uno con los de cero unos. Aquellos que sean distintos por una única posición y que tengan en esa posición 0 y 1, se sustituyen por un mintérmino que tenga un guión en esa posición.

Por ejemplo, de la tabla 3.5 que ya mencionamos, debemos comparar P_7 con P_5 , P_6 , P_{10} y P_{12} . Sin embargo, tanto P_{10} como P_{12} difieren en más de una posición con P_7 , por lo que no se les procesa en este primer paso. Tendríamos las comparaciones que se encuentran a continuación de la tabla 3.5.

Tabla 3.6. Método de Quine-McCluskey

Núm.	mintérmino	# de 1	
P_7	0111	3	✓
P_{11}	1011		✓
P_{14}	1110		✓
P_5	0101	2	✓
P_6	0110		✓
P_{10}	1010		✓
P_{12}	1100		✓
P_2	0010	1	✓
P_4	0100		✓
P_8	1000		✓
P_0	0000	0	✓

$$\begin{array}{cccc} P_7 = & 0 & 1 & 1 & 1 \\ P_5 = & 0 & 1 & 0 & 1 \\ \hline & 0 & 1 & - & 1 \end{array}$$

$$\begin{array}{cccc} P_7 = & 0 & 1 & 1 & 1 \\ P_6 = & 0 & 1 & 1 & 0 \\ \hline & 0 & 1 & 1 & - \end{array}$$

En una primera tabla comparamos entre sí aquellos mintérminos esenciales que difieren en un 1, y que tienen tres o dos términos en 1 (que están en la primera y segunda clase) –procederemos a asignarle al resultado un nuevo número R_i para poder seguir comparando; vale la pena mencionar que, en este caso, el subíndice i no refleja ninguna propiedad en cuanto al número binario representado, sino que es únicamente secuencial a partir del 16-. Están marcados con una palomita aquellos términos que son “simplificados” –aparecen combinándose después con algún otro término– de tal manera que al final recogeremos de las tablas aquellos términos que no estén palomeados.

Combinación	Comparando	Cadena	Mintérm.	Id.	
P_7	0111	01 – 1	$\overline{w}xz$	R_{16}	✓
P_5	0101				
P_7	0111	011 –	$\overline{w}xy$	R_{17}	✓
P_6	0110				
P_{11}	1011	101 –	$w\overline{x}y$	R_{18}	
P_{10}	1010				
P_{14}	1110	–111	xyz	R_{19}	
P_6	0110	–111			
P_{14}	1110	1 – 10	$wy\overline{z}$	R_{20}	✓
P_{10}	1010				
P_{14}	1110	11 – 0	$wx\overline{z}$	R_{21}	✓
P_{12}	1100				

A continuación comparamos los minterminos esenciales de la segunda clase con los de la tercera clase, pero sólo aquellos que difieran en una sola posición.

Combinación	Comparando	Cadena	Mintérm.	Id.	
P_5 P_4	0101 0100	011—	$\overline{w}xy$	$= R_{17}$	✓
P_6 P_2	0110 0010	0—10 0—10	$\overline{w}y\overline{z}$	R_{22}	✓
P_6 P_4	0110 0100	01—0	$\overline{w}x\overline{z}$	R_{23}	✓
P_{10} P_2	1010 0010	—010	$\overline{x}y\overline{z}$	R_{24}	✓
P_{10} P_8	1010 1000	10—0	$w\overline{x}\overline{z}$	R_{25}	✓
P_{12} P_4	1100 0100	—100	$x\overline{y}\overline{z}$	R_{26}	✓
P_{12} P_8	1100 1000	1—00	$w\overline{y}\overline{z}$	R_{27}	✓

Por último, en esta primera etapa, comparamos aquellos minterminos esenciales de la tercera y cuarta clase entre sí, si es que difieren únicamente en una posición, lo que se muestra en la siguiente página.

Combinación	Comparando	Cadena	Mintérm.	Id.	
P_2 P_0	0010 0000	00—0	$\overline{w}\overline{x}\overline{z}$	R_{28}	✓
P_4 P_0	0100 0000	0—00	$\overline{w}\overline{y}\overline{z}$	R_{29}	✓
P_8 P_0	1000 0000	—000	$\overline{x}\overline{y}\overline{z}$	R_{30}	✓

Con esto hemos terminado la comparación de los minterminos esenciales originales, quedando “vivos” aquellos que no participaron en ninguna reducción. Sin embargo, los once minterminos esenciales que teníamos se combinaron con al menos algún otro, por lo que no sobrevive ningún mintermino con más de tres variables, un considerable ahorro en cuanto al costo de la implementación.

A continuación compararemos los minterminos que cumplan con tener el guión en la misma posición y que difieran en uno en el número de unos:

Combinación	Comparando	Cadena	Mintérm.	Id.	
R_{16} R_{23}	01 – 1 01 – 0	$\bar{w}x$	01 – –	R_{31}	
R_{20} R_{22}	1 – 10 0 – 10	– – 10	$y\bar{z}$	R_{32}	✓
R_{20} R_{27}	1 – 10 1 – 00	1 – –0	$w\bar{z}$	R_{33}	✓
R_{21} R_{23}	11 – 0 01 – 0	1 – –0	$x\bar{z}$	$= R_{33}$	✓
R_{21} R_{25}	11 – 0 10 – 0	1 – –0	$w\bar{z}$	$= R_{33}$	✓
R_{22} R_{29}	0 – 10 0 – 00	0 – –0	$\bar{w}\bar{z}$	R_{34}	✓
R_{23} R_{28}	01 – 0 00 – 0	0 – –0	$\bar{w}\bar{z}$	$= R_{34}$	✓
R_{24} R_{30}	–010 –000	–0 – 0	$\bar{x}\bar{z}$	R_{35}	
R_{26} R_{30}	–100 –000	– – 00	$\bar{y}\bar{z}$	R_{36}	✓
R_{27} R_{29}	1 – 00 0 – 00	– – 00	$\bar{y}\bar{z}$	$= R_{36}$	✓

Volvemos a hacer lo mismo con los términos que aparecen en la tabla anterior, que tienen dos guiones, viendo cuáles difieren en una posición. Se muestra a continuación.

Combinación	Comparando	Cadena	Mintérm.	Id.	
R_{32} R_{36}	– – 10 – – 00	– – –0	\bar{z}	R_{38}	
R_{33} R_{34}	1 – –0 0 – –0	– – –0	\bar{z}	$= R_{38}$	

De las tablas anteriores, tomando todos aquellos términos que no están palomeados (que no se usaron para eliminar variables), tenemos la siguiente fórmula, que representa al circuito dado:

$$f(w, x, y, z) = w\bar{x}y + xyz + \bar{w}x + \bar{x}\bar{z} + \bar{z}$$

La fórmula original requiere de 37 compuertas para implementarla (sin considerar ningún tipo de reutilización de productos ya calculados, ni uso de factorización) mientras

que la simplificación obtenida requiere de únicamente 11 compuertas (el lector interesado lo puede verificar).

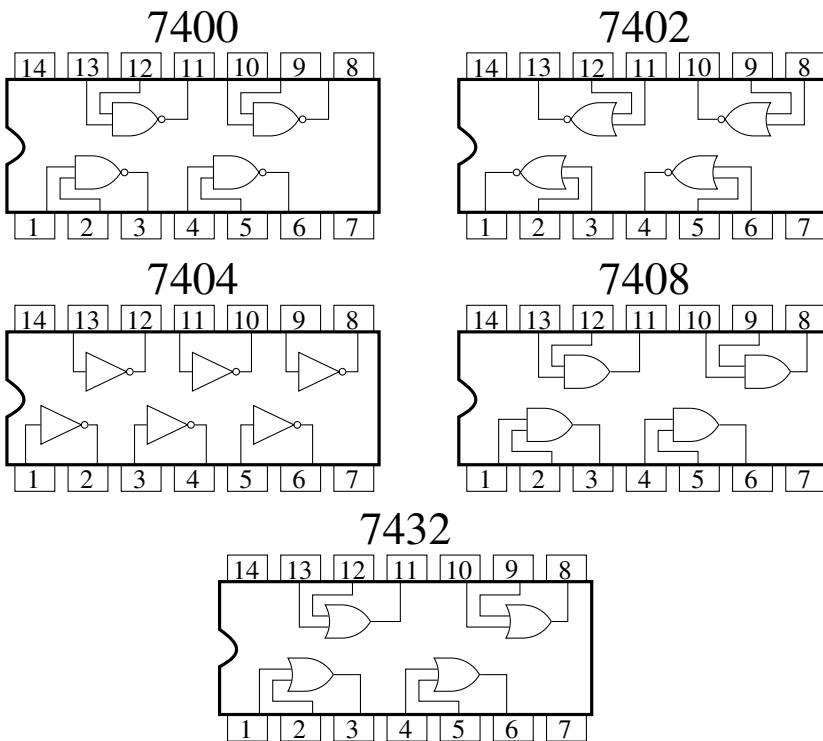
Hay que notar que el algoritmo utilizado para la minimización presupone la comparación dos a dos de todos los minterminos que tienen k unos contra todos los términos que tienen $k - 1$ unos, lo que hace que la complejidad del algoritmo lo convierta en un algoritmo *intratable*, algo de lo que no había conciencia sino hasta 1976, en que se definió la teoría de complejidad de algoritmos. Por ello podemos decir que no hay algoritmo *razonable* para minimizar fórmulas booleanas y se deben utilizar, en todo caso, *heurísticas*, que son procesos computacionales que, aunque llevan mucho menos tiempo de ejecución, no garantizan que la solución sea, en efecto, óptima.

Para terminar, si un mintermino no contiene a alguna variable y se usa el método de Quine-McCluskey, se puede poner directamente el guión en la posición de la variable que no aparece.

Colofón: tabletas (*chips*) de compuertas

Para fabricar circuitos lógicos se dispone de conjuntos de compuertas con un número fijo de compuertas iguales, presentadas en una tableta (*chip*), por lo que muchas veces se usan las equivalencias lógicas para utilizar más eficientemente las compuertas lógicas. En la figura 3.17 se muestran distintos tipos de tabletas disponibles.

Figura 3.17. Algunas tabletas disponibles con compuertas lógicas



Ejercicios

3.3.1.- Use el método de Quine-McCluskey para simplificar el circuito correspondiente a las siguientes tablas de verdad.

(a)

x	y	z	$x + (y(x\bar{z}))$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(b)

x	y	z	$\bar{x}yz + x(\bar{y}z + y\bar{z} + yz)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

(c)

x	y	z	$x + (y(\bar{x} + \bar{z}))$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(d)

x	y	$x(x + y)$
0	0	0
0	1	0
1	0	1
1	1	1

3.3.2.- Use el método Quine-McCluskey para minimizar la función de la figura 3.13.

3.3.3.- Use el método Quine-McCluskey para minimizar las funciones del ejercicio (3.2.5).

3.3.4.- Use el método de Quine-McCluskey para simplificar la siguiente expresión booleana: $wxy + wx\bar{z} + w\bar{y}z + \bar{x}\bar{y}z$.

3.4. Dispositivos digitales combinatorios

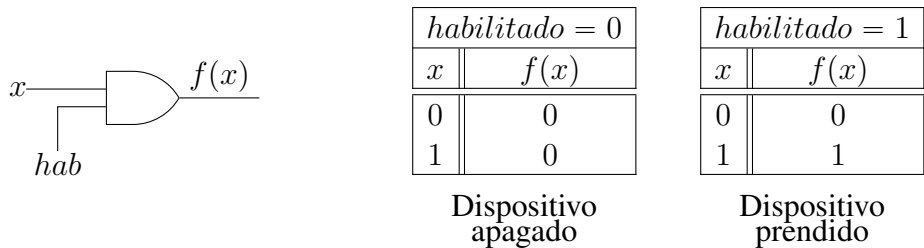
A partir de circuitos lógicos se pueden construir dispositivos digitales que son característicos de las computadoras actuales. Los circuitos lógicos pueden ser de naturaleza *combinatoria*, que consisten de un conjunto de compuertas lógicas cuyas salidas, en todo momento, se calculan usando operaciones lógicas directamente de la combinación de

sus valores de entrada. Ya vimos uno de estos dispositivos que es el sumador (tanto el medio como el completo).

Por otro lado tenemos circuitos lógicos que tienen elementos cuya función es retroalimentar al circuito algunos de los valores producidos por el mismo circuito. En este tipo de circuitos se debe tomar en cuenta lo que se conoce como el retardo (*delay*) del dispositivo, ya que el paso de una señal por una compuerta se lleva alrededor de 10 nanosegundos. Esto quiere decir, por ejemplo, que si la salida de una compuerta se alimenta a la entrada de otra, esto va sucediendo en orden y con un cierto retardo. Si además la compuerta a la que se retroalimenta es ella misma o alguna que recibe sus primeras entradas al mismo tiempo que ésta, se debe tomar en cuenta dicho retardo. Es esta característica la que da el nombre de *secuencial* a este tipo de circuitos o dispositivos. Continuaremos por el momento trabajando con circuitos combinatorios.

Muchos de los dispositivos que presentaremos cuentan con una entrada que *habilita* al dispositivo; funciona como un interruptor que puede estar prendido o apagado y que cuando está prendido permite que el circuito funcione. Por ejemplo, si queremos un dispositivo que deje pasar la señal cuando esté habilitado pero la inhiba cuando no, podemos pensar en una compuerta *and* donde una de las entradas es la señal de habilitación y la otra es la señal que se desea transmitir –véase figura 3.18.

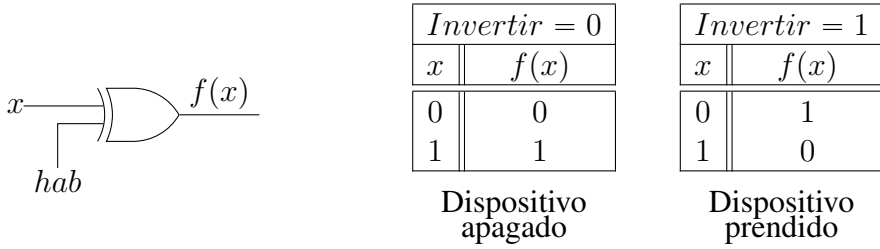
Figura 3.18. Circuito que inhibe o deja pasar una señal



En este caso se puede pensar en la entrada que corresponde a *x* como la línea de entrada, mientras que la que corresponde a *habilitar* como una línea de control.

Otro dispositivo común es lo que se conoce como inversor. En este caso el papel del dispositivo es, simplemente, complementar la señal, en caso de que esté habilitado. Una compuerta *xor* funciona como un inversor si consideramos a una de las entradas como la línea de dato y la otra entrada como la señal de control –véase figura 3.19–.

Figura 3.19. Inversor usando una compuerta *xor*



3.4.1. Multiplexores

Un multiplexor es un dispositivo que permite elegir una de varias entradas para ser emitida como salida del dispositivo. Consiste de k líneas de entrada, $\lceil \log_2 k \rceil$ líneas de control⁴ –que permiten elegir alguna de las k entradas– y una línea de salida que corresponde a la línea de entrada elegida.

Si suponemos que tenemos cuatro líneas de entrada a elegir, d_0, d_1, d_2 y d_3 , requerimos de dos líneas de control para elegir entre cuatro posibilidades, s_0 y s_1 (00,01,10,11). La tabla de cuál línea elegir se encuentra en la figura 3.20(a).

El dispositivo digital que selecciona una de cuatro entradas se puede ver en la figura 3.21. La tabla de verdad para este circuito se encuentra la figura 3.20(b).

Figura 3.20. Función de selección y tabla de verdad

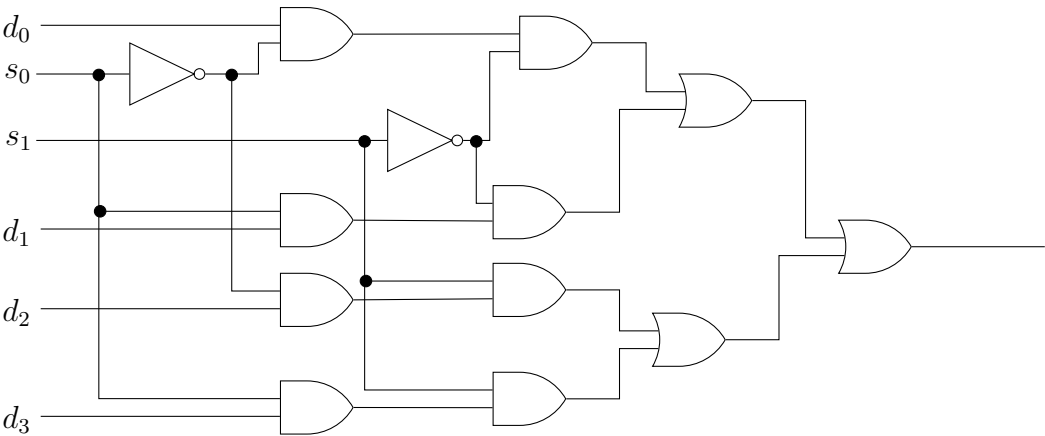
(a)

s_1	s_0	se elige la línea
0	0	0
0	1	1
1	0	2
1	1	3

(b)

s_0	s_1	d_0	d_1	d_2	d_3	Out
0	0	1	–	–	–	1
0	1	–	1	–	–	1
1	0	–	–	1	–	1
1	1	–	–	–	1	1

Figura 3.21. Multiplexor de cuatro entradas



Esta tabla muestra particularidades que no hemos revisado hasta ahora. La combinatoria nos dice que la tabla debería tener $2^6 = 64$ renglones y, sin embargo, esta tabla tiene

⁴El resultado de $\log_2 k$ es un número real, pero el número de líneas tiene que ser entero, por lo que le aplicamos la función $\lceil \log_2 k \rceil$ cuyo significado es el menor entero que sea mayor o igual a $\log_2 k$. Similarmente tenemos $\lfloor x \rfloor$ que significa el mayor entero menor o igual a x .

únicamente cuatro renglones, uno por cada posible combinación de s_0 y s_1 . Los guiones en la tabla es lo que se conoce como condiciones que no importan (*don't care*), ya que el resultado no va a tomar en cuenta esos valores. En los otros 60 casos el circuito entregará el valor de falso. De esta tabla podemos dar una expresión booleana relativamente sencilla para el multiplexor con cuatro entradas:

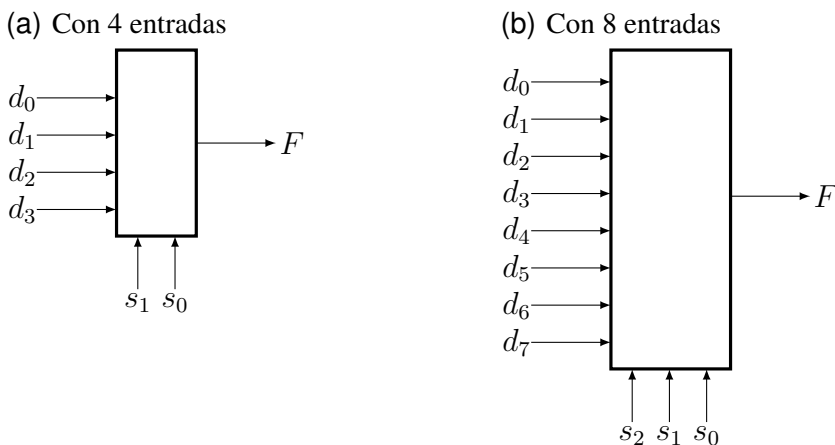
$$Out = \overline{s_1}(\overline{s_0}d_0) + \overline{s_1}(s_0d_1) + s_1(\overline{s_0}d_2) + s_1(s_0d_3)$$

Definimos el *nivel de un circuito* como el número máximo de compuertas por las que tiene que pasar una señal. En el circuito de la figura 3.13, en la página 124, el nivel del circuito es 5, pues la señal de x tiene que pasar por cinco compuertas –aunque no nada más la señal de x –. Si tenemos la función xyz , usando compuertas de dos entradas tendríamos nivel 2, mientras que usando compuertas de tres entradas tendríamos nivel 1. El nivel de un circuito incide directamente sobre el tiempo que el circuito se va a tardar en dar una respuesta una vez recibidas todas sus entradas.

Los paréntesis en la expresión anterior se usan para seguir usando compuertas de dos entradas únicamente, aunque podríamos usar compuertas de tres o cuatro entradas y reducir el nivel del circuito a dos, lo que representa un menor tiempo de ejecución con un costo mayor por compuerta.

Se utilizan diagramas de bloque para este tipo de dispositivos. Al multiplexor de 4 entradas, por ejemplo, le corresponde el diagrama de la figura 3.22(a) y a un multiplexor de 8 entradas le corresponde el diagrama de bloque de la figura 3.22(b). Esta clase de representaciones se conocen como diagramas de bloque y dada su utilidad los utilizaremos también más adelante.

Figura 3.22. Diagramas de bloque para multiplexores

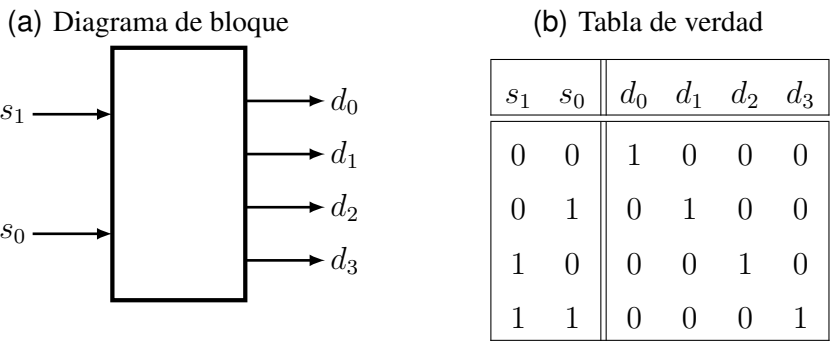


Como se puede ver en estas figuras, las líneas que seleccionan funcionan en notación binaria: para elegir entre cuatro posibles entradas se requieren dos líneas (posiciones binarias), de $00 = 0$ a $11 = 3$, pero para elegir entre 8 posibles datos se requiere de tres líneas (bits) que van de $000 = 0$ a $111 = 7$.

3.4.2. Decodificadores binarios

Un *decodificador* es un dispositivo que recibe un número binario de m bits –que puede ser visto como m líneas de señal– y produce como salida otro número binario de n bits – m y n pueden ser iguales. El diagrama de bloque de un decodificador con dos líneas de entrada y cuatro líneas de salida⁵ se puede ver en la figura 3.23.

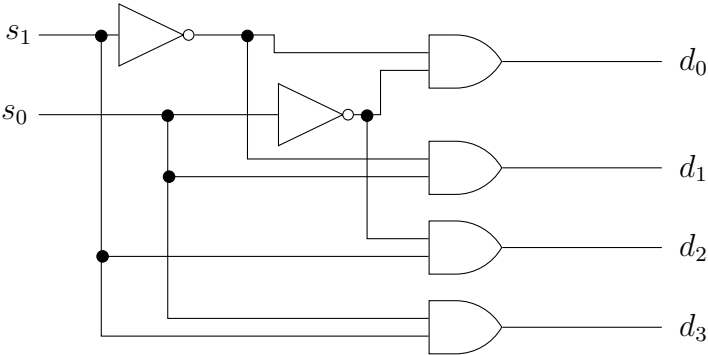
Figura 3.23. Decodificador binario de 2×4



El decodificador ilustrado recibe un número binario entre cero y tres y elige la línea que corresponde a ese número para que valga 1 mientras el resto de las líneas valen 0.

El circuito digital que corresponde a la tabla de verdad se encuentra en la figura 3.24.

Figura 3.24. Implementación de un decodificador de 2×4



3.4.3. Retardos y tiempo total

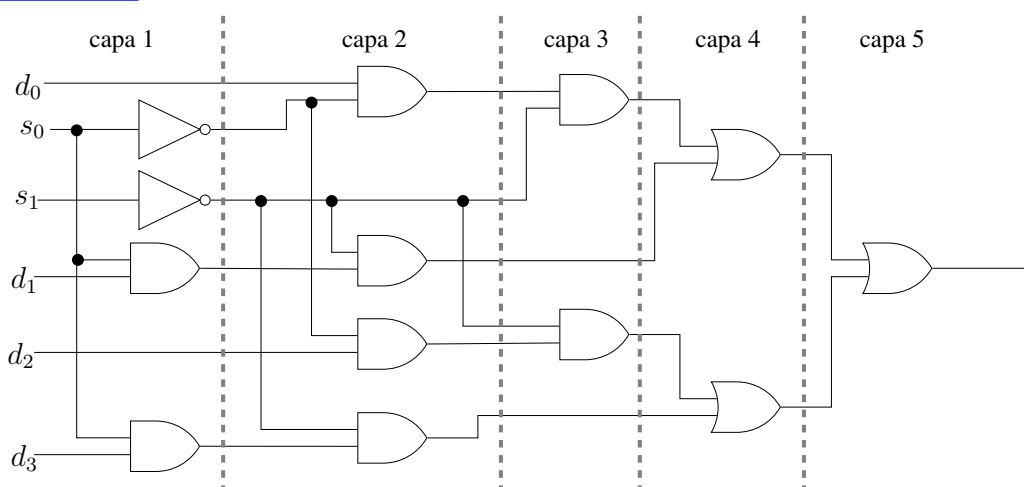
Aunque cuando observamos los circuitos lógicos estamos pensando en una respuesta instantánea en cada compuerta y la transmisión de la señal también la pensamos inmediata si esto se comportara así en la realidad no seríamos capaces de construir circuitos combinatorios pues tendríamos a todas las compuertas accionadas exactamente al mismo tiempo, sin esperar a los resultados entregados por la compuerta anterior. Cada compuerta por la

⁵Un decodificador con n líneas de entrada y k líneas de salida se representa como $n \times k$.

que pasa una señal presenta lo que se como un *retardo*, además que la señal toma tiempo proporcional a la longitud del cable para transmitirse de una compuerta a otra. Esto nos permite organizar y ordenar los circuitos combinatorios para garantizar que una compuerta no funcione hasta en tanto estén presentes todas sus entradas, posiblemente generadas por compuertas anteriores.

Cuando tenemos un circuito combinatorio el orden queda determinado por las *capas* por las que pasan las señales y la distancia entre las capas. A todos los retardos que suceden simultáneamente es a lo que llamamos una capa. En la figura 3.25 mostramos las capas que podemos considerar en el multiplexor de la figura 3.21, aunque la longitud y posicionamiento de las compuertas no son únicos ni forzosamente óptimos.

Figura 3.25. Capas en un multiplexor de cuatro entradas



Si bien, como acabamos de mencionar, no podemos garantizar que la longitud de los cables es óptima o que el orden en que aparecen las compuertas en cada capa es la mejor, sí es seguro que el circuito requiere de cinco capas, pues el número de capas depende de cómo se vayan combinando las señales para que las salidas de algunas compuertas puedan ser utilizadas como entradas a otras. Una compuerta no puede actuar hasta que reciba todas las señales que espera, por lo que las capas dan un orden de ejecución. De esto, el tiempo total que se llevaría el circuito en llevar las señales de entrada hasta la salida sería la suma de los retardos máximos en cada capa, agregando asimismo el tiempo de retardo máximo al recorrer los cables, ya que todas las compuertas de una misma capa pueden trabajar en paralelo.

Ejercicios

- 3.4.1.- Cuando decimos que tenemos un multiplexor $n \times k$ queremos decir que el multiplexor elige k valores de entre n posibles. Diseñe un multiplexor 32×1 usando dos diagramas de bloque de multiplexores 16×1 (puede utilizar compuertas adicionales).

- 3.4.2.- Utilice dos decodificadores de 2×4 (los diagramas de bloque) para diseñar un decodificador de 4×8 (puede utilizar compuertas adicionales).
- 3.4.3.- Diseñe la tabla de verdad de un decodificador 3×10 de binario a decimal.
- 3.4.4.- Diseñe la tabla de verdad para un decodificador 4×16 .
- 3.4.5.- Diseñe un circuito decodificador con cuatro entradas para la tabla de verdad que se encuentra a continuación en la siguiente página (el “—” significa que no importa si es 0 o 1):

Entradas				Salidas		
D ₃	D ₂	D ₁	D ₀	A ₁	A ₀	V
0	0	0	0	—	—	0
0	0	0	1	0	0	1
0	0	1	—	0	1	1
0	1	—	—	1	0	1
1	—	—	—	1	1	1

A este circuito se le conoce como un codificador de prioridad. Note que, a diferencia de los decodificadores, este tipo de circuitos combinatorios tienen más entradas que salidas.

- 3.4.6.- Para el circuito correspondiente a la figura 3.13, determine la capa en la que se encuentra cada una de las compuertas.

3.5. Circuitos digitales secuenciales

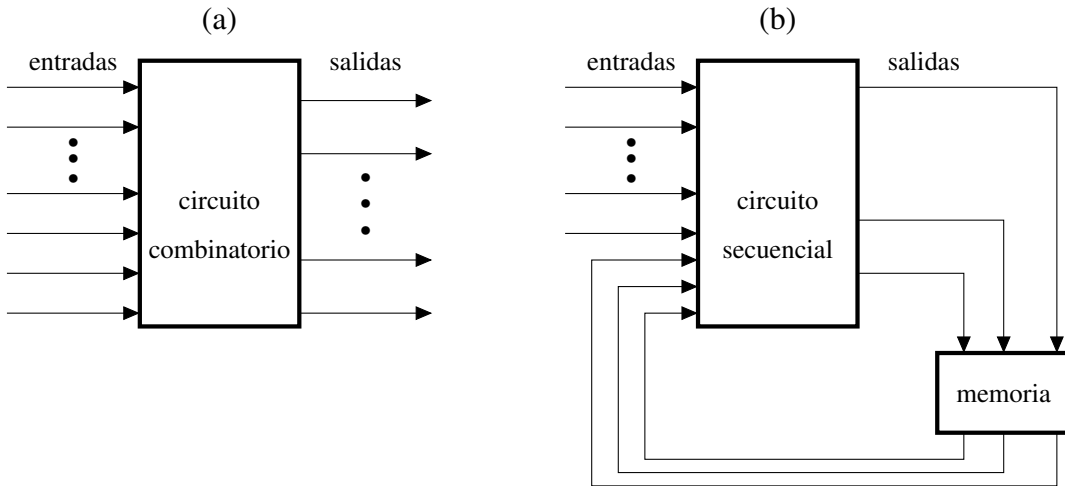
Mientras que en los circuitos combinatorios, como los que hemos visto hasta ahora, las salidas de las compuertas se conectan siempre a compuertas posteriores, en los circuitos secuenciales algunas de las salidas se retroalimentan a compuertas anteriores.

Para construir componentes de computadoras es necesario contar con circuitos capaces de “recordar”, es decir, de representar un *estado*. “Recordar” quiere decir almacenar, por lo que los circuitos secuenciales reflejan no sólo las entradas actuales, sino también la historia de entradas que ha habido, que es precisamente a lo que llamamos un estado.

Esto se logra haciendo que la respuesta del circuito se rija tanto por las entradas actuales como por el resultado de entradas anteriores; a esto último se le conoce como retroalimentación (*feedback*). A este tipo de circuitos se les conoce también como elementos de *memoria* y la salida de cualquier elemento de memoria depende tanto de las entradas como de los valores que tenga almacenados dentro del circuito. La manera de almacenar estado es retroalimentando alguna de las salidas a las entradas, por lo que siempre se considera un

estado inicial indeterminado ya que no ha habido retroalimentación. Todos los elementos de memoria, por lo tanto, presentan un *estado* y son secuenciales. Esto último quiere decir que su comportamiento se observa a través del tiempo. Un circuito combinatorio, con un diagrama de bloques, se puede ver en la figura 3.26(a) mientras que el diagrama de bloques de un circuito secuencial se observa en la figura 3.26(b).

Figura 3.26. Comparación entre circuitos combinatorios y secuenciales

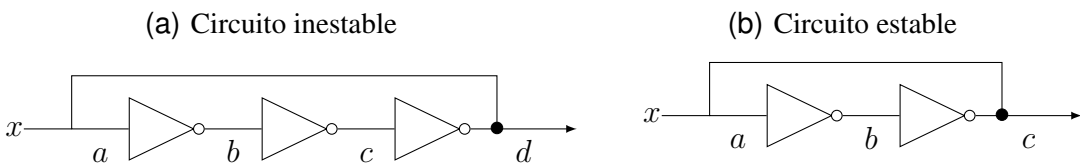


Para describir el comportamiento (o la salida) de un circuito secuencial se debe, como ya mencionamos, observarlo a través del tiempo y se vuelve muy importante el orden en que las compuertas se activan (reciben una señal).

Los circuitos combinatorios están contruidos a partir de compuertas, mientras que los circuitos secuenciales tienen como componente fundamental a los circuitos de pestillos⁶ –en adelante *circuitos latch*– y a los biestables –en adelante *circuitos flip flop*–.

Las figuras 3.27(a) y 3.27(b) presentan dos circuitos secuenciales simples.

Figura 3.27. Circuitos simples con retroalimentación



Tratemos de analizar el comportamiento de estos circuitos secuenciales. El circuito de la figura 3.27(a) tiene tres compuertas *not* que invierten el valor de su entrada. Si suponemos que al encender el circuito el valor de la entrada x es 0, al final del circuito tendremos que d es 1 en la primera vuelta; eso hace que el valor de entrada a la primera compuerta cambie a 1 ($a = 1$), activando nuevamente la compuerta ante el cambio de la señal, se invierte al pasar

⁶Los circuitos de pestillo y los biestables **también** están contruidos con compuertas, pero la diferencia principal es que se utilizan como un bloque.

por la primer compuerta ($b = 0$), se vuelve a invertir al pasar por la segunda compuerta ($c = 1$) y la tercera compuerta vuelve a invertir el valor, quedando $d = 0$. Si dejamos correr el tiempo el circuito está recibiendo la entrada retroalimentada en a . Como la señal alimentada a la primera compuerta está cambiando, el circuito sigue activado, por lo que, a través del tiempo y tomando en cuenta los retardos de las compuertas, obtendremos valores distintos dependiendo del momento en que observemos a d . Si se desea saber el valor de la señal d en un momento dado, ésta puede valer 0 o 1 y nunca deja de activarse el circuito. A esto se le llama un circuito *inestable*, ya que no produce (mantiene) el mismo valor en ausencia de cambios en las señales.

En cambio, en el circuito de la figura 3.27(b), si al encenderse el circuito la señal de x es 0 ($a = 0$), al pasar por la primera compuerta se invierte ($b = 1$) y al pasar por la segunda compuerta se vuelve a invertir ($c = 0$). Al retroalimentarse esta señal a la primera compuerta, la entrada no cambia de valor, por lo que el circuito ya no se activa. No importa cuánto tiempo dejemos transcurrir o el momento en el que lo hagamos, cuando se verifique el valor de c éste seguirá siendo 1 –similarmente si se inicia con un valor de 1–. Decimos entonces que el circuito es *estable*.

Como las compuertas digitales requieren de señal eléctrica para funcionar suponemos que al encenderse el primer valor detectado en la salida es aleatorio en el caso de los dos circuitos secuenciales que acabamos de revisar.

Los circuitos secuenciales se clasifican, a su vez, en síncronos y asíncronos, dependiendo de si utilizan o no, respectivamente, los pulsos de un reloj. En los circuitos secuenciales síncronos el cambio de estado se da relativo al pulso de un reloj, mientras que en los asíncronos el cambio de estado se da cuando hay un cambio en la entrada. Revisaremos primero los circuitos secuenciales asíncronos, por parecerse éstos más a los combinatorios.

3.5.1. Circuitos secuenciales asíncronos

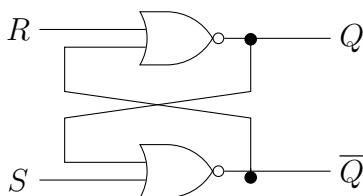
El uso de este tipo de circuitos es frecuente en sistemas que deben responder rápidamente sin esperar el pulso del reloj, además de tener un bajo consumo de energía y menor interferencia electromagnética. También se utilizan para comunicar entre sí a subsistemas síncronos donde cada uno de los subsistemas tiene su propio reloj.

Una desventaja de los circuitos secuenciales asíncronos es que su análisis y diseño no es sencillo, pues no se puede predecir el orden o el momento en que las señales van a cambiar o van a ser consumidas, incluso tomando en cuenta los retardos en las compuertas. El orden en que se procesen las señales puede alterar el resultado del circuito.

Un circuito de pestillo (en adelante *latch*) es el circuito secuencial más sencillo considerado como una memoria y se usa para establecer un estado en un circuito secuencial. La información que logra almacenar es la correspondiente a un bit. El latch puede mantener un estado binario indefinidamente, hasta que una señal de entrada le indique que debe cambiar de estado. Los latches⁷ más sencillos tienen dos señales de entrada –conocidas como S

(*set* o establecer) y R (*reset*), y producen dos salidas⁸ generalmente denotadas con Q y \overline{Q} , donde Q se considera el estado del circuito (la señal almacenada). En la figura 3.28 vemos el diagrama de un circuito *latch*.

Figura 3.28. Diagrama de un circuito *latch* básico



Este latch recibe el nombre de latch- SR por los nombres de sus entradas. Está combinando dos compuertas *nor* —compuertas *or* cuya salida se invierte— y su estado almacena la señal de Q . Los nombres de las salidas son Q y \overline{Q} , ya que letra Q se usa generalmente para representar los estados de un sistema.

El latch- SR tiene dos estados útiles. Cuando las salidas son $Q = 1$ y $\overline{Q} = 0$ decimos que está en el estado *establecido* (el latch recuerda que el último cambio de señal fue habilitar la señal *Set*), mientras que si las salidas son $Q = 0$ y $\overline{Q} = 1$ se encuentra en el estado *restablecido* (el latch recuerda que el último cambio de señal fue habilitar la señal *Reset*). Las salidas Q y \overline{Q} deben ser complementarias, pero si ambas entradas son 1, esto indicaría que se desea establecer y restablecer al mismo tiempo, por lo que se presenta un estado inútil en el que ambas salidas son 0.

El funcionamiento de un circuito secuencial no se puede describir directamente con una tabla de verdad, pues una de las entradas de cada compuerta es la salida de alguna otra compuerta, que se lleva a cabo en el siguiente instante de tiempo. Debido a la retroalimentación, aunque no hubiese cambio en las entradas originales, puede haber cambios en las señales retroalimentadas, lo que cambiaría el estado del circuito. El tiempo se puede medir discretamente, con un reloj o por los retardos del circuito. En el caso de los circuitos secuenciales asíncronos, se dice que transcurre una unidad de tiempo cuando cambia alguna de las señales que se alimenta a las compuertas. Por esta razón, el comportamiento de un circuito secuencial se muestra en lo que se conoce como una *tabla característica* que muestra el estado del sistema conforme van cambiando las señales que se le alimentan al mismo. En dicha tabla se debe describir un estado inicial respecto a una cierta entrada, en el que no se conoce el estado del sistema, y estados posteriores conforme las señales de entrada o retroalimentación van cambiando. Es importante mencionar que sólo una de las señales cambia en cierto momento y no ambas simultáneamente. Si se considerara este último caso se nos presentaría lo que se conoce como una *condición de carrera* (*race condition*), en la que no podríamos predecir cuál de las señales se procesa primero.

⁸Estamos usando el plural en español del término en inglés, aunque no sea del todo correcto, gramaticalmente hablando.

⁸Algunos autores usan únicamente una salida, Q , tomando como dado que \overline{Q} se puede obtener a partir de aquélla.

Recordemos la tabla de verdad de una compuerta *nor*:

p	q	$\overline{p + q}$
0	0	1
0	1	0
1	0	0
1	1	0

A continuación mostramos la tabla característica del circuito latch-SR cuando se enciende con la señal de *Set* habilitada. Las columnas con los encabezados Q y \overline{Q} representan el estado en el que se encuentra el sistema en el momento en el que cambia alguna de las señales. Para t_0 representa el estado inicial. Las columnas con el encabezado Q' y \overline{Q}' representa el estado al que pasa el circuito, y que se repite en el siguiente renglón como estado actual.

Tabla 3.7. Tabla característica del circuito latch-SR con $SR = 10$

t	S	R	Q	\overline{Q}	Q'	\overline{Q}'
t_0	1	0	0	1	0	0
t_1	1	0	0	0	1	0
t_2	1	0	1	0	1	0
t_3	1	0	1	0	1	0

De la tabla anterior podemos ver que el sistema se estabiliza en t_2 , ya que si no se cambia la entrada, el circuito ya no se activará pues no se producen cambios en ninguna de las señales.

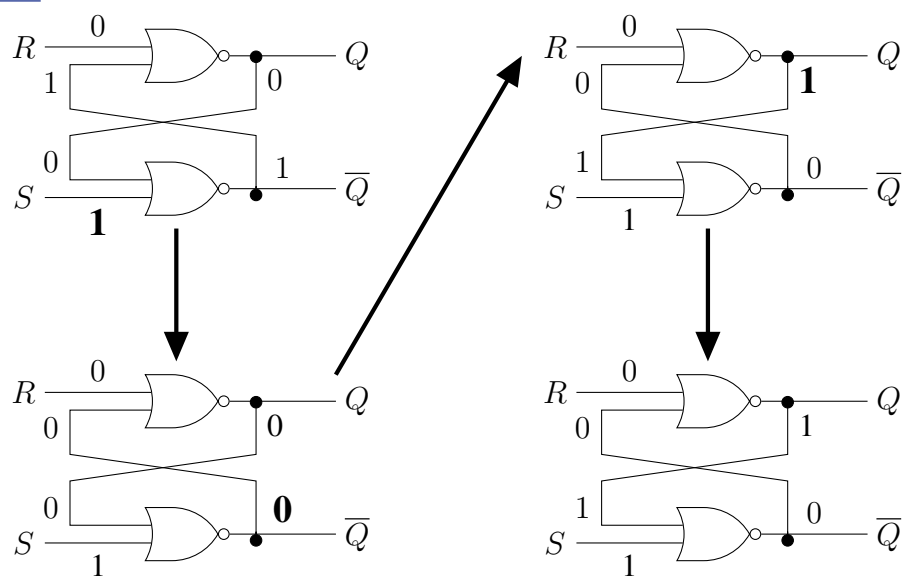
En la figura 3.29 en la siguiente página se aprecian los cambios de señal que va teniendo el latch-SR hasta que alcanza a estabilizar su respuesta. Se encuentra resaltada la señal que cambia y obliga al circuito a seguirse activando.

Si suponemos que el estado inicial es con $Q\overline{Q} = 10$, el circuito alcanzaría inmediatamente un estado estable, donde el valor de Q debe ser 1, pues la señal emitida fue la de establecer la señal. Observemos este comportamiento en la tabla 3.8.

Tabla 3.8. Tabla característica para $SR = 10$ y $Q\overline{Q} = 10$

t	S	R	Q	\overline{Q}	Q'	\overline{Q}'
t_0	1	0	1	0	1	0
t_1	1	0	1	0	1	0

Figura 3.29. Transiciones en un circuito latch-SR



De lo anterior, sin importar el estado anterior del circuito, al aplicar la señal $SR = 10$ el circuito se quedará en el estado $Q\bar{Q} = 10$. La única diferencia es el retardo con que alcanza este estado, dependiendo del estado anterior del circuito.

Para el caso en que la entrada $SR = 01$, veamos las tablas características, dependiendo de cuál es el estado anterior en el momento en que se aplica esta señal –ver tabla 3.9.

Tabla 3.9. Tablas características para $SR = 01$

Circuit diagram for SR=01 at t_0 . Inputs: $R=1$, $S=0$. Current outputs: $Q=0$, $\overline{Q}=1$. The circuit consists of two NOR gates. The top NOR gate has inputs R and \overline{Q} (which is 1). Its output is Q . The bottom NOR gate has inputs S and Q (which is 0). Its output is \overline{Q} .

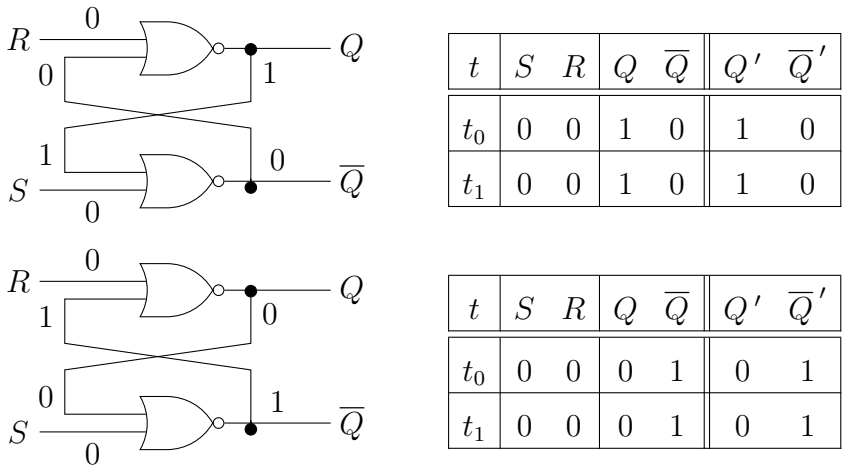
t	S	R	Q	\overline{Q}	Q'	\overline{Q}'
t_0	0	1	0	1	0	1
t_1	0	1	0	1	0	1

Circuit diagram for SR=01 at t_1 . Inputs: $R=1$, $S=0$. Current outputs: $Q=0$, $\overline{Q}=1$. The circuit consists of two NOR gates. The top NOR gate has inputs R and \overline{Q} (which is 1). Its output is Q . The bottom NOR gate has inputs S and Q (which is 0). Its output is \overline{Q} .

t	S	R	Q	\overline{Q}	Q'	\overline{Q}'
t_0	0	1	1	0	0	0
t_1	0	1	0	0	0	1
t_2	0	1	0	1	0	1
t_3	0	1	0	1	0	1

Si la entrada al circuito cambia a $SR = 00$, la tabla característica que obtendríamos sería la que se encuentra a continuación, en la siguiente página.

Tabla 3.10. Tablas características del circuito latch-SR con $SR = 00$



De las tablas características anteriores podemos ver que al cambiar a la entrada $S = 0$ y $R = 0$, el circuito mantiene el estado anterior al que tenía.

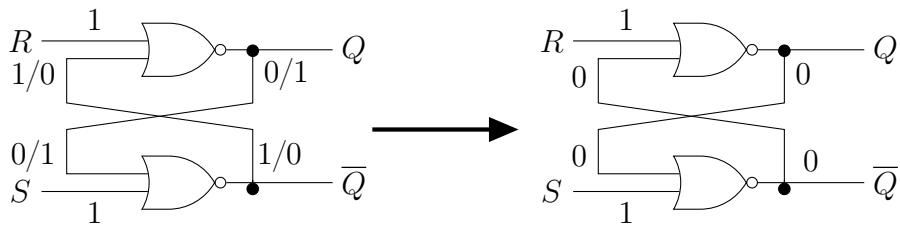
Falta revisar el caso para $SR = 11$. Observemos la tabla 3.11 a continuación.

Tabla 3.11. Tabla característica para la entrada $SR = 11$

t	S	R	Q	\bar{Q}	Q'	\bar{Q}'
t_0	1	1	0	1	0	0
t_1	1	1	0	0	0	0
t_2	1	1	0	0	0	0

t	S	R	Q	\bar{Q}	Q'	\bar{Q}'
t_0	1	1	1	0	0	0
t_1	1	1	0	0	0	0
t_2	1	1	0	0	0	0

En ambos casos, como ambas compuertas *nor* tienen al menos una señal positiva, la salida en la que se estabiliza el circuito es $Q = \bar{Q} = 0$. Pero no podemos tener $Q = \bar{Q}$ por lo que esta combinación es inservible y está prohibida.



Como se ve en estas tablas y en los distintos esquemas de los circuitos *latch-SR*, se puede observar que estos circuitos secuenciales son capaces de recordar el último estado de las entradas en el sentido de determinar cuál de las dos entradas, S o R , fue la última en tener el valor 1.

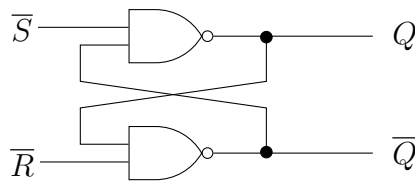
Un latch-SR funciona de manera muy similar a la de un apagador de luz. Si se desea prender el foco se cambia la entrada en S a 1 y de regreso a 0 ($R = 0$). Si se desea apagar el foco se cambia la entrada R a 1 y de regreso a 0 ($S = 0$). Si el foco está prendido no

pasa nada si se cambia la entrada S como acabamos de describir; similarmente si el foco está apagado —el foco está representado por la salida Q —.

La condición normal de entrada para el latch- SR es $SR = 00$. Si se desea cambiar la salida se ingresa a R o a S un 1.

Se deja al lector verificar que el latch- SR se puede implementar usando compuertas *nand*, como se muestra en la figura 3.30.

Figura 3.30. Implementación del latch- SR con compuertas *nand*



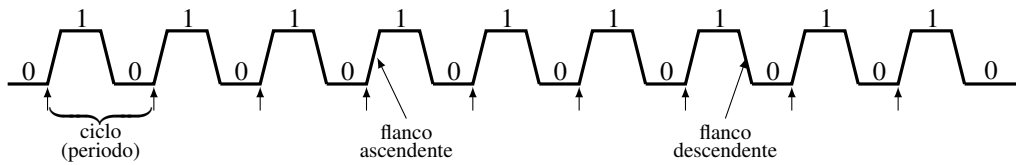
Antes de proseguir con circuitos secuenciales síncronos revisaremos brevemente lo que es un pulso de reloj y cómo funciona.

3.5.2. Pulsos de reloj

El tema de los pulsos de reloj juega un papel importante en los circuitos con que se construyen las computadoras modernas, por lo que introduciremos este tema antes de discutir los distintos tipos de memorias.

El rendimiento de una computadora está determinado por tres factores: el número de instrucciones ejecutadas, la duración del ciclo del reloj y el número de ciclos por instrucción. Si bien el primer factor está determinado por el compilador y/o el programador, los dos factores siguientes dependen del diseño del procesador y, en particular, del reloj con que cuente el procesador.

Los relojes se usan en la lógica secuencial para determinar cuándo es necesario actualizar a un elemento que contiene un estado. Un *reloj* es un dispositivo independiente que corresponde a una señal de control (o entrada) que está siempre presente y que conforme transcurre el tiempo toma valores de 1 y 0, positivo y negativo respectivamente. En general, los relojes están implementados con osciladores que mantienen una frecuencia, cada cierto tiempo cambian de valor. Aunque los cambios de valor no son “instantáneos”, pues se trata de dispositivos eléctricos, se considera 1 cuando se rebasa un cierto umbral y 0 cuando se cae debajo de ese umbral. Un esquema de cómo es interpretado el ciclo de un reloj se presenta en la figura 3.31 que se encuentra en la página que sigue, donde las flechas verticales indican el inicio de un periodo de reloj. Se puede observar que todos los periodos van a durar el mismo tiempo (*cycle time*). Es claro en el esquema que los cambios de señal también consumen tiempo.

Figura 3.31. Esquema de un reloj

Hay dos maneras en que el dispositivo particular va a responder a esta señal:

Activación por valor (disparo por nivel⁹) El circuito espera un nivel particular en la señal. En cuanto la señal alcanza este nivel, el circuito reacciona.

Activación por transición (disparo por flanco¹⁰) El circuito no toma en cuenta el nivel de la señal sino que reacciona cuando *pasa* de un nivel a otro. Si el dispositivo actúa cuando la señal pasa de 0 a 1 se dice que se activa por *flanco ascendente* y si lo hace cuando pasa de 1 a 0 se dice que se activa por *flanco descendente*. En algunos casos el dispositivo está construido para responder a ambos flancos.

Daremos el significado de algunos términos y atributos relacionados con un reloj en una computadora.

Ciclo: Se refiere a a lo que marcamos como un periodo y que constituye desde el inicio del flanco ascendente hasta el final del flanco descendente (que es el inicio del siguiente ciclo).

Tiempo de ciclo: El tiempo que se requiere para que la señal complete un ciclo completo.

Tiempo de ascenso y descenso: En teoría, el cambio de la señal de 0 a 1 o de 1 a 0 es instantáneo, pero en la práctica nada es instantáneo, por lo que estos dos tiempos miden lo que le lleva a la señal cambiar de 0 a 1 o de 1 a 0 respectivamente.

Frecuencia del reloj: También conocido como velocidad del reloj, es el número de ciclos por segundo. Usualmente se mide en MHz o GHz, las unidades estándar para medir frecuencia.

Un reloj es, entonces, una señal continua con un tiempo de ciclo fijo, lo que determina también la frecuencia del reloj. Como se ve en la figura 3.31 en la página anterior, un ciclo se divide en dos porciones: cuando el reloj está arriba y cuando el reloj está abajo. Dependiendo del tipo de activación que se tenga va a quedar determinado de dónde a dónde es cada porción.

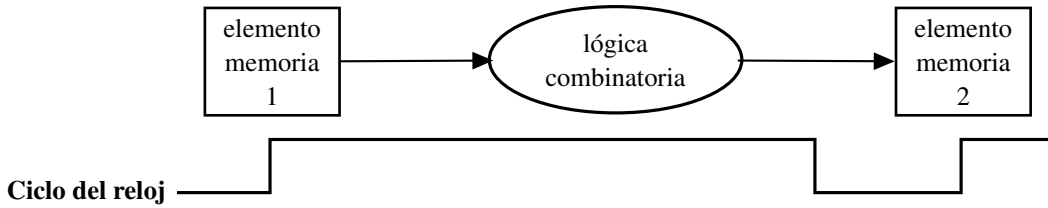
La restricción mayor en un sistema que trabaja con un reloj (*síncrono*) es que las señales que se almacenan en el elemento del estado (la memoria) deben ser *válidas* cuando se presenta la señal activa del reloj (ya sea que se defina por cambio o nivel). Una señal es válida si es estable (esto es, no cambia) y el valor no va a cambiar mientras las entradas no cambien. Las entradas a un bloque lógico combinatorio vienen de un elemento de memoria

⁹En inglés *level triggered*.

¹⁰En inglés, *edge triggered*.

y la salida se escribe en un elemento de memoria. El pulso del reloj determina cuándo los elementos de memoria se actualizan. Esto se puede observar en la figura 3.32.

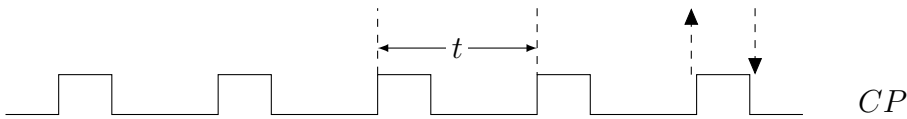
Figura 3.32. Funcionamiento de un sistema síncrono



Si la sincronización es con el cambio de señal se evita que diferentes partes del sistema entren en una especie de carrera. Claro que el ciclo del reloj debe durar lo suficiente para que los valores de entrada sean estables cuando empieza el cambio en la señal del reloj.

En general, los pulsos del reloj se representan como en la figura 3.33.

Figura 3.33. Diagrama de los pulsos de un reloj



3.5.3. Circuitos secuenciales síncronos

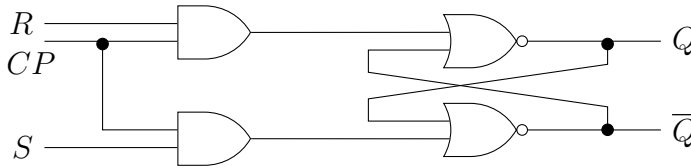
Los latches que vimos son asíncronos, lo que quiere decir que la salida queda determinada poco después de que cambie alguna señal, sea ésta de entrada o de retroalimentación¹¹. Sin embargo, la mayoría de las computadoras hoy en día son síncronas, lo que significa que la salida de los circuitos secuenciales cambia relacionada no nada más con las entradas sino también con el pulso de la señal del reloj del procesador. La idea es que el pulso del reloj *dispare* el proceso de las entradas, al combinarlo con las entradas *R* y *S* del latch. De hecho estamos agregando una señal de entrada al circuito, que está siempre presente oscilando entre habilitada y no habilitada. La salida de un latch o flip flop corresponde siempre al estado almacenado.

Tenemos una versión síncrona de los latches, así como los circuitos flip flop que utilizan, a su vez, latches síncronos. La diferencia fundamental entre los circuitos síncronos latch y flip flop es el punto en el que el reloj provoca que el estado del circuito realmente cambie. En un latch síncrono el estado cambia cuando hay un cambio en alguna de las entradas y mientras el pulso del reloj esté habilitado, mientras que en los circuitos flip flop el cambio de estado se lleva a cabo únicamente en el cambio de señal, cuando la señal dispara.

¹¹De la sección sobre circuitos secuenciales asíncronos debe quedar claro que los cambios en la salida no son instantáneos y pueden requerir de más de una retroalimentación.

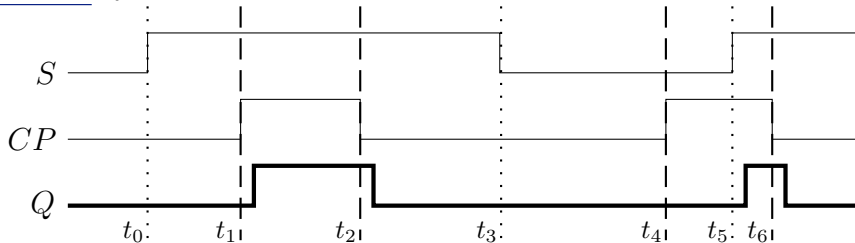
En la figura 3.34 presentamos el diagrama lógico de un latch síncrono. Estamos suponiendo que las señales de S y R son complementarias, así como las salidas Q y \bar{Q} . Llamamos CP a la señal dada por el reloj (*Clock Pulse*).

Figura 3.34. Diagrama lógico de un latch- SR síncrono



Si observamos los cambios de estado que se producen en un latch- SR de acuerdo al pulso del reloj, se observaría como se ve en la figura 3.35.

Figura 3.35. Operación del latch- SR síncrono



En un latch, la salida se reconsidera cada vez que cambia una señal en alguna de las compuertas del circuito. Esto sucede en los puntos marcados con t_i . Las líneas punteadas corresponden a cambios en la señal de S mientras que las líneas quebradas corresponden a cambios en la señal del reloj. En t_0 S toma el valor 1 (sube), pero como la señal del reloj vale 0 (está abajo) no sucede nada en la señal de salida. En t_1 la señal de S se encuentra alta y sube la señal del reloj, provocando que la señal de Q , con un pequeño retardo, también suba. En t_2 baja la señal del reloj, por lo que Q también baja. En t_3 , aunque la señal de S baja, la señal de Q se encuentra ya baja, por lo que no cambia la salida. En t_4 , aunque sube la señal del reloj, la de S permanece baja, por lo que Q permanece baja. Cuando en t_5 sube la señal de S , como la señal del reloj está alta, sube Q . Pero en t_6 , cuando baja la señal del reloj, ocasiona que baje la señal de Q , que se mantiene baja a pesar de que la señal de S se encuentra alta.

En la figura 3.36 de la siguiente página se muestra el diagrama de bloque correspondiente al circuito de la figura 3.34, así como su tabla de verdad.

Para garantizar que la entrada al latch síncrono sea $R = \bar{S}$ se construye lo que se conoce como un circuito latch- D , donde el circuito recibe una sola señal D (S) y esta señal se invierte para incorporarla al circuito en lugar de la señal R (\bar{D}). De esta manera se garantiza que no se puede presentar el caso de $SR = 11$. Los diagramas lógicos y de bloque, así como la tabla de verdad, se encuentran en la figura 3.37, también en la siguiente página.

Figura 3.36. Diagrama de bloque de un latch-*SR* síncrono y su tabla de verdad

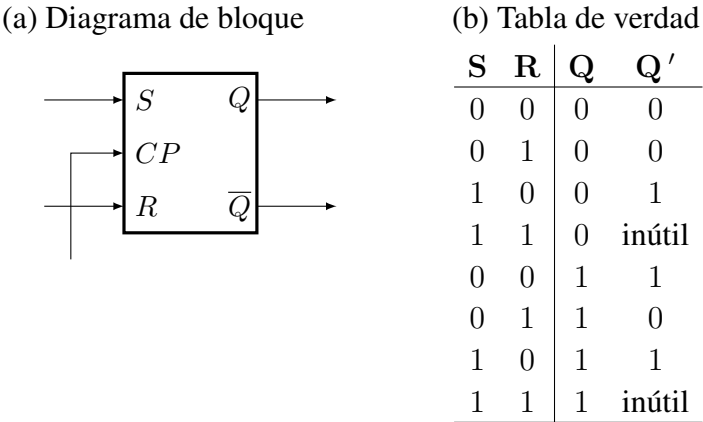
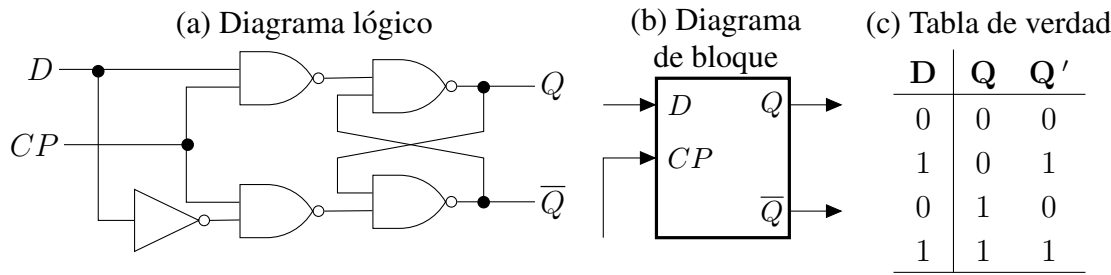


Figura 3.37. Latch-*D* síncrono



Un problema con los latches es que una vez que se dispara la señal del reloj, la salida puede cambiar mientras la señal del reloj se encuentre en 1. Esto hace que la salida no sea confiable respecto a los pulsos del reloj. Por esto, la salida de un latch, para retroalimentarla al mismo circuito, no se puede hacer directamente, ni a través de circuitos combinatorios.

Los circuitos flip flop se construyen para que una vez habilitado el reloj, solo observen sus entradas cuando se realiza el cambio de señal de 0 a 1 en el pulso del reloj. Como acabamos de mencionar, el estado de un circuito flip flop síncrono únicamente observa sus entradas en el cambio del pulso, lo que garantiza que la señal retroalimentada es únicamente la del estado y señal de entrada actual. Existen tres tipos básicos de circuitos flip flop síncronos¹². Empezaremos por el más sencillo, el flip flop *D*.

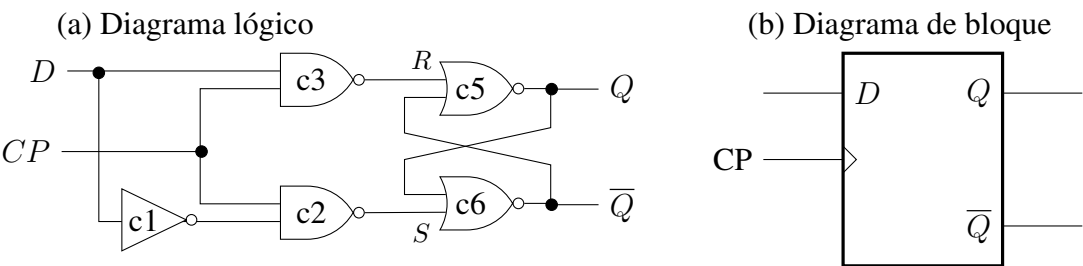
Recordemos que la señal del reloj se dispara cuando cambia de 0 a 1. En realidad, cuando vemos las tablas de los circuitos síncronos, los cambios en la señal sólo tendrán efecto si este cambio se refleja en el cambio del pulso. Si, por ejemplo, la señal de entrada cambiara dos veces durante el ciclo y la señal del reloj no se disparara, no se reflejarían los cambios en las entradas en cuanto a la respuesta del circuito. Por ello, en las tablas de verdad de los circuitos síncronos nunca se toma en cuenta la señal del reloj, sino únicamente el estado actual *Q*, la(s) señal(es) de entrada y, como resultado, el estado resultante *Q'*, en el siguiente instante de tiempo.

¹²En adelante omitiremos que los circuitos son síncronos.

Circuito flip flop D

Un circuito flip flop muy útil es el tipo *D*, que utiliza como bloques a los circuitos latch *D*. Este circuito lo que hace es imitar al circuito flip flop *SR*, pero garantizando que las únicas entradas que se presentan son $SR = 01$ y $SR = 10$ al alimentar la señal *D* y su negación. De esta manera no hay posibilidad de que se presenten las entradas $SR = 00$ o $SR = 11$. Los diagramas lógico y de bloque se encuentra en la figura 3.38.

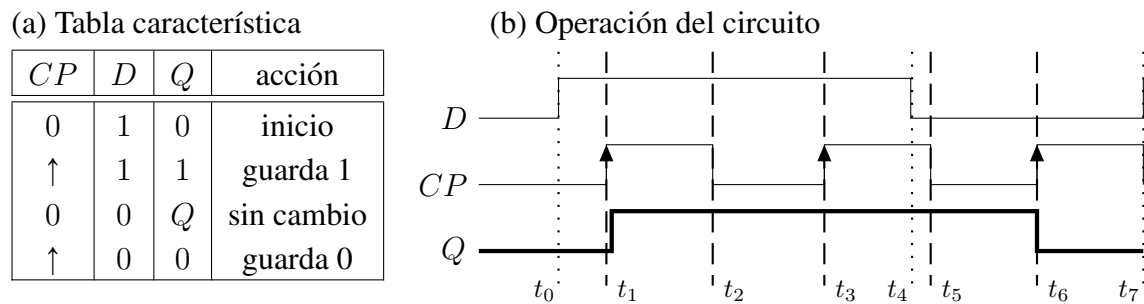
Figura 3.38. Circuito flip flop D



El triángulo que se encuentra dentro del bloque en la subfigura (b), donde termina la línea para el pulso del reloj (CP), indica que el circuito flip flop *D* cambiará de estado únicamente en el flanco ascendente del reloj (\uparrow).

En la figura 3.39 se encuentra la tabla de verdad de este circuito flip flop y una pequeña muestra de su operación.

Figura 3.39. Tabla de verdad y operación del circuito flip flop *D*



Nuevamente se debe recordar que este tipo de circuitos observa sus valores de entrada y estado actual únicamente cuando cambia el pulso del reloj de 0 a 1 y mantiene el valor alcanzado hasta el siguiente flanco ascendente del reloj, independientemente de si cambian o no los valores de entrada. En estas condiciones, si el valor de la entrada *D* es 1, sin importar el estado actual, el siguiente estado será 1 y así permanecerá hasta que, con el pulso del reloj la entrada que se detecte sea 0. En resumen, el circuito flip flop-*D* se usa para capturar la señal de datos (*D*) presente en la línea en el momento en que cambia el pulso de 0 a 1.

En la figura 3.39 observamos que en los momentos en que se consulta el valor de *D* son t_1 , t_3 y t_6 donde se dan los flancos ascendentes. En t_1 se establece el valor de $Q = 1$. En

t_3 como D sigue valiendo 1, el valor de Q se mantiene. En t_6 , sin embargo, el valor de D ya cambió por lo que Q cambia a 0. Los cambios de señal en t_0 , t_4 y t_7 no se reflejan en el momento en que suceden porque no son simultáneos al flanco ascendente del pulso del reloj.

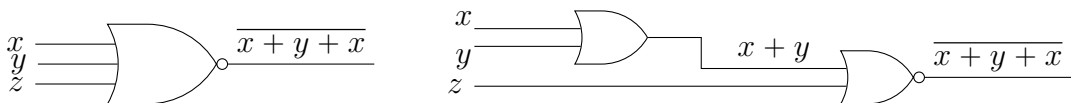
Circuito flip flop JK

Este es un circuito secuencial muy versátil y es una extensión del circuito flip flop- SR , excepto que todos sus estados están perfectamente determinados sin la inestabilidad que tiene el circuito flip flop- SR cuando $SR = 11$. El nombre de JK viene de su creador, Jack Kilby, un ingeniero de Texas Instruments que inventó el circuito integrado.

Al igual que el circuito flip flop- SR , el JK tiene dos entradas, J y K , y un pulso de reloj. Además tiene dos compuertas *and* con tres entradas cada una, a las que retroalimenta las salidas de Q y \overline{Q} . Cambia de estado con el flanco cuando el pulso del reloj cambia de 1 a 0 (flanco descendente o \downarrow). El logro del circuito flip flop JK es resolver el problema de cuando en un circuito flip flop SR se habilitan al mismo tiempo R y S , lo que es una entrada que no se permite pues hace $Q = \overline{Q}$, una salida inconsistente.

Es de notar que hasta ahora hemos utilizado únicamente compuertas de dos entradas. Una compuerta *nor* de tres entradas representa a la expresión $\overline{x + y + z}$. Por asociatividad se puede descomponer en dos capas, $\overline{(x + y) + z}$, por lo que tenemos lo que se muestra en la figura 3.40.

Figura 3.40. Circuitos con compuertas de más de dos entradas



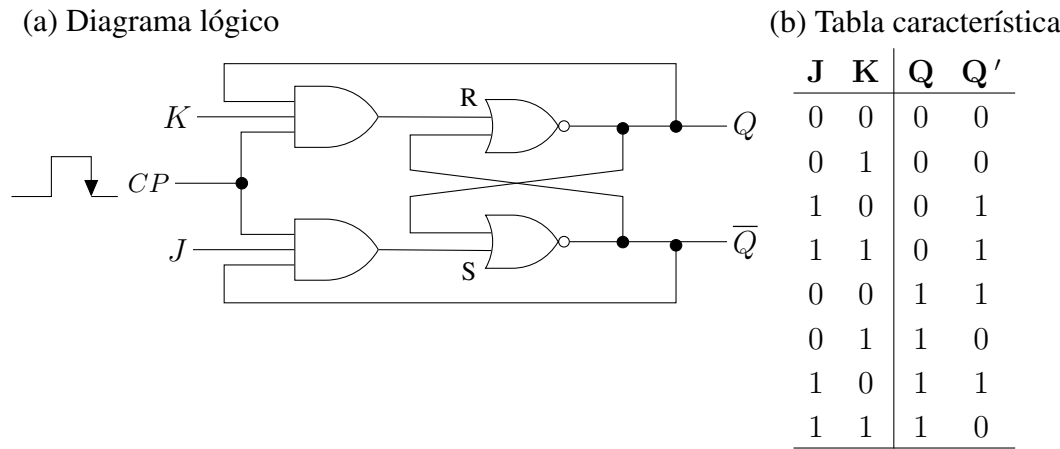
De manera similar, para obtener una compuerta *nand* de tres entradas, simplemente alimentamos dos de las compuertas a una primera compuerta *and* y el resultado de esta compuerta, junto con la tercera entrada, la alimentamos a una compuerta *nand*.

Regresando al tema de los circuitos flip flop K, Un diagrama para este circuito flip flop y su tabla característica se encuentra en la figura 3.41 en la página que sigue.

Podemos pensar en las entradas J y K como *set* y *reset* respectivamente. Los cambios de estado durante el flanco descendente del reloj obedecen el siguiente comportamiento:

J	K	Q	\overline{Q}	Modo
0	0	Q	\overline{Q}	Mantiene el estado anterior
0	1	0	1	Restablecer la señal
1	0	1	0	Establece Q
1	1	\overline{Q}	Q	Intercambia (<i>toggles</i>) las salidas

Figura 3.41. Circuito secuencial flip flop JK y su tabla característica

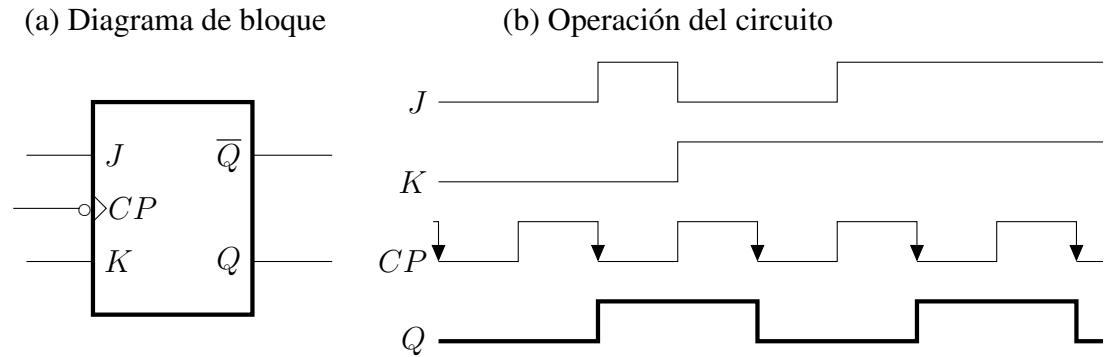


De la tabla anterior, el circuito flip flop JK reacciona de la siguiente manera:

- i. Si $JK = 01$ o $JK = 10$, el estado resultante es $Q\bar{Q} = 01$ o $Q\bar{Q} = 10$ respectivamente.
- ii. Si $JK = 00$, el estado permanece igual.
- iii. Si $JK = 11$, el estado se invierte.

El diagrama de bloque y un esquema de su comportamiento se encuentran en la figura 3.42. El pequeño círculo y el triángulo que corresponden a la señal del pulso del reloj indican que los cambios se hacen en el flanco descendente.

Figura 3.42. Diagrama de bloque y comportamiento de un circuito flip flop JK

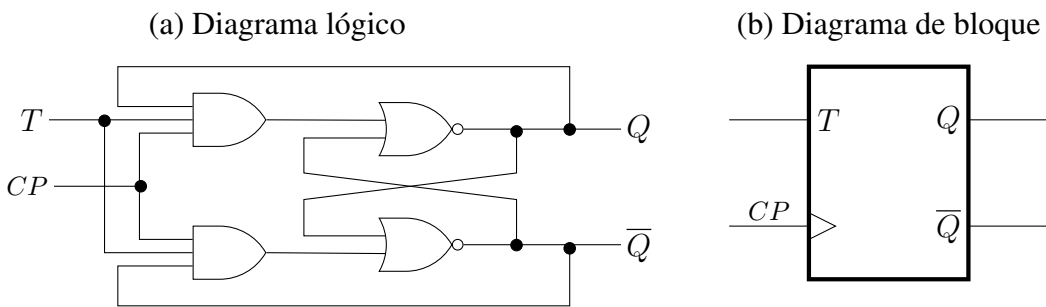


Se recomienda al lector observar la operación del circuito respecto a la tabla característica, insistiendo en que las señales se observan únicamente en el flanco descendente del reloj.

Circuito flip flop T

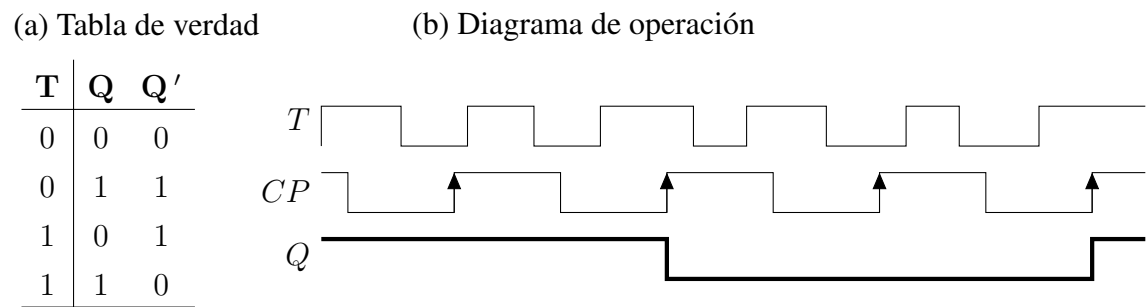
El circuito flip flop T es una versión simplificada del circuito flip flop JK , haciendo que la misma entrada sea alimentada tanto a J como a K . Su única función es cambiar de estado en cada ciclo del reloj (**Toggle**) siempre que la entrada esté habilitada (en el flanco ascendente del reloj) y lo consigue juntando las dos entradas en una sola. Los diagramas lógico y de bloque se pueden ver en la figura 3.43. El circuito reacciona en el flanco ascendente del pulso, como se puede observar en su diagrama de bloque.

Figura 3.43. Circuito flip flop T



Podemos pensar en un flip flop tipo T como un contador de un bit, por lo que en cada pulso del reloj simplemente invierte su entrada. Es muy útil para construir contadores o divisores del ciclo del reloj. Por ejemplo, cuando T se mantiene en 1, si la frecuencia del reloj es de, por ejemplo, 4MHz, la frecuencia que se obtiene de la salida del flip flop T es de 2MHz. La tabla de verdad para este circuito se encuentra a continuación (recordar que la señal T se detecta únicamente en el cambio del pulso del reloj de 0 a 1, aunque pudiera estar construido para detectar el cambio del pulso del reloj de 1 a 0). Su tabla de verdad y un esquema de su operación se encuentra en la figura 3.44.

Figura 3.44. Tabla de verdad y esquema de operación del circuito flip flop T



Veremos a continuación algunos de los usos de los circuitos flip flop en la construcción de computadoras.

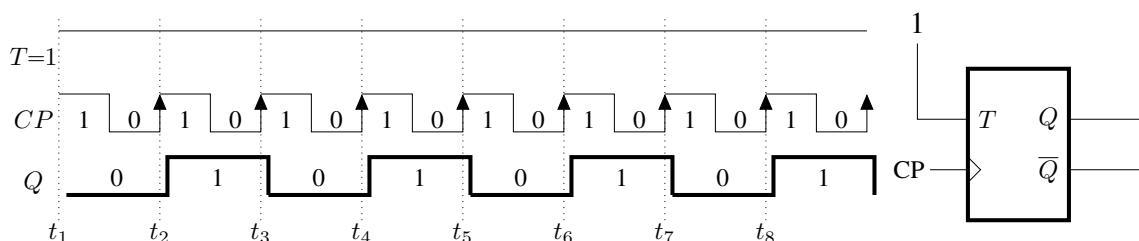
Contadores

Por la característica que tienen los circuitos secuenciales de poder almacenar información, se usan tanto para distintos tipos de registros como para contadores. Únicamente revisaremos los contadores en esta ocasión.

Los contadores son, en sí mismos, circuitos secuenciales que se construyen con grupos de circuitos flip flop. Pueden ser síncronos, donde la señal del reloj llega a todos los circuitos al mismo tiempo, o asíncronos, donde cada circuito flip flop recibe su señal de reloj dependiendo de la posición que ocupen en el contador. Por ejemplo, un contador binario puede contar el número de pulsos que se le dan de entrada. La mayoría de los contadores que vamos a encontrar están implementados con circuitos flip flop JK , pero se les alimenta una única señal a ambas entradas, por lo que en la práctica se está trabajando con circuitos flip flop T , que corresponden a los circuitos flip flop JK donde la señal de entrada es $J = K = T$. Para simplificar un poco los diagramas utilizaremos circuitos flip flop T en lo que sigue. Veremos primero los contadores síncronos.

Sabemos que un circuito flip flop T simplemente invierte la señal cuando ésta es positiva y se presenta el flanco ascendente del reloj. Si consideramos una entrada constante positiva (1 o +5v), tendremos que el circuito flip flop T invertirá su señal con cada flanco ascendente, como se muestra en el diagrama en la figura 3.45. Cada vez que se presenta el flanco ascendente del reloj, con un pequeño retardo, la señal de Q se complementa (invierte).

Figura 3.45. Efecto de una entrada positiva a un circuito flip flop T



El circuito de la figura 3.45 en la página anterior es un contador de un bit. Como después de alcanzar el máximo regresa a 0, se conoce como un contador cíclico. En t_2 , $T = 1$, tenemos el flanco ascendente del reloj, por lo que con un pequeño retardo la salida Q pasa de 0 a 1. En t_3 se presenta el flanco ascendente del reloj con $T = 1$, por lo que la señal de Q se invierte y pasa a 0. Cada vez que se presenta el flanco ascendente del pulso del reloj, como la señal de T es 1, la salida en Q se invierte. Podemos observar que el circuito flip flop T divide en 2 la frecuencia del reloj.

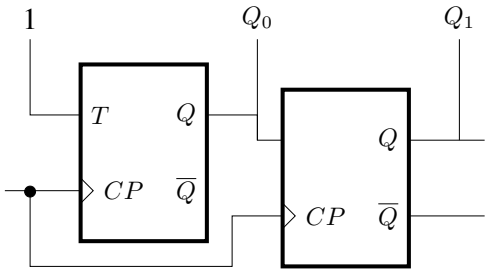
Para construir un contador de dos bits que cuente de 0 a 3, podemos conectar dos circuitos flip flop T . El primero recibe la señal positiva de T , como el que vimos arriba, y corresponde al bit menos significativo del número. Para el segundo bit, el más significativo, usamos un segundo circuito flip flop T . Para sincronizarlos aplicamos la misma señal de reloj a ambos. Queremos que las señales de salida pasen por el ciclo de la tabla 3.12.

Tabla 3.12. Contador de dos bits

Q_0	Q_1	<i>binario</i> (Q_1Q_0)	<i>decimal</i>
0	0	00	0
1	0	01	1
0	1	10	2
1	1	11	3
0	0	00	0

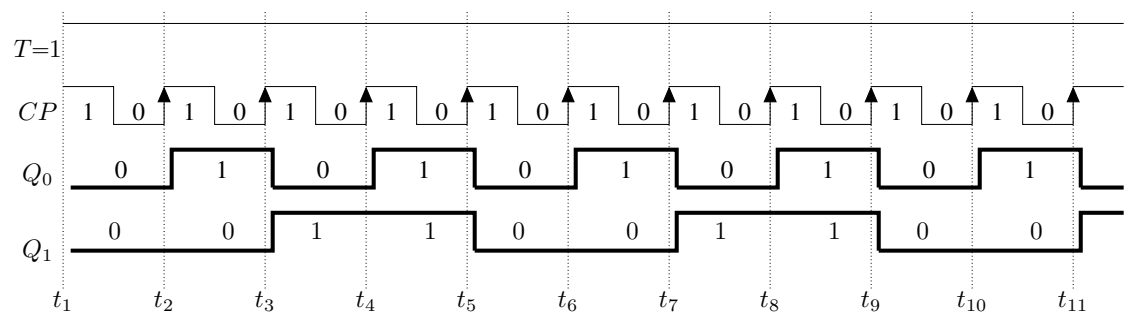
Q_0 está controlado exclusivamente por el reloj y corresponde, simplemente, a la salida del primer circuito flip flop T que invierte su entrada cada vez que aparece el flanco ascendente del reloj. Cuando este primer flip flop está en 1 y recibe el flanco ascendente del reloj para regresar a 0, es cuando la entrada al segundo flip flop debería pasar a 1. Esto se consigue alimentándole al segundo circuito flip flop la señal de salida del primero. Veamos el diagrama de bloque en la figura 3.47.

Figura 3.46. Contador cíclico de dos bits



En la figura 3.47 mostramos el esquema de operación de un contador de dos bits síncrono.

Figura 3.47. Contador cíclico de dos bits



Si observamos el diagrama del comportamiento vemos que el contador está actuando de acuerdo a lo que se espera de él. Pasa por los valores 00, 01, 10, 11, 00, 01, 10, 11, En t_1 , como Q_0 no es 1, no hay cambio en Q_1 . En t_2 , debido al retraso en el cambio de la

salida, Q_0 vale todavía 1 cuando se presenta el flanco ascendente del reloj, por lo que Q_1 cambia a 1. En t_3 no cambia porque Q_0 vale 0 todavía. En t_4 , Q_1 cambia a 0 porque el valor de Q_0 es todavía 1. Si observamos el diagrama de operación, la señal de Q_1 cambia de 0 a 1 en t_6 y t_{10} , que es cuando Q_0 pasa de 1 a 0.

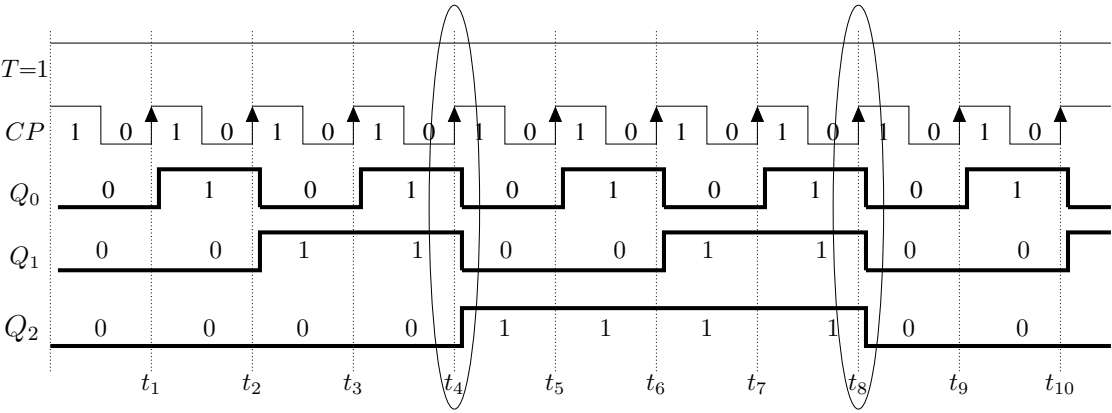
El primer circuito flip flop está controlado por el reloj y el segundo por la salida del primero. Los cambios en el circuito flip flop se llevarán a cabo en la subida de la señal (de 0 a 1).

Si deseamos hacer un contador de tres bits, los estados por los que debe pasar el sistema son los siguientes:

Q_0	Q_1	Q_2	<i>binario</i> ($Q_2Q_1Q_0$)	<i>decimal</i>
0	0	0	000	0
1	0	0	001	1
0	1	0	010	2
1	1	0	011	3
0	0	1	100	4
1	0	1	101	5
0	1	1	110	6
1	1	1	111	7
0	0	0	000	0

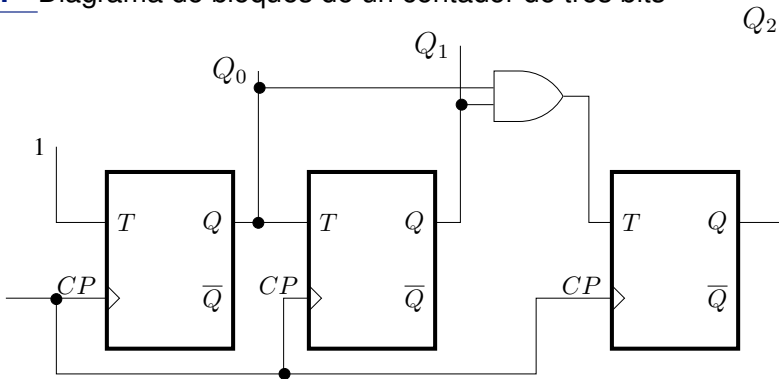
Q_2 pasa de 0 a 1 cuando tanto Q_0 como Q_1 ambos cambian a 0. Q_2 regresa a 0 nuevamente cuando tanto Q_0 como Q_1 valen 1 y tenemos el flanco ascendente del reloj. Este contador utiliza tres circuitos flip flop, excepto que la entrada de T para el tercer circuito debe ser la conjunción de Q_0 y Q_1 . El diagrama del comportamiento de este contador es como se ve en la figura 3.48.

Figura 3.48. Comportamiento de un contador cíclico de tres bits



Del diagrama de comportamiento podemos ver que el cambio de señal de Q_2 se da cuando Q_0 y Q_1 están por cambiar a 0 – el cambio no es instantáneo con el pulso del reloj, por lo que estas dos señales valen 1 en ese momento—. Estos cambios se dan en t_4 y t_8 (aparecen rodeados de una elipse en la figura). De lo anterior, la entrada T a Q_2 debe ser la conjunción de Q_0 y Q_1 , como se ve en la figura 3.49 en la siguiente página. Como todos los circuitos flip flop reciben la misma señal del reloj, el retardo que observamos es el mismo para todos.

Figura 3.49. Diagrama de bloques de un contador de tres bits



Si quisiéramos que el contador fuera de cuatro bits (0 al 15) le agregaríamos otro circuito flip flop a la derecha de Q_2 . Queremos que cambie de 0 a 1 cuando las tres posiciones menos significativas sean 1, similarmente a como cambia Q_2 . Las transiciones por las que pasa el contador se encuentran en la tabla 3.13 en la siguiente página. El cambio de 0 a 1 se daría en t_8 en la figura 3.48 únicamente, que es cuando las tres señales valen 1 y se presente el flanco ascendente del reloj.

Similarmente a como hicimos para tres bits, tendríamos que hacer una conjunción de Q_0 , Q_1 y Q_2 para alimentarla en T . El diagrama del contador de cuatro bits se encuentra en la figura 3.50 a continuación. El esquema de su comportamiento se encuentra en la figura 3.51 en la siguiente página.

Figura 3.50. Diagrama de bloques de un contador de cuatro bits

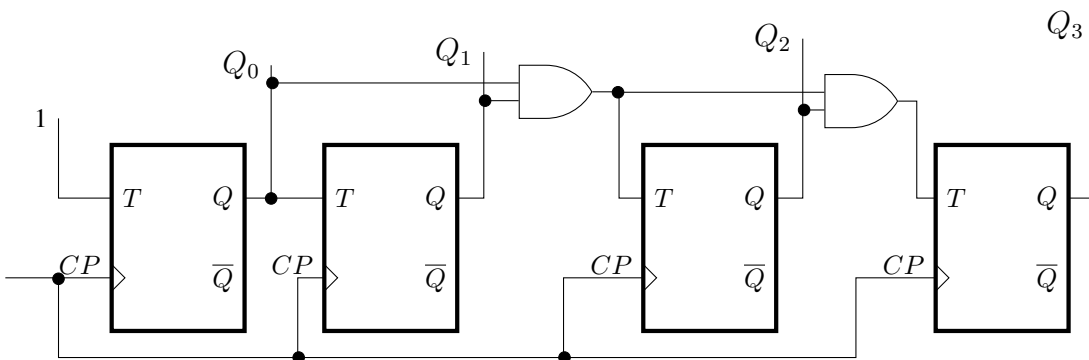
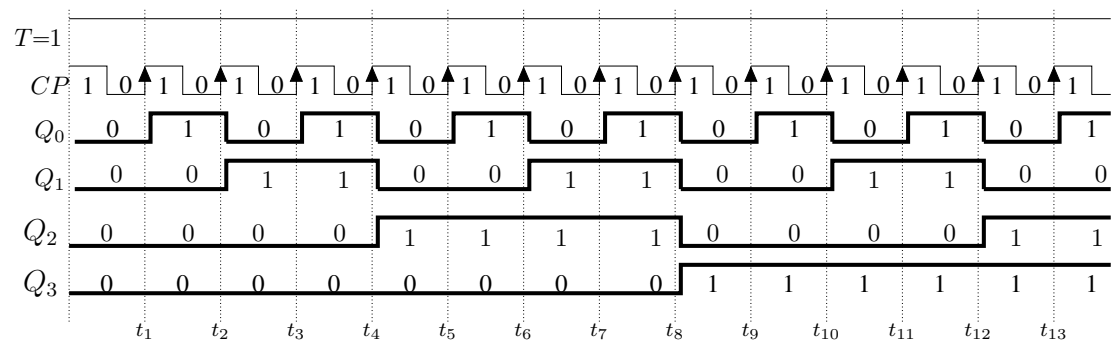


Tabla 3.13. Transiciones en un contador de cuatro bits

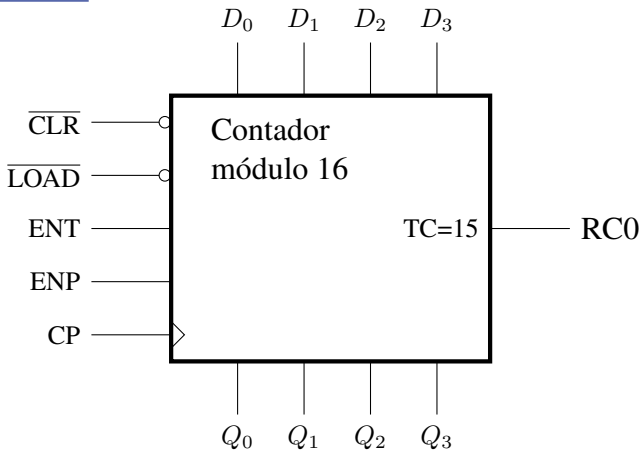
Q_0	Q_1	Q_2	Q_3	<i>binario</i> ($Q_3Q_2Q_1Q_0$)	<i>decimal</i>
0	0	0	0	0000	0
1	0	0	0	0001	1
0	1	0	0	0010	2
1	1	0	0	0011	3
0	0	1	0	0100	4
1	0	1	0	0101	5
0	1	1	0	0110	6
1	1	1	0	0111	7
0	0	0	1	1000	8
1	0	0	1	1001	9
0	1	0	1	1010	10
1	1	0	1	1011	11
0	0	1	1	1100	12
1	0	1	1	1101	13
0	1	1	1	1110	14
1	1	1	1	1111	15
0	0	0	0	0000	0

Figura 3.51. Comportamiento de un contador cíclico de cuatro bits



Para conectar contadores en cascada (base 16) podemos usar una tableta (*chip*) (74163) cuya configuración se ve en la figura 3.52 de la siguiente página.

Figura 3.52. Configuración de la tableta 74163



Las líneas a la izquierda de la tableta, así como D_0 a D_3 son entradas, mientras que Q_0 a Q_3 y RC0 son salidas. ENT y ENP sirven para habilitar a la tableta. CLR, como su nombre lo indica, sirve para poner en ceros el contador. Si al iniciar el contador la línea marcada con $\overline{\text{LOAD}}$ recibe un pulso, se cargará el valor inicial dado por D_0 a D_3 . Las líneas Q_0 a Q_3 proporcionan el estado del contador. La línea RC0 emite un pulso cada vez que el contador pasa de 1111 a 0000, lo que permite conectarlo a otra tableta igual y constituye el acarreo.

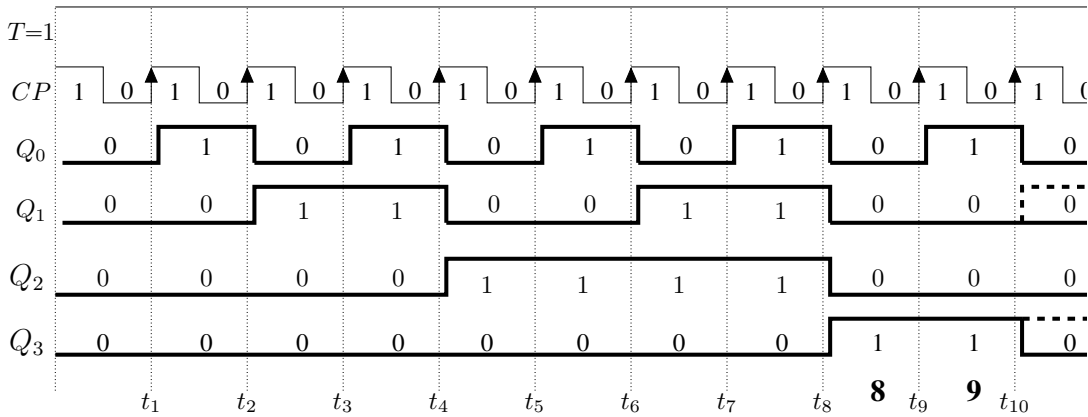
Supongamos ahora que queremos un contador que cuente de 0 a 9. Requerimos de cuatro circuitos flip flop T porque el número 9 se forma con cuatro bits, pero ahora las transiciones quedarían como se muestran en la tabla 3.14.

Tabla 3.14. Transiciones en un contador de cuatro bits que cuenta de 0 a 9

Q_0	Q_1	Q_2	Q_3	binario ($Q_3Q_2Q_1Q_0$)	decimal
0	0	0	0	0000	0
1	0	0	0	0001	1
0	1	0	0	0010	2
1	1	0	0	0011	3
0	0	1	0	0100	4
1	0	1	0	0101	5
0	1	1	0	0110	6
1	1	1	0	0111	7
0	0	0	1	1000	8
1	0	0	1	1001	9
0	0	0	0	0000	0

Para conseguir esto debemos detectar la presencia del 9 (1001) y regresar a 0 (0000). Observemos el comportamiento que debe tener en la figura 3.53. Queremos que en t_{10} pase a 0 (0000) y no a 10 (0101), como lo hace el contador original. Se muestra el viejo comportamiento con línea quebrada y sustituimos los unos por ceros.

Figura 3.53. Comportamiento de un contador cíclico de 0 a 9

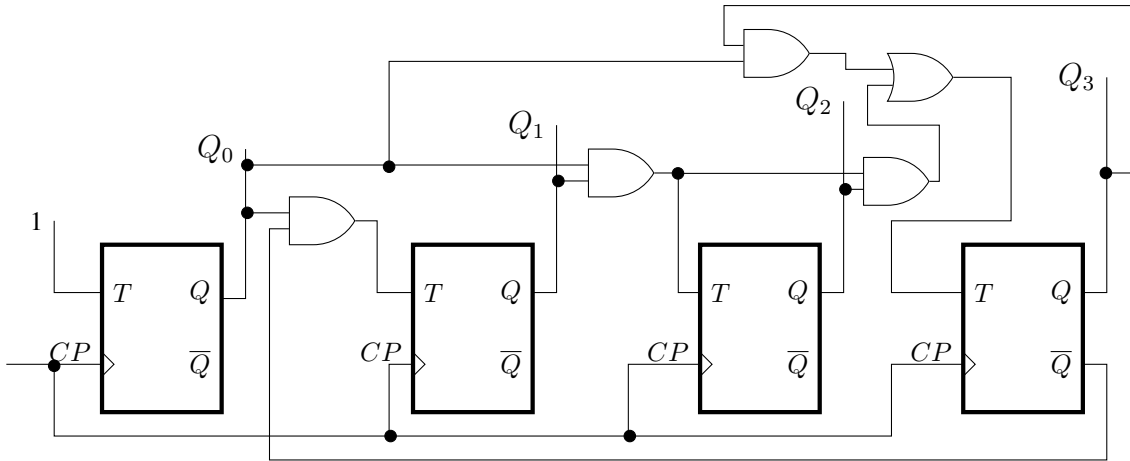


Donde deseamos hacer los cambios es en t_{10} . Para Q_0 no hay cambios en las entradas, ya que naturalmente se invierte y pasa a 0 para el siguiente flanco del reloj. Con Q_1 no está tan fácil, pues en el contador normal Q_1 se invertiría con el siguiente flanco y se pondría en 1, cuando deseamos que no se invierta y quede en 0. Observemos entre el 0 y el 9, cuándo deseamos que la señal T a Q_1 hace que Q_1 se invierta, para garantizar que lo siga haciendo entre 0 y 9, pero que no lo haga para 10 (que su entrada a T sea 0). Si observamos el diagrama, Q_1 se invierte en t_2, t_4, t_6 y t_8 (la inversión en t_{10} es la que no queremos), lo que quiere decir que lo hace cuando Q_0 es 1 y Q_3 es 0, por lo que modificamos la entrada T a Q_1 para que no invierta (sea 0) cuando Q_3 sea 1. Esto lo conseguimos agregando una compuerta *and* en la entrada T a Q_1 donde alimentamos a T la señal $Q_0\overline{Q_3}$. Afortunadamente, $\overline{Q_3}$ es producida por el cuarto circuito flip flop, por lo que simplemente tomamos esa señal. Para Q_2 tampoco tenemos problema pues va a permanecer en 0 y no se va a invertir. Para Q_3 tenemos que forzar que se invierta (que la señal de T sea 1). Para ello analizamos los puntos en los que Q_3 se invierte entre 0 y 9, y esto es cuando $Q_0Q_1Q_2 = 1$ en t_8 . Pero ahora queremos que se invierta también cuando $Q_0Q_3 = 1$, por lo que tenemos que alimentar en T la expresión $Q_0Q_1Q_2 + Q_0Q_3$. El diagrama del circuito queda entonces como se ve en la figura 3.54 en la siguiente página.

Otro tipo importante de contadores son los contadores asíncronos, donde el único circuito flip flop que recibe directamente la señal del pulso del reloj es el correspondiente a Q_0 . La respuesta de Q_0 se alimenta, después de ser procesada, como si fuera la señal del reloj de Q_1 y así sucesivamente. A las entradas para T se les alimenta un 1 en todos los circuitos utilizados. Eso causa en este tipo de contadores un efecto como de ola y los retrasos se van acumulando conforme se avanza en los circuitos flip flop, por lo que no cambian to-

dos los circuitos al mismo tiempo su salida. Por ello a este tipo de contadores se les conoce como contadores de ola (*ripple counters*). No ahondaremos en este tema por salirse de los objetivos de este texto.

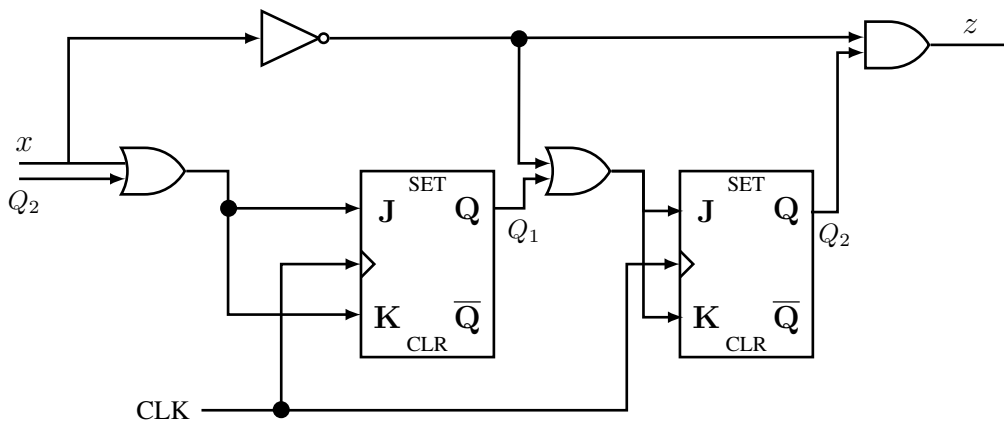
Figura 3.54. Diagrama de bloques de un contador de cuatro bits que va de 0 a 9



Ejercicios

3.5.1.- Verificar que el latch–SR se puede implementar con compuertas *nand*, obteniendo el mismo comportamiento.

3.5.2.- Tenemos el siguiente circuito secuencial:



Elabora el diagrama del comportamiento con la sucesión de valores de $x = 1001$ y valores iniciales $Q_1 = Q_2 = 0$.

3.5.3.- Usando circuitos flip flop, elabora un contador que cuente módulo 5. ¿De cuántos bits es el contador?

Lógica de predicados | 4

Hasta ahora hemos utilizado fórmulas de la lógica proposicional siempre que se trata de representar proposiciones en español; esto es, enunciados que son falsos o verdaderos. Analicemos los siguientes enunciados:

- Todo plátano es amarillo.
- Algunas especies de pájaros migran.
- Todos los vaqueros usan sombrero.
- Ningún perro maúlla.
- Baja California Sur es el único estado de la República Mexicana con mar en tres de sus cuatro bordes.

Se puede observar que todos y cada uno de los enunciados anteriores son proposiciones, pues tienen un valor de falso o verdadero. Sin embargo, difieren de los estudiados anteriormente pues no reconocemos en los enunciados palabras correspondientes a conectivos lógicos, por lo que la única manera que tenemos, por el momento, de formalizarlos, es simplemente con una sola variable proposicional asignada a todo el enunciado.

El lenguaje de la lógica proposicional estudiado en el capítulo anterior no tiene suficiente poder expresivo para analizar proposiciones y argumentos que requieren de una clase de enunciados como los anteriores, que contienen referencias a colectividades de objetos.

Considérese por ejemplo el siguiente razonamiento:

Algunas personas van al teatro. Todos los que van al teatro se divierten. De manera que algunas personas se divierten.

La intuición dice que el argumento es correcto. Sin embargo la representación correspondiente en lógica proposicional es:

$$p, q / \therefore r \quad ; \text{Incorrecto!}$$

Esta situación nos permite concluir únicamente que el argumento en lógica proposicional es incorrecto en el sentido de que la conclusión no es consecuencia lógica de las premisas. Sin embargo, a partir de este hecho no es posible concluir que el argumento en lenguaje natural sea incorrecto, pues podría ser que lo sea con base en ciertos principios lógicos más fuertes. Tal posibilidad requiere el desarrollo de un lenguaje de especificación formal más poderoso, así como una lógica adecuada al mismo, llamada lógica de predicados.

4.1. Predicados

Consideremos los siguientes enunciados:

- Cualquier empleado tiene un jefe.
- Algunos programas usan ciclos.
- Hay una lista que está ordenada.

Como acabamos de argumentar, para representar a cada uno de estos enunciados la única forma de hacerlo, con las herramientas que tenemos hasta el momento, es mediante fórmulas proposicionales atómicas; es decir, mediante una simple variable proposicional para cada uno de ellos. De los dos ejemplos anteriores vemos que esta representación no es adecuada, ya que no es capaz de reflejar la estructura interna del enunciado, algo de lo que no debemos sustraernos. Buscamos una herramienta lógica que tome en cuenta, de alguna manera, a esa estructura interna. Por ejemplo, el enunciado *algunos programas usan ciclos* trata acerca de programas, ciclos y la acción de usar. Estas componentes de la estructura interna de un enunciado se clasifican como *individuos* (u objetos) y propiedades (o relaciones) atribuibles a los individuos; a estas últimas las llamamos *predicados*. Tanto los individuos como los predicados se definen en un contexto particular dependiendo del problema que queramos representar. Este contexto se conoce como *universo de discurso*, el cual es una colección de todas las personas, ideas, cosas, estructuras de datos, etcétera, necesarios para analizar una fórmula o argumento lógico.

Veamos algunos ejemplos para hacer la distinción entre predicados e individuos en universos de discurso. En cada caso los individuos se encuentran encerrados en una caja y los predicados son las partes del enunciado que describen las relaciones entre ellos, así como las acciones que los individuos llevan a cabo; por ejemplo, *ser colegas*; *ser padre de*; *ser canario*; *ser la suma de*; *usar*; *visitar*; *ir*; *jugar*; etcétera.

- El universo de discurso son personas:

- Isabel y María son colegas.
- Pedro es el padre de Juan.

- El universo de discurso son los animales:

- `Piolín` es un canario.
- `Claudio` es un gallo.

- El universo son números:

La suma de `2` y `3` es `5`.

El producto de `10` y `2` es negativo.

- El universo consta de lenguajes de programación, algoritmos y programas:

- `Haskell` es un lenguaje funcional con el que se puede escribir el algoritmo `Quicksort` en una línea.
- Este `programa` en Java usa `clases`.

- El universo puede constar de diversas clases de individuos, como en el caso en que los siguientes enunciados se usen en un mismo contexto:

- La infanta `Christian` visita `museos`.
- `El teatro` al que la condesa `Karla Michelle` fue ayer tiene asientos cómodos.
- Su majestad `Martha Elena III` y el perro imperial `Maya` juegan en el `jardín` de palacio.

En el caso de estos enunciados, el universo tiene al menos personas, animales y lugares.

Aunque parezca que podemos utilizar lógica proposicional para representar a los individuos y relaciones, esto no es así. Por ejemplo, no tiene sentido decir que el primer enunciado se formaliza como $p \wedge q$ donde p significa *Isabel es colega* y q significa *María es colega*, ya que la conjunción de estos dos enunciados no consigue explicar la relación de colegas *entre* Isabel y María.

En *lógica de predicados* utilizamos la notación $P(t_1, t_2, \dots, t_n)$ para describir que la propiedad o relación P se da entre los individuos t_1, t_2, \dots, t_n . Expresemos algunos de los ejemplos que dimos arriba con esta nueva notación:

- $Colegas(Isabel, María)$, con $P = \text{ser colega}$, $t_1 = \text{Isabel}$ y $t_2 = \text{María}$.
- $Padre(Pedro, Juan)$, con $P = \text{padre de}$, $t_1 = \text{Pedro}$ y $t_2 = \text{Juan}$.
- $Canario(Piolín)$, con $P = \text{ser canario}$ y $t_1 = \text{Piolín}$.
- $Suma(2, 3, 5)$, con $P = \text{suma}$, $t_1 = 2$ y $t_2 = 3$ son los sumandos y $t_3 = 5$ es el resultado.

Podemos ver en estos ejemplos que cada predicado P recibe un número distinto de argumentos de entrada t_1, \dots, t_n —escribimos el nombre de los predicados con mayúsculas para distinguirlos de las funciones—. Al número de argumentos de un predicado le llamamos *índice* o *aridad* del predicado. También vemos que el orden, en muchos de ellos, es impor-

tante. Por ejemplo, el predicado *Padre* tiene un significado muy distinto si cambiamos el orden de los argumentos, lo mismo que el predicado *Suma*. Una vez que se ha definido un predicado con un determinado índice, queda prohibido cambiarle el índice posteriormente. Por ejemplo, en el primer predicado, *Colegas*, el índice es 2, lo cual impide formar expresiones como *Colegas*(Juan, Lupe, Rosa), aun cuando esto tenga sentido desde nuestra intuición. Si se desea utilizar un número de argumentos distinto al definido inicialmente por el índice, es necesario definir otro predicado; por ejemplo *Colegas'*(Juan, Lupe, Rosa). Los predicados de índice uno, es decir de un solo argumento, reciben el nombre específico de *propiedades* mientras que los de un índice mayor a uno son *relaciones*.

4.1.1. Variables y cuantificadores

Hasta ahora el uso de predicados en lugar de variables proposicionales podría parecer simplemente otra manera de escribir enunciados. Por ejemplo, la proposición

Anastasia recita poesía nórdica

se representa con predicados como *Recita*(Anastasia, poesía nórdica), lo cual parece ser simplemente una manera distinta de escribir el mismo enunciado en español. La principal diferencia es que el predicado puede cambiar de argumentos, como en

Recita(Licantro, odas en sánscrito).

Más aún, podemos sustituir individuos por variables, como en *Recita*(x , y). De esta manera podemos definir predicados de manera más formal, como los que siguen:

- $F(x, y)$ significa que x es padre de y .
- $E(x)$ significa que x es un estudiante.
- $J(x, y)$ significa que x es más joven que y .

Es importante remarcar que los nombres de las variables no importan siempre y cuando se usen de forma consistente. Sin embargo, obsérvese que las expresiones anteriores no corresponden a proposiciones, puesto que los valores de x e y están indeterminados, por lo que resulta imposible verificar si el predicado *Recita* se cumple (es verdadero). Las variables juegan el papel de representantes de valores concretos, como un estudiante, un número o un programa particular. Obsérvese entonces que un mismo predicado puede representar un número potencialmente infinito de proposiciones, una por cada individuo que pueda tomar el lugar de cada una de las variables del predicado.

Consideremos ahora los siguientes enunciados:

- *Hay un gato rayado.*
- *Algunas personas van al teatro.*
- *Todos los programas en Java usan clases.*
- *Todos los estudiantes trabajan duro.*

- *Algunos* estudiantes se duermen en clase.
- *Ocho de cada diez* gatos lo prefieren.
- *Nadie* es más tonto que yo.
- *Al menos seis* estudiantes están despiertos.
- *Hay una infinidad* de números primos.
- *Hay más* computadoras PC que Mac.

Todos estos enunciados tienen en común el hecho de que no involucran a ningún individuo en particular. Aun cuando tenemos predicados a nuestra disposición, necesitamos un mecanismo para formalizar las partes de los enunciados que se refieren a una cantidad, como *todos*, *algunos*, *hay*, *nadie*, *cualquiera*, A estas cantidades las llamamos *cuantificadores*.

Por ejemplo, para el enunciado *Todos los estudiantes son más jóvenes que algún profesor*, entendiendo que el universo de discurso son las personas de la Facultad de Ciencias, sería inoperante escribir todos los posibles predicados para estudiante, profesor y ser más joven, $E(\text{Karla})$, $E(\text{Hugo})$, . . . , $P(\text{Elisa})$, $P(\text{Favio})$, $J(\text{Karla}, \text{Favio})$, Más aún, en algunos casos esto resulta imposible, como con la frase *hay una infinidad de números primos*.

Este problema se soluciona al emplear operadores de cuantificación sobre individuos indeterminados, \forall (se lee *para todo*) y \exists (se lee *existe*), los cuales siempre van seguidos de una variable que representa a los individuos de la colectividad que se está especificando. Por ejemplo, para decir *todos hablan español* escribimos $\forall x E(x)$, donde $E(x)$ significa que x *habla español*. Similarmente, si $C(x)$ significa que x es cuervo, entonces, para especificar que *hay un cuervo*, escribimos $\exists x C(x)$. Adicionalmente, usando cuantificadores en combinación con la lógica proposicional, podemos representar enunciados más complejos, como por ejemplo *todos los estudiantes son más jóvenes que algún profesor*, cuya especificación es como sigue:

$$\forall x (E(x) \rightarrow \exists y (P(y) \wedge J(x, y))),$$

donde se está considerando que $P(x)$ significa x es profesor; $E(x)$ significa x es estudiante y $J(x, y)$ significa x es más joven que y .

A continuación vamos a definir el lenguaje formal de la lógica de predicados que incluye todos los elementos discutidos hasta ahora, para después volver al tema de especificación formal.

4.2. Sintaxis de la lógica de predicados

En esta sección definimos formalmente lo que se conoce como un lenguaje de la lógica de predicados de primer orden, el cual, a diferencia del caso proposicional, varía dependiendo del universo de discurso particular, y de las relaciones y propiedades de los individuos que se deseen especificar.

4.2.1. Términos

Los *términos* son la categoría sintáctica que representa individuos del universo de discurso.

Definición 4.1 (término) Un *término* es una constante, una variable o bien un símbolo de función aplicado a otros términos.

Los términos se generan mediante la siguiente gramática. En los casos en que aparezca una coma (“,”), ésta es parte de la sintaxis. El metasímbolo “...” significa “más de los anteriores” y no forma parte de los símbolos terminales de la gramática.

$\text{term} ::= \text{var}$	(4.1)	$\text{const} ::= c$	(4.10)
$\text{term} ::= \text{const}$	(4.2)	$\text{const} ::= \dots$	(4.11)
$\text{term} ::= \text{func}(\text{lista-de-term})$	(4.3)	$\text{func} ::= f$	(4.12)
$\text{var} ::= x$	(4.4)	$\text{func} ::= g$	(4.13)
$\text{var} ::= y$	(4.5)	$\text{func} ::= h$	(4.14)
$\text{var} ::= z$	(4.6)	$\text{func} ::= \dots$	(4.15)
$\text{var} ::= \dots$	(4.7)	$\text{lista-de-term} ::= \text{term}$	(4.16)
$\text{const} ::= a$	(4.8)	$\text{lista-de-term} ::= \text{term}, \text{lista-de-term}$	(4.17)
$\text{const} ::= b$	(4.9)		

Cada símbolo de la categoría *func* tiene asociado un número fijo de argumentos (el índice o aridad del símbolo). A veces escribimos $f^{(n)}$ para indicar que el símbolo f tiene índice n .

Veamos a continuación algunos ejemplos.

Ejemplo 4.1. Supongamos que el universo consta de los países del mundo y sus ciudades.

- Las variables x e y denotan a países cualesquiera.
- La constante a denota a Alemania y la constante b a Brasil.
- El símbolo funcional f de índice 1 denota a la operación que recibe un país y devuelve su ciudad capital. Es decir, $f(x)$ es la capital de x . Esto es posible dado que cada país tiene una única capital, de manera que dicha asociación es una función bien definida (es *funcional*). En particular $f(a)$ denota a Berlín y $f(b)$ a Brasilia.

Ejemplo 4.2. Si el universo consta de números naturales, entonces:

- La constante a denota al individuo 0 y la constante b al individuo 1.
- Los términos funcionales $f^{(2)}(x, y)$ y $g^{(2)}(x, y)$ denotan a los individuos $x + y$ y $x * y$ respectivamente.

- En tal caso, los individuos 2 y 4 se representan mediante $f(b, b)$ y $g(f(b, b), f(b, b))$ respectivamente.

4.2.2. Fórmulas

Una vez definidos los términos podemos construir las fórmulas del lenguaje, las cuales representan las relaciones entre individuos, así como a los enunciados generales del lenguaje. Empecemos con las fórmulas más simples, las atómicas.

Definición 4.2 (fórmula atómica) Una *fórmula atómica* es una expresión de la forma $P(t_1, \dots, t_n)$, donde P es un símbolo de predicado de índice n y t_1, \dots, t_n son términos.

Ejemplo 4.3. Definimos los símbolos de predicado $P^{(2)}, R^{(3)}, Q^{(1)}$, los símbolos de función $f^{(1)}$ y $g^{(2)}$, y las constantes a, b y c . Las siguientes son fórmulas atómicas:

- $P(b, f(y))$
- $Q(g(f(a), c))$
- $R(z, f(g(a, c)), b)$

Ahora que tenemos fórmulas atómicas podemos combinarlas con los conectivos proposicionales para obtener fórmulas más complejas.

Ejemplo 4.4. En el universo de discurso de los números naturales, si $a + b = c + b$ entonces $a = c$.

Definimos las constantes a, b, c y los siguientes símbolos de función:

$f(x, y)$ para representar $x + y$, $igual(x, y)$ para representar $x = y$

La especificación queda entonces como sigue:

$$igual(f(a, b), f(c, b)) \rightarrow igual(a, c)$$

Ejemplo 4.5. En la expresión *Bombón es un gato que araña* tenemos lo siguiente:

- El universo de discurso son los animales (los mamíferos, los felinos, cualquier conjunto, raza o familia que incluya a los gatos).
- Los predicados que definimos son:

$G(x)$ x es un gato

$A(x)$ x araña

Siendo *Bombón* uno de los individuos concretos del universo de discurso, estará repre-

sentado por una constante, su propio nombre. La expresión lógica queda como sigue:

$$G(\text{Bombón}) \wedge A(\text{Bombón})$$

Otros ejemplos son:

$$\text{Perro}(x) \rightarrow \text{Tienecola}(x)$$

$$\text{Madre}(x, y) \wedge \text{Madre}(x, z) \rightarrow \text{Hermanos}(y, z)$$

$$\text{Calif}(x) \rightarrow x \geq 0 \wedge x \leq 10$$

Obsérvese que en el último ejemplo los predicados \geq , \leq se usan de manera infija como es usual en matemáticas. ¿Cual es el universo de discurso en cada caso?

De los ejemplos anteriores se observa que las fórmulas con predicados se generan de la misma manera que las fórmulas de la lógica proposicional, sólo que las fórmulas atómicas han cambiado de simples variables proposicionales a predicados que involucran términos. Veamos la gramática formal, en la que usamos el metasímbolo “|” para separar alternativas de sustitución para un mismo símbolo no terminal de manera abreviada, y el metasímbolo “...” para denotar “más como los anteriores”.

$$E ::= \text{pred}(\text{lista-de-term}) \quad (4.18)$$

$$E ::= \neg E \quad (4.19)$$

$$E ::= E \rightarrow E \quad (4.20)$$

$$E ::= E \vee E \quad (4.21)$$

$$E ::= E \wedge E \quad (4.22)$$

$$E ::= E \leftrightarrow E \quad (4.23)$$

$$E ::= (E) \quad (4.24)$$

$$\text{pred} ::= P | Q | R | \dots \quad (4.25)$$

$$\text{lista-de-term} ::= \text{term} \quad (4.26)$$

$$\text{lista-de-term} ::= \text{term}, \text{lista-de-term} \quad (4.27)$$

Nuevamente, cada símbolo de la categoría *pred* tiene asociado un número fijo de argumentos.

4.2.3. Fórmulas cuantificadas

Finalmente definiremos las fórmulas que involucran cuantificadores, las cuales proporcionan una gran expresividad al lenguaje.

Definición 4.3 (cuantificaciones) Sea E una fórmula. La expresión $\forall x E$ es la *cuantificación universal* de E con respecto a x y representa al enunciado *para todo x se cumple E* . Análogamente, la expresión $\exists x E$ es la *cuantificación existencial* de E con respecto a x y representa al enunciado *existe un x que cumple E* . En ambos casos la fórmula E se conoce

como el *alcance de la cuantificación* y la variable que figura inmediatamente después del cuantificador se conoce como *variable del cuantificador*.

Veamos algunos ejemplos de especificación.

Ejemplo 4.6. Supongamos que el universo de discurso es el universo astronómico. Sea $S(x)$ el predicado *ser sol* y $P(x)$ el predicado *ser planeta*. Vamos a traducir algunos enunciados sencillos que involucran cuantificadores.

- Todo es un sol: $\forall x S(x)$
- Todo es un planeta: $\forall y P(y)$
- Existe un planeta y un sol: $\exists x \exists y (P(x) \wedge S(y))$
- Cualquiera es sol o planeta: $\forall z (P(z) \vee S(z))$

En la sección 4.3 discutiremos el proceso de especificación más ampliamente. A continuación ejemplificamos el concepto de *alcance*.

Ejemplo 4.7. Encerraremos en un cuadro los alcances, marcando la variable de las cuantificaciones correspondientes.

$$\forall x \left(\left(x > i \wedge i > j \right) \rightarrow \exists i \exists j \left(\boxed{x > i \wedge x > j} \right) \right)$$

El recuadro de guiones marca el alcance de la cuantificación $\forall x$ mientras que el recuadro de línea sólida marca el alcance de las cuantificaciones $\exists i$ y $\exists j$.

Agregamos a la gramática que acabamos de dar para la lógica de predicados, las reglas que describen la cuantificación universal y existencial como fórmulas lógicas:

$$E ::= \forall \text{ var } E \quad (4.28)$$

$$E ::= \exists \text{ var } E \quad (4.29)$$

Con esto nuestra gramática para fórmulas de la lógica de predicados queda completa.

4.2.4. Variables libres y ligadas

Consideremos el enunciado *todos son blancos*; si deseamos especificarlo formalmente, primero debemos definir un predicado $B(x)$ cuyo significado es *x es blanco*, para después cuantificar universalmente, obteniendo la fórmula $\forall x B(x)$. Ahora bien, consideremos la fórmula $\forall y B(y)$, ¿qué enunciado en español se especifica ahora? Fácilmente nos damos cuenta que su significado es nuevamente *todos son blancos*; es decir, el nombre particular de la variable utilizada, en este caso x o y , es irrelevante para entender el significado de la

fórmula. Por esta razón a la variable de un cuantificador se le conoce también como variable *artificial o monigote*¹, porque únicamente marca el lugar o la posición. En contraste, consideremos las fórmulas $B(x)$ y $B(y)$, y supongamos que el universo son los números naturales, siendo B el predicado *ser par*. Con esta información no es posible entender el significado² de estas fórmulas, pues hace falta saber el valor de sus variables. Por ejemplo, si x es 3 entonces $B(x)$ significa *3 es par*, mientras que si y vale 8 entonces $B(y)$ significa *8 es par*, de donde claramente $B(x)$ y $B(y)$ no significan lo mismo, pues su significado depende del valor particular de sus variables. La pregunta inmediata es: ¿cuándo es relevante el valor particular de una variable para conocer el significado y valor de verdad de una fórmula? Para responderla introducimos los conceptos de *variable libre y ligada*.

Definición 4.4 (variable libre) Se dice que una **presencia específica** de una variable x en una fórmula A es *libre* si no es la variable artificial de un cuantificador ni figura dentro del alcance de una cuantificación cuya variable artificial también es x .

En la siguiente tabla presentamos algunas fórmulas y la lista de variables libres de cada una de ellas.

Cuantificación	Variables libres
$\forall x((x > i \wedge i > j) \rightarrow (x > j))$	i, j
$\exists x(x > i \wedge i > j)$	i, j
$\forall i \forall j((x > i \wedge i > j) \rightarrow (x > j))$	x
$\forall i((x > i \wedge i > j) \rightarrow (x > j))$	x, j
$\exists i \exists j(x > i \wedge i > j)$	x
$\exists j(x > i \wedge i > j)$	i, x

Las variables que no figuran libres en una fórmula se denominan *ligadas o acotadas*. Veamos una definición más detallada.

Definición 4.5 (variable ligada o acotada) Decimos que una **presencia específica** de una variable x en una fórmula A es *ligada o acotada* si x es una variable artificial de A o cae dentro del alcance de un cuantificador con la misma variable artificial x .

Los enunciados en español se formalizan mediante fórmulas que no tienen variables libres, a las cuales llamamos también enunciados, sentencias o fórmulas cerradas.

Definición 4.6 (enunciado) Un *enunciado* es una fórmula A que no contiene presencias libres de variables.

¹En inglés *dummy*

²El valor de verdadero o falso

Ejemplo 4.8. En la expresión

$$\overset{(1)}{i} > 0 \vee \forall \overset{(2)}{i} (0 \leq \overset{(3)}{i} \rightarrow x \cdot \overset{(4)}{i} = 0)$$

tenemos cuatro presencias de i . La primera presencia de i , anotada con (1), es una presencia libre, pues no se encuentra dentro de ninguna cuantificación. El valor que contribuya a la expresión dependerá del estado en el que se la evalúe. La segunda presencia es la variable artificial de un cuantificador, por lo que es ligada. Las otras dos presencias de i también son ligadas ya que están en el alcance del cuantificador, por lo que el valor de la cuantificación no depende del estado que tenga i . Finalmente, la presencia de x es una *presencia libre*, ya que no es la variable artificial de la cuantificación en la que está; su contribución a la expresión dependerá también del estado en el que se evalúe la expresión.

Ejemplo 4.9. En la expresión

$$(\overset{(1)}{k} + \overset{(2)}{j}) > 0 \wedge \exists \overset{(3)}{j} (0 \leq \overset{(4)}{j} \leq 5 \wedge \overset{(5)}{k} < \overset{(6)}{j})$$

las presencias (1) y (5) de k son presencias libres, pues en el primer caso la k se encuentra fuera de la cuantificación y, aunque en el segundo caso se encuentra dentro de una cuantificación, la variable artificial es j , no k .

La presencia (2) de j es distinta a las presencias (3), (4) y (6), pues mientras la primera se encuentra fuera de una cuantificación y es, por lo tanto, presencia libre, las otras tres se encuentran dentro de una cuantificación donde la variable artificial es ella misma, por lo que son presencias acotadas.

El valor de esta fórmula dependerá del estado en el que se evalúen los valores libres de j y k .

Puede suceder que el usar una misma variable (j en el ejemplo anterior) para dos papeles distintos —el papel de presencia libre en (2) y de presencia ligada en (4) y (6)— lleve al lector a confusión. En estos casos es recomendable cambiar el nombre a todas las presencias *ligadas* de la variable en cuestión que participan directamente en la cuantificación para eliminar confusiones. De hacer esto, la expresión que dimos arriba quedaría como sigue:

$$(\overset{(1)}{k} + \overset{(2)}{j}) > 0 \wedge \exists \overset{(3)}{i} (0 \leq \overset{(4)}{i} \leq 5 \wedge \overset{(5)}{k} < \overset{(6)}{i}).$$

El concepto de variables libres o ligadas es similar al que presentan los lenguajes de programación con estructura de bloques. En ellos tenemos la posibilidad de *anidar* pedazos de código como se muestran a continuación.

```

1  var i: integer;
2  procedure p(var x: integer);
3      var i: integer;
4      begin
5          i := x * x;
6          x := 2 * i;
7      end

```


La presencia de i en la línea 1 es libre, ya que no se encuentra dentro de un bloque. La presencia de x en 2 es ligada y hace el papel de una declaración, pues le da nombre a una variable artificial. Las presencias de x en 5 y 6, así como las presencias de i dentro del procedimiento son acotadas (líneas 5 y 6), ya que estos identificadores sólo tienen significado dentro del procedimiento. Si cambiáramos los identificadores de x a y en todas las presencias acotadas, y de i a k también en las presencias acotadas, el procedimiento obtenido sería Como sigue y hace exactamente lo mismo que el original.

```

1    var i: integer;
2    procedure p(var y: integer);
3        var i: integer;
4    begin
5        k:= y * y;
6        y:= 2 * k;
7    end

```

Para terminar esta sección deseamos hacer hincapié en lo siguiente:

- Al trabajar con predicados es muy importante que el universo de discurso esté bien definido y sea claro.
- Los términos y las fórmulas son categorías ajenas; es decir, ningún término es fórmula y ninguna fórmula es término.
- Los términos denotan exclusivamente individuos u objetos.
- Las fórmulas atómicas (predicados) denotan únicamente proposiciones o propiedades acerca de los términos.
- Únicamente los individuos u objetos son cuantificables. Esta característica justifica la denominación *primer orden* que se le da a la lógica de predicados que estamos estudiando.

Ejercicios

4.2.1.- Sean $f^{(2)}$ y $g^{(3)}$ símbolos de función y d una constante. ¿Cuáles de las siguientes expresiones son términos? Justifique su respuesta.

- | | |
|-----------------------|--|
| a) $g(d, d)$ | b) $f(d, x)$ |
| c) $f(x, g(y, z), d)$ | d) $g(d, g(x, y, f(z, d)), f(f(d, x), w))$ |
| e) $g(x, f(y, z), d)$ | f) $g(g(y, y, f(d, d)), f(w, g(d, x, y)))$ |

4.2.2.- Sea a una constante, $f^{(1)}$ un símbolo de función, y $P^{(2)}$, $Q^{(2)}$ y $R^{(1)}$ símbolos de predicado. ¿Cuáles de las siguientes expresiones son fórmulas? Justifique su respuesta.

- | | |
|--------------------|--|
| a) $P(a, x)$ | g) $R(f(w))$ |
| b) $Q(a, f(a))$ | h) $Q(\neg R(x), w)$ |
| c) $f(a)$ | i) $P(x, y) \rightarrow \exists z Q(z, y)$ |
| d) $Q(Q(a, x), x)$ | j) $\forall f Q(f(y), y)$ |
| e) $R(Q(x, x))$ | k) $\neg R(f(z)) \vee \neg Q(a, f(w))$ |
| f) $P(f(y), a)$ | l) $\forall x \exists y Q(x, y) \wedge R(w)$ |

4.2.3.- Sean c y d constantes, $f^{(1)}$, $g^{(2)}$ y $h^{(3)}$ símbolos de función, y $P^{(3)}$ y $R^{(2)}$ símbolos de predicado. ¿Cuáles de las siguientes expresiones son fórmulas? Justifique su respuesta.

- | | |
|--|--|
| a) $Q(c, d, c)$ | d) $\exists w P(g(h(x, f(d), x), g(w, w)), h(x, x, x), c)$ |
| b) $\forall x P(f(d), h(g(c, x), d, y))$ | e) $\exists u (Q(z, z, z) \rightarrow R(y, y))$ |
| c) $\forall x P(f(d), h(R(y, y), d))$ | f) $\forall x \forall y (g(x, y) \rightarrow P(x, y, x))$ |

4.2.4.- Para cada una de las fórmulas de este inciso, clasifique las presencias de variables en libres o ligadas. Además dé el alcance de cada cuantificador.

- | | |
|---|---|
| a) $R(x, y) \wedge L(y)$ | g) $\forall x (L(x) \rightarrow R(f(x, a), x) \wedge C(f(x, a), a))$ |
| b) $\forall x R(x, f(y, z)) \wedge L(y)$ | h) $R(f(x, y), z) \wedge \exists y R(f(y, x), z) \rightarrow \forall w I(x, y)$ |
| c) $\exists x \exists y R(x, y) \wedge C(x, y)$ | i) $\forall x (C(x, z) \wedge R(f(y, x), f(x, y))) \rightarrow C(y, z)$ |
| d) $\exists y C(x, y) \vee \exists z R(x, z)$ | j) $\exists y (I(f(x, y), f(y, x)) \wedge D(r(x)))$ |
| e) $\exists y C(a, y) \vee L(a)$ | k) $\exists y (C(x, f(y, z)) \wedge D(y) \wedge \forall x I(z, r(y)))$ |
| f) $D(f(x, y)) \vee \forall z C(z, r(y))$ | l) $\forall x \exists z I(z, r(x)) \rightarrow C(z, y) \wedge D(y)$ |
| m) $C(f(x, y), z) \wedge \exists y C(f(y, x), z) \rightarrow \forall x (L(x) \wedge L(y) \wedge I(x, y))$ | |

4.2.5.- Para cada una de las siguientes fórmulas, clasifique las presencias de variables en libres o ligadas. Además dé el alcance de cada cuantificador.

- | |
|--|
| a) $\forall x \exists z (Q(z, y) \wedge \exists y R(x, f(x)))$ |
| b) $P(x, a, y) \vee \exists y (P(x, y, a) \wedge R(a, z))$ |
| c) $W(f(x, a), g(y)) \wedge \forall x \exists y S(f(x, a), g(z))$ |
| d) $\neg R(f(x, x), w, g(x)) \wedge \forall x \exists y T(x, y, g(z))$ |
| e) $\forall w T(w, x, g(y)) \rightarrow \neg \exists z R(x, f(w, y))$ |

4.2.6.- Construya para cada inciso una fórmula A que cumpla las condiciones dadas.

- | |
|--|
| a) A es un enunciado que es una cuantificación de una implicación, donde el consecuente es una fórmula existencial. |
| b) A es un enunciado y es una cuantificación existencial de una conjunción. |
| c) A es una fórmula que incluye cuantificadores pero tiene al menos tres presencias libres de exactamente dos variables. |

- d) A es un enunciado y una disyunción de un enunciado atómico con una fórmula existencial cuyo alcance es un predicado ternario.
- e) A es una fórmula que no es un enunciado y tiene dos cuantificadores universales con variables distintas, un cuantificador existencial, y además se convierte en un enunciado al cuantificarla universalmente.

4.2.7.- Considere la siguiente fórmula A :

$$A =_{def} \forall x \exists y \left(P(x, g(w, y)) \wedge R(a, v, f(w)) \right) \rightarrow \neg Q(a, z) \vee \forall z (T(z, y, x, a))$$

- a) Dé el alcance de cada cuantificación.
- b) Clasifique cada presencia de una variable de A en libre o ligada. Indique esto debajo o arriba de cada presencia de variable en la misma fórmula.
- c) Dé los conjuntos de variables libres y ligadas de A y en caso necesario construya una variante de la fórmula tal que dichos conjuntos sean ajenos.

4.3. Especificación formal

El proceso de especificación o traducción del español a la lógica formal no siempre es sencillo. Algunas frases del español no se pueden traducir de una manera completamente fiel a la lógica de predicados, como veremos en algunos ejemplos. Sin embargo, el proceso es de gran importancia pues es la base de muchos métodos de especificación formal utilizados en inteligencia artificial o ingeniería de software (como ejemplo tenemos el lenguaje de especificación Z). A continuación presentamos algunos consejos, observaciones y ejemplos que pretenden facilitar la especificación del español en términos de la lógica.

- Únicamente podemos especificar afirmaciones o proposiciones; no es posible traducir preguntas, exclamaciones, órdenes, invitaciones, etcétera.
- La idea básica es extraer predicados a partir de los enunciados dados en español, de manera que el enunciado completo se construya al combinar fórmulas atómicas mediante conectivos y cuantificadores. Por ejemplo, la frase *me gustan los tacos y las pizzas* debe traducirse como *me gustan los tacos y me gustan las pizzas*. Análogamente *iré de vacaciones a la playa o a la montaña* significa *iré de vacaciones a la playa o iré de vacaciones a la montaña*.
- La conjunción “y” se traduce como \wedge . La palabra “pero” también, aunque el sentido original del español se pierde. Por ejemplo *te doy dulces pero haces la tarea* sólo puede traducirse en *Te doy dulces y haces la tarea*, lo cual es diferente en el lenguaje español, donde esta clase de ejemplos señalan una especie de condición que no puede capturarse de manera general en la lógica.
- La disyunción es inclusiva: *comeremos pollo o vegetales* incluye el caso en que se coman ambos.

- Con la implicación hay que ser cautelosos, sobre todo en el caso de frases de la forma *A sólo si B* lo cual es equivalente con *Si no B entonces no A*, que a su vez es equivalente con *Si A entonces B*. Es un error común intentar traducir dicha frase inicial mediante $B \rightarrow A$.
- Si en español aparecen frases como *para todos*, *para cualquier*, *todos*, *cualquiera*, *los*, *las*, debe usarse el cuantificador universal \forall .
- Si en español hay frases como *para algún*, *existe un*, *alguno*, *alguna*, *algunos*, *algunas*, *uno*, *una*, *alguien*, generalmente se usa el cuantificador existencial \exists .

Importante: En ciertas ocasiones, frases en español que involucran las palabras *alguien* o *algo*, deben especificarse con un cuantificador universal y no uno existencial. Por ejemplo, el enunciado *si hay alguien demasiado alto entonces se pegará con el marco de la puerta* se puede reescribir en español como *cualquiera demasiado alto chocará con el marco de la puerta*, lo cual nos lleva a $\forall x(A(x) \rightarrow C(x))$. El lector debe convencerse de que no es posible traducir esta oración usando un cuantificador existencial.

- Pronombres como *él*, *ella*, *eso* no se refieren a un individuo particular sino que se usan como referencia a algo o alguien mencionado previamente, por lo que obtienen significado del contexto particular. Cuando un pronombre aparezca en un enunciado debe uno averiguar primero a quién o a qué se refiere. Por ejemplo, en el enunciado *Martha es amiga de Lupita pero ella no es amiga de Karla* debe traducirse como *Martha es amiga de Lupita y Lupita no es amiga de Karla*. Similarmente, cuando necesitamos de variables y cuantificadores, como en *hay un perro con manchas y él ladra en las mañanas* es un error tratar de traducir por separado *hay un perro con manchas* y *él ladra en las mañanas* puesto que lo que existe está ligado con lo que ladra por la conjunción, de manera que debe utilizarse una variable que modele esta conexión.
- Las variables no se mencionan en español sino que son sólo un formalismo para representar individuos. Por ejemplo, la fórmula $\forall x(M(x) \rightarrow T(x))$ puede traducirse como *Cualquier minotauro es troyano* y no como *para cualquier x, si x es minotauro entonces x es troyano*. Enunciados de esta forma sólo sirven como pasos intermedios en el proceso de traducción pues no forman parte del español correcto ni de la lógica formal. A este mismo respecto es prácticamente imposible que en una traducción del español figuren variables libres.
- Los esquemas $\forall x(A \rightarrow B)$ y $\exists x(A \wedge B)$ son de gran utilidad y bastante comunes. Menos comunes, aunque también adecuados, son los esquemas $\forall x(A \wedge B)$, $\forall x(A \vee B)$ y $\exists x(A \vee B)$.
- El esquema $\exists x(A \rightarrow B)$, si bien es una fórmula sintácticamente correcta, es extremadamente raro que provenga de una especificación en español.
- El hecho de que se usen dos o más variables distintas no implica que éstas representen a elementos distintos del universo, de manera que para especificar dos individuos

distintos no es suficiente contar simplemente con variables distintas. Las fórmulas $\exists x P(x)$ y $\exists x \exists y (P(x) \wedge P(y))$ resultan lógicamente equivalentes por lo que expresan lo mismo; a saber, que algo cumple P . Si estamos hablando de elementos distintos del universo, se debe agregar explícitamente que x e y tienen la propiedad de ser distintas, es decir $x \neq y$.

4.3.1. Juicios aristotélicos

Una gran parte de las especificaciones en lenguaje natural pueden formalizarse mediante instancias de alguno de los cuatro juicios aristotélicos básicos, los cuales se refieren a dos relaciones y expresan las posibilidades de que éstas se cumplan o no en ciertos individuos.

Ejemplo 4.10. Tomemos como universo de discurso al reino animal. Vamos a construir los llamados *juicios aristotélicos fundamentales* a partir de las propiedades *ser perico* y *ser feo*. Primero definimos los predicados necesarios:

Predicados:

$$P(x) \quad x \text{ es perico} \qquad F(x) \quad x \text{ es feo}$$

(a) Juicio universal afirmativo: *Todos los pericos son feos*,

$$\forall x (P(x) \rightarrow F(x)).$$

(b) Juicio existencial afirmativo: *Algunos pericos son feos*,

$$\exists x (P(x) \wedge F(x)).$$

(c) Juicio existencial negativo: *Algunos pericos no son feos*,

$$\exists x (P(x) \wedge \neg F(x)).$$

(d) Juicio universal negativo: *Ningún perico es feo*, lo cual es equivalente a decir que cualquier perico no es feo o bien todos los pericos no son feos; de manera que las dos siguientes especificaciones son correctas:

$$\neg \exists x (P(x) \wedge F(x)), \forall x (P(x) \rightarrow \neg F(x)).$$

Una pregunta común es por qué en la especificación formal son muy comunes los juicios aristotélicos afirmativos. En las especificaciones formales que requieren una cuantificación universal se prefiere el formato $P(x, \dots) \rightarrow Q(x, \dots)$, mientras que en aquellas que requieren una cuantificación existencial se prefiere la forma $P(x) \wedge Q(x)$. Cuando tenemos una cuantificación universal examinamos todo el universo de discurso para comprobar que todo aquel que cumple $P(x, \dots)$ también cumple $Q(x, \dots)$. Si el individuo que estamos examinando no cumple $P(x, \dots)$ no nos interesa qué pasa con $Q(x, \dots)$, por lo que queremos que la cuantificación sea verdadera fijándonos únicamente en aquellos individuos que cumplen $P(x, \dots)$. Si usáramos el esquema $P(x, \dots) \wedge Q(x, \dots)$, la cuantificación sería falsa si en el universo de discurso hubiese individuos que no cumplen con $P(x, \dots)$.

En el caso de la cuantificación existencial deseamos encontrar, en el universo de discurso al menos a un individuo que cumpla con $P(x, \dots)$ y $Q(x, \dots)$. Si usamos la fórmula $P(x, \dots) \rightarrow Q(x, \dots)$, al examinar al primer individuo que no cumpla con $P(x, \dots)$ daríamos por buena la cuantificación, pues en el caso de que el antecedente sea falso, la condicional es verdadera; no seguiríamos revisando el universo de discurso para ver si encontramos a un individuo que cumpla con $P(x, \dots)$; la cuantificación se evaluaría a verdadero aun cuando no existiese ningún individuo como el que queremos. Con la conjunción garantizamos que tiene que existir al menos un individuo que cumpla con ambos predicados.

En el siguiente ejemplo nos servimos de juicios aristotélicos para obtener especificaciones más complejas.

Ejemplo 4.11. Tenemos los siguientes predicados en el universo de discurso de los habitantes de la Ciudad de México:

$I(x)$ x es inteligente.

$E(x)$ x es estudiante de la Facultad de Ciencias.

$M(x)$ a x le gusta la música.

Especificar con cuantificaciones los siguientes enunciados:

- Todos los estudiantes de la Facultad de Ciencias son inteligentes.

$$\forall x(E(x) \rightarrow I(x))$$

- A algunos estudiantes inteligentes les gusta la música.

$$\exists x(E(x) \wedge I(x) \wedge M(x))$$

- Todo aquel a quien le gusta la música es un estudiante que no es inteligente.

$$\forall x(M(x) \rightarrow E(x) \wedge \neg I(x))$$

Ejemplo 4.12. En este ejemplo observamos el significado de las distintas combinaciones de dos cuantificaciones. Sea $Q(x, y)$ el predicado x quiere a y .

- | | |
|---|------------------------------------|
| • Todos quieren a alguien: | $\forall x \exists y Q(x, y)$ |
| • Alguien quiere a todos: | $\exists x \forall y Q(x, y)$ |
| • Todos se quieren, o bien, todos quieren a todos: | $\forall x \forall y Q(x, y)$ |
| • Algunos se quieren entre sí, o bien alguien quiere a alguien: | $\exists x \exists y Q(x, y)$ |
| • Alguno no es querido por nadie: | $\exists x \forall y \neg Q(y, x)$ |
| • Alguien no quiere a nadie: | $\exists x \forall y \neg Q(x, y)$ |
| • Todos no quieren a alguien: | $\forall x \exists y \neg Q(x, y)$ |
| • Nadie quiere a todos: | $\neg \exists x \forall y Q(x, y)$ |
| • Nadie quiere a nadie: | $\forall x \forall y \neg Q(x, y)$ |
-

4.3.2. Negaciones

Con frecuencia necesitaremos traducir la negación de una cuantificación, lo cual ejemplificamos a continuación.

Ejemplo 4.13. La negación de una cuantificación puede obtenerse simplemente anteponiendo el operador de negación, por ejemplo:

- *No todos son leones* se traduce como $\neg\forall x L(x)$.
- *No existen leones* se traduce como $\neg\exists x L(x)$.

Sin embargo, estas traducciones no proporcionan información suficiente y pueden mejorarse usando equivalencias intuitivas del español:

- *No todos son leones* es lo mismo que *existe algo que no es león*, cuya traducción es $\exists x \neg L(x)$.
- *No existen leones* es lo mismo que *cualquiera no es león*, cuya traducción es $\forall x \neg L(x)$.

Por supuesto que en los dos casos ambas traducciones deben ser equivalentes en la lógica pues lo son en español. Analizaremos esto con más detalle en la subsección 4.4.4.

Veamos un ejemplo más elaborado.

Ejemplo 4.14. Traducir el enunciado *no todos los planetas tienen una luna*. Definimos los predicados $P(x)$, $L(x)$, $T(x, y)$ para *ser planeta*, *ser luna* y x *tiene a* y respectivamente.

- Lo más simple es especificar primero la cuantificación universal y anteponer la negación, obteniendo $\neg\forall x (P(x) \rightarrow \exists y (L(y) \wedge T(x, y)))$.
- Otra opción es transformar la frase a una equivalente en español que permita una estructura lógica que nos dé más información. En este caso el enunciado original es equivalente a *existe un planeta que no tiene lunas*, cuya especificación es

$$\exists x (P(x) \wedge \neg\exists y (L(y) \wedge T(x, y))).$$

- Es posible refinar aún más la traducción si observamos que la frase *no existe una luna tal que x la tenga* se puede reescribir como *para toda luna, x no la tiene*, obteniendo así la especificación más refinada posible,

$$\exists x (P(x) \wedge \forall y (L(y) \rightarrow \neg T(x, y))).$$

4.3.3. Contando objetos

Como ya mencionamos al principio de esta sección, el hecho de usar variables diferentes no implica que se refieran necesariamente a individuos distintos, de manera que para representar cantidades particulares se requiere especificar explícitamente que ciertos individuos no son iguales. Veamos algunos ejemplos.

Ejemplo 4.15. En las siguientes especificaciones se utiliza el predicado binario de igualdad = de manera infija, en lugar de *igual*(x, y). Además, las fórmulas del esquema $\neg \text{igual}(t, s)$ se escriben como $t \neq s$, como es usual en matemáticas.

- *Hay al menos una luna*, esto resulta equivalente a *hay una luna*:

$$\exists x L(x).$$

- *Hay más de una luna*; es decir, *existen al menos dos lunas*:

$$\exists x \exists y (L(x) \wedge L(y) \wedge x \neq y).$$

Obsérvese que se hace explícito el hecho de que las lunas denotadas por x e y son diferentes.

- *Hay al menos tres lunas*. De manera similar al enunciado anterior, usamos tres variables y hacemos explícito el hecho de que denotan a tres individuos diferentes:

$$\exists x \exists y \exists z (L(x) \wedge L(y) \wedge L(z) \wedge x \neq y \wedge x \neq z \wedge y \neq z).$$

En general es posible definir el enunciado *hay al menos n objetos* de manera análoga. Sin embargo es imposible especificar que existe una infinidad de objetos. ¿Por qué?

- *Existe un único sol*. Lo usual aquí es especificar que hay un sol y que cualesquiera dos soles en realidad son iguales:

$$\exists x (S(x) \wedge \forall y \forall z (S(y) \wedge S(z) \rightarrow y = z)).$$

Este esquema es de gran utilidad en matemáticas y suele abreviarse como $\exists! x P(x)$ para cualquier predicado P , indicando que sólo hay uno que lo cumple.

- *Hay a lo más un sol*, lo cual es equivalente a *cualquiera dos soles son el mismo*.

$$\forall x \forall y (S(x) \wedge S(y) \rightarrow x = y).$$

Obsérvese que esta especificación incluye el caso en que no haya soles. Otra posibilidad es especificar que *no es cierto que existen al menos dos soles*.

4.3.4. Micromundos

En inteligencia artificial un micromundo es un modelo artificialmente simple de una situación real; por ejemplo, si se desea programar un robot para que mueva objetos de manera inteligente, basta modelar los movimientos deseados sin tomar en cuenta sus dimensiones

reales ni la cantidad total de objetos en juego, para lo cual basta considerar una idealización del mundo real con pocos objetos. A continuación especificamos algunas descripciones para dos micromundos similares a los utilizados en inteligencia artificial.

El micromundo de cubos

En este micromundo hay cubos de color amarillo, azul o rojo. Un cubo puede estar sobre otro o en el piso. Definimos los predicados $S(x, y)$ representando que el cubo x está sobre el cubo y ; $A(x)$, $Az(x)$ y $R(x)$ que representan que un cubo puede ser de color amarillo, azul o rojo respectivamente; $L(x)$ significa que el cubo x está libre, es decir que ningún cubo está sobre el cubo x ; y la constante p representa al piso. Veamos algunas especificaciones.

- Ningún cubo amarillo está libre:

$$\forall x(A(x) \rightarrow \neg L(x))$$

- Hay un cubo azul libre y un cubo rojo libre:

$$\exists x \exists y (Az(x) \wedge L(x) \wedge R(y) \wedge (y)).$$

- Cualquier cubo amarillo tiene un cubo sobre él:

$$\forall x(A(x) \rightarrow \exists y(S(y, x) \wedge x \neq y)).$$

- No todos los cubos azules están libres:

$$\exists x(Az(x) \wedge \neg L(x)).$$

- Hay un cubo azul sobre el piso con un cubo amarillo sobre él y un cubo rojo sobre el amarillo:

$$\exists x \exists y \exists w (Az(x) \wedge A(y) \wedge R(w) \wedge S(x, p) \wedge S(y, x) \wedge S(w, y)).$$

Un mundo de triángulos, círculos y cuadrados

El micromundo consta de una cuadrícula de cualquier tamaño donde en cada cuadro puede haber figuras que son círculos, cuadrados o triángulos, las cuales pueden ser pequeñas, medianas o grandes. También se tienen las relaciones dadas por la posición: sur, norte, este, oeste; y las relaciones dadas por estar en la misma columna y en el mismo renglón. Los predicados para las figuras son: $T(x)$, $C(x)$ y $S(x)$ para triángulo, círculo y cuadrado respectivamente; para tamaño tenemos $P(x)$, $M(x)$ y $G(x)$ para pequeño, mediano y grande. Para la posición tenemos $Su(x, y)$, $N(x, y)$, $E(x, y)$ y $O(x, y)$; por ejemplo, $N(x, y)$ significa x está al norte de y . Finalmente, tenemos $Co(x, y)$ y $R(x, y)$ para indicar que x está en la misma columna o renglón que y , respectivamente. Hagamos algunas descripciones para este micromundo.

- Hay círculos medianos y cuadrados grandes:

$$\exists x(C(x) \wedge M(x)) \wedge \exists y(S(y) \wedge G(y)).$$

- No hay cuadrados pequeños:

$$\forall x (S(x) \rightarrow \neg P(x)).$$

- Hay un triángulo al sur de todos los círculos:

$$\exists x (T(x) \wedge \forall y (C(y) \rightarrow Su(x, y))).$$

- No hay dos triángulos en el mismo renglón:

$$\neg \exists y \exists x (T(x) \wedge T(y) \wedge R(x, y)).$$

- Hay un círculo tal que todos los círculos al oeste de él son grandes:

$$\exists x (C(x) \wedge \forall y (C(y) \wedge O(y, x) \rightarrow G(y))).$$

Con esto terminamos esta sección y a continuación nos ocupamos brevemente de algunos aspectos semánticos de la lógica de predicados. Regresaremos a los micromundos después de revisar algunos conceptos relacionados.

Ejercicios

4.3.1.- Para los siguientes predicados proponga un universo de discurso adecuado:

- a) $A(x)$ x tiene los pétalos amarillos
- b) $menor(x, \text{MÍNIMO})$ x es menor que el mínimo
- c) $P(x, y)$ x es padre de y
- d) $R(x)$ x ruge
- e) $mayor(x, 0)$ x es mayor que 0

4.3.2.- Considere los siguientes enunciados donde se usan predicados. Determine cuáles son los predicados necesarios y escriba cada uno de los enunciados en cálculo de predicados. El universo de discurso es el conjunto de todas las personas.

- a) Los enemigos de mis enemigos son mis amigos.
- b) Los amigos van y vienen; los enemigos se acumulan.
- c) Juan aprecia a María y María aprecia a Lupita; entonces Juan aprecia a Lupita.
- d) Juan es familiar de Rosa; Rosa es familiar de Guillermo; entonces Juan es familiar de Guillermo.

4.3.3.- Considere los siguientes enunciados donde se usan predicados. Determine cuáles son los predicados necesarios y escriba cada uno de los enunciados en cálculo de predicados. El universo de discurso es el conjunto de todos los animales.

- a) Los leones comen carne cruda.
- b) Sólo los leones rugen.

- c) El piquete de abejas duele mucho.
- d) La boa constrictora no es venenosa.
- e) La víbora de cascabel es venenosa.
- f) No hay mamíferos venenosos.

4.3.4.- Considere los siguientes enunciados donde se usan predicados. Determine cuáles son los predicados necesarios y escriba el enunciado usándolos. El universo de discurso son los animales de la selva.

- a) Los leones son feroces.
- b) Los elefantes asustan a los leones.
- c) Los ratones asustan a los elefantes.
- d) De esto, los ratones asustan a animales feroces.

4.3.5.- Considere los siguientes enunciados donde se usan predicados. Determine cuáles son los predicados necesarios y escriba el enunciado usándolos. El universo de discurso son las computadoras asignadas a Ciencias de la computación.

- a) La computadora x ha sido invadida (*hackeada*) desde la computadora y .
- b) La computadora x funciona con el sistema operativo Linux.
- c) La computadora x funciona con el sistema operativo Windows.
- d) El servidor del taller de Lenguajes de programación, que funciona con Linux, no fue *hackeado*.
- e) El servidor del taller de Ingeniería de software, que funciona con Windows, sí fue *hackeado*.
- f) Si una computadora tiene el sistema Linux no puede ser *hackeada*.

4.3.6.- Traduzca los siguientes enunciados a cuantificaciones universales y (o) existenciales, donde el universo de discurso son los días de la semana, suponiendo los predicados y constantes que siguen:

- $S(x)$ el día x está soleado
- $N(x)$ el día x está nublado
- L la constante "Lunes"
- M la constante "Martes"

- a) Todos los días están soleados.
- b) Algunos días no están nublados.
- c) Todo día que está soleado no está nublado.
- d) Algunos días están soleados y nublados.
- e) Ningún día es al mismo tiempo soleado y nublado.
- f) Siempre está soleado sólo si está nublado.
- g) Ningún día es soleado.
- h) El lunes estuvo soleado, por lo que todos los días estarán soleados.
- i) Se nubló el lunes y el martes.
- j) Si algún día está nublado, entonces todos los días estarán soleados.

4.3.7.- Escriba las fórmulas con cuantificadores de los siguientes enunciados, suponiendo que el universo de discurso son las personas, usando la siguiente asignación para los predicados:

- $J(x)$ x es juez
- $M(x)$ x es mujer
- $R(x, y)$ x respeta a y
- $A(x)$ x es abogado
- $Q(x)$ x es químico

- a) Hay algunas abogadas mujeres que son químicas.
- b) Ninguna mujer es abogado y químico.
- c) Algunos abogados sólo respetan jueces.
- d) Los jueces respetan sólo a los jueces.
- e) Todas las abogadas mujeres respetan a algún juez.
- f) Algunas mujeres no respetan a ningún abogado.

4.3.8.- Sea $T(x, y)$ el predicado x puede tomarle el pelo a y , donde el dominio consiste de todos los seres humanos. Use cuantificadores para expresar cada uno de los siguientes enunciados:

- a) Todo mundo puede tomarle el pelo a Juan.
- b) María puede tomarle el pelo a cualquiera.
- c) Cualquiera puede tomarle el pelo a alguien.
- d) Nadie puede tomarle el pelo a cualquiera.
- e) Siempre hay alguien que le puede tomar el pelo a cualquiera.
- f) Hay exactamente una persona a quien cualquiera puede tomarle el pelo.
- g) Hay alguien que puede tomarle el pelo a exactamente una persona distinta de sí mismo.

4.3.9.- Sea $S(x)$ el predicado x es un estudiante, $P(x)$ el predicado x es un maestro y $Q(x, y)$ el predicado x le hace una pregunta a y , donde el dominio consiste de toda la comunidad de la Facultad de Ciencias. Traduzca los siguientes enunciados a cuantificaciones:

- a) Luisa le preguntó al Profesor Miguel una pregunta.
- b) Cada estudiante le hizo una pregunta al Profesor García.
- c) Todo profesor ha hecho una pregunta al Profesor López o bien el Profesor Pérez les ha hecho una pregunta.
- d) Algún estudiante no le ha hecho preguntas a ningún profesor.
- e) Hay un profesor a quien ningún estudiante le ha hecho nunca ninguna pregunta.
- f) Un estudiante le ha hecho preguntas a todos los profesores.
- g) Hay un profesor que le ha hecho preguntas a cada uno de los profesores.
- h) Hay un estudiante al que ningún profesor le ha hecho preguntas.

4.3.10.- Hacer las siguientes traducciones, definiendo previamente el universo de discurso y los predicados necesarios.

- a) Los perros muerden a los carteros.
- b) Existe un perro que muerde a los carteros.
- c) Existe un cartero que es mordido por todos los perros.
- d) Hay un perro que no muerde carteros.
- e) Hay un cartero que no es mordido por perros.
- f) Hay un perro que es cartero y se muerde a sí mismo.

4.3.11.- Traduzca las siguientes oraciones a fórmulas, donde el universo de discurso son las novelas, usando los siguientes predicados:

- $S(x)$ x es una novela de espías
- $M(x)$ x es una novela de misterio
- $L(x)$ x es larga
- $M(x, y)$ x es mejor que y

- a) Todas las novelas de espías son largas.
- b) No todas las novelas de misterio son una novela de espías.
- c) Sólo las novelas de misterio son largas.
- d) Algunas novelas de espías son de misterio.
- e) Las novelas de espías son mejores que las de misterio.
- f) Sólo las novelas de espías son mejores que las de misterio.

4.3.12.- Traduzca los siguientes argumentos a la lógica de predicados. Especifique el universo de discurso y explique el significado de cada predicado usado.

- a) A algunos pacientes les caen bien todos los doctores. A ningún paciente le cae bien una enfermera. Por lo tanto ningún doctor es enfermera.
- b) Todos los empleados de la empresa INC deben de saber usar Cobol. Todos los empleados de INC que escriben aplicaciones deben de saber Excel. Roxana trabaja para la empresa INC, pero ella no sabe Excel. Ingrid sabe Excel pero no Cobol. Por lo tanto Roxana no escribe aplicaciones e Ingrid no trabaja para INC.

4.3.13.- Exprese las siguientes especificaciones de un sistema de cómputo usando lógica de predicados. Declare previamente los predicados que va a utilizar.

- a) Si hay menos de 30 megabytes libres en el disco duro, se envía un mensaje de advertencia a todos los usuarios.
- b) Ningún directorio puede abrirse ni ningún archivo puede cerrarse si se han detectado errores en el sistema.
- c) El sistema de archivos no puede respaldarse si hay algún usuario con sesión activa.
- d) Pueden recibirse archivos de vídeo cuando hay al menos 8 megabytes de memoria disponible y la velocidad de conexión es al menos de 56 kilobits por segundo.

4.3.14.- Considere la siguiente información acerca de la copa mundial de fútbol:

- a) *Si Alemania gana contra Italia, entonces no pierde todos sus partidos.*
- b) *Brasil vence a cada equipo contra el que Alemania pierde, excepto a él mismo.*

Realice la especificación formal de lo anterior utilizando exclusivamente el lenguaje $\mathcal{L} = \{G^{(2)}, br, al, it\}$ donde $G(x, y)$ denota la relación *x le gana a y* y las constantes denotan a los países respectivos.

4.3.15.- Realice la especificación formal acerca de las siguientes propiedades de la función que verifica la pertenencia de un elemento a una lista dada. Defina primero el lenguaje a utilizar.

- a) Ningún elemento pertenece a la lista vacía.
- b) Si cierto elemento es la cabeza de una lista, entonces dicho elemento pertenece a esa lista.
- c) La cabeza de una lista pertenece a dicha lista.
- d) Si un elemento pertenece a una lista entonces o es la cabeza de esa lista o pertenece a la cola.
- e) Si un elemento pertenece a una lista entonces pertenece a cualquier otra lista cuya cola es la lista anterior.
- f) Si un elemento pertenece a la concatenación de dos listas entonces pertenece a alguna de estas dos listas.
- g) Si un elemento pertenece a una lista entonces dicha lista es la concatenación de otras dos donde la segunda tiene como cabeza a dicho elemento.
- h) Que un elemento pertenezca a la reversa de una lista equivale a que pertenezca a la lista.

4.4. Semántica informal

En esta sección nos dedicaremos a dar ciertas ideas acerca de la semántica de la lógica de predicados. Lo haremos de manera informal e intuitiva, dado que la semántica formal requiere de conceptos matemáticos más avanzados que caen fuera del alcance de este libro.

4.4.1. Dominios de interpretación

Antes de determinar cuándo una fórmula de la lógica de predicados es verdadera debemos formalizar el concepto de universo de discurso; eso se hace mediante un dominio de interpretación, el cual es un conjunto no vacío en el que se definen matemáticamente los significados de los símbolos de constante, predicado y función usados en las especificaciones formales, de manera que un símbolo de constante será un individuo, y los predicados y funciones serán operadores entre elementos del universo, que devuelven otro elemento del

universo en el caso de una función, o bien un valor booleano en el caso de un predicado. Veamos algunos ejemplos.

Tabla 4.1. Distintos universos y dominios

Con:	Se representa a	Operadores	Comentarios:
\mathbb{Z}	Los enteros	$+$ $-$ \times mod div $neg?, $	Los operadores para los números naturales siguen siendo válidos y agregamos la operación de resta, el operador de decisión <i>ser negativo</i> y el operador de divisibilidad.
\mathbb{N}	Los números naturales	$0, 1$ $+$ \times mod div $>, <,$ $par?,$ $primo?$	Los números naturales tienen definidas la suma y el producto. También tienen residuo entero (<i>mod</i>) y cociente entero (<i>div</i>). Como interpretación de constantes tenemos la identidad 1 y elemento nulo 0, así como los operadores de decisión para orden, par y primo que corresponden a predicados.
\mathbb{R}	Los números reales	$\sqrt{\quad}$ $\lfloor \quad \rfloor$ $\lceil \quad \rceil$ $rac?$ π e	Agregamos las operaciones de raíz cuadrada (válida sólo para reales positivos), mayor entero menor o igual, menor entero mayor o igual, el predicado ser racional y las constantes π y e
\mathbb{Q}	Números racionales	$\div,$ num den $simp$ $etc.$	Aquí ya es posible usar la división; tenemos además operadores que devuelven el numerador o denominador de un racional y el operador de simplificación que elimina factores comunes.
\mathbb{B}	Los booleanos $\{1, 0\}$	\wedge, \vee $\rightarrow, \leftrightarrow$ \neg	\mathbb{B} es el tipo booleano, que recibe su nombre del matemático inglés George Boole (1815-1864) que creó las bases algebraicas para la lógica. Los operadores son los mismos definidos en el capítulo dos.

Continúa en la siguiente página

Tabla 4.1. Distintos universos y dominios

(continúa...)

Con:	Se representa a	Operadores	Comentarios:
\mathcal{MB}	Micromundo de cubos	piso <i>azul?</i> <i>verde?</i> <i>libre?</i> <i>sobre?</i> <i>arriba?</i> mover etc.	El universo es heterogéneo pues tiene al piso. Aquí se tienen predicados para los colores; si se desea tener una función que devuelva el color de un objeto es necesario añadir los colores al dominio y también predicados para decidir si un individuo es color o cubo.
\mathcal{H}	Los seres vivos que pertenecen al reino Fungi	<i>comestible?</i> <i>venenoso?</i> <i>=-familia?</i> <i>mismo?</i> etc.	Si se desean operaciones que devuelvan lugar de origen o dimensiones, por ejemplo, el dominio debe volverse heterogéneo para incluir lugares y números.
\mathcal{FC}	La Facultad de Ciencias	inscribir estudiar calificar <i>estudiante?</i> <i>profesor?</i> <i>reprueba?</i> etc.	Un dominio donde hay personas, salones, clases, números de cuenta, libros, apuntes, etc..
\mathcal{MF}	Micromundo de figuras	<i>cuadrado?</i> <i>círculo?</i> <i>triángulo?</i> <i>pequeño?</i> <i>al-norte?</i> etc.	En este dominio sólo hay figuras pero no es completamente homogéneo pues las figuras son de tres clases diferentes, por lo que necesitamos de los predicados para cuadrado, círculo y triángulo. Sin embargo, y a diferencia de otros mundos heterogéneos, el dominio está claramente partido en tres clases distintas de objetos.
$MiBiblio$	Una biblioteca	elegir prestar ordenar <i>está?</i> etc.	El dominio es heterogéneo y debe contener libros, libreros, personas, etc.

4.4.2. Noción informal de verdad

Si el dominio de interpretación (universo de discurso) que esté en uso es finito, entonces, en teoría, podemos asignar el valor de falso o verdadero a cada predicado analizando todas las posibles combinaciones de individuos en dicho universo de discurso. Por ejemplo, si tenemos el predicado $>$ (*mayor que*) y nuestro universo de discurso consiste de los enteros 1, 2, 3 y 4, entonces podemos hacer una tabla que asigne los valores de falso o verdadero a cada pareja de individuos, como podemos observar en la tabla 4.2.

Tabla 4.2. Asignación para el predicado $>$ (mayor que)

$>$	1	2	3	4
1	0	0	0	0
2	1	0	0	0
3	1	1	0	0
4	1	1	1	0

Por supuesto que esto resulta más complicado si el universo es demasiado grande, como en el caso en que el universo conste de los enteros 1, \dots , 1000. Por otra parte, si el universo en cuestión consta de todos los números naturales, resulta imposible construir la tabla pues tendría un número infinito de columnas y un número infinito de renglones, como se ve en la tabla 4.3.

Tabla 4.3. Asignación del predicado $>$ con un conjunto infinito

$>$	1	2	3	4	5	6	7	8	10	11	12	13	14	15	16	17	18	19	20
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
.....																						
.....																						

Si el índice de un predicado es uno o dos y el universo finito, también es fácil ver la asignación en una tabla. Sin embargo, si el índice es tres o más, aunque el universo sea relativamente pequeño, ya no es fácil visualizar dicha asignación. De manera que el uso de tablas de verdad para la lógica de predicados es inadecuado. Esto es de esperarse, ya que la noción de verdad en lógica de predicados es mucho más complicada puesto que depende de un mundo en particular, al contrario de lo que sucedía en la lógica de proposiciones, donde

en el fondo el único mundo o universo de discurso es el de los valores booleanos *cierto* o *falso*. En esta nueva lógica, el universo de discurso puede incluir literalmente cualquier cosa: números, conjuntos, piedras, flores, árboles, palabras, galaxias, etcétera. De manera que la noción de verdad dependerá del mundo que hayamos fijado de antemano. Es claro que al cambiar éste, el valor de verdad de una fórmula también puede hacerlo.

Antes de dar una definición de verdad analicemos el caso de los cuantificadores con un ejemplo sencillo en el micromundo de figuras:

- *Todos son círculos*: $\forall xC(x)$. Esto será cierto si y sólo si al revisar cada objeto del micromundo, el objeto resulta ser un círculo. Si suponemos que hay n objetos, denotados por las constantes a_1, \dots, a_n , entonces $\forall xC(x)$ será cierto si y sólo si $C(a_1)$ es cierta y $C(a_2)$ es cierta y ... y $C(a_n)$ es cierta; es decir, si y sólo si la conjunción $C(a_1) \wedge \dots \wedge C(a_n)$ es cierta. Obsérvese que esto no puede ser una definición, pues en el caso en que el universo sea infinito es imposible formar la conjunción de todos los objetos.
- *Existe algo pequeño*: $\exists xP(x)$. Similarmente a la cuantificación universal, esta fórmula es cierta si y sólo si alguno de los objetos a_1, \dots, a_n resulta ser pequeño; es decir, si la disyunción $P(a_1) \vee \dots \vee P(a_n)$ es cierta.

Esta idea intuitiva nos lleva a una definición informal de verdad para cualquier fórmula de la lógica de predicados, la cual damos a continuación.

Definición 4.4.1. Dada una fórmula A de la lógica de predicados, definimos cuándo A es verdadera en un mundo o universo de discurso dado \mathcal{M} , de acuerdo a su forma sintáctica, como sigue:

- Si A es una fórmula atómica, digamos $P(t_1, \dots, t_n)$, entonces A es verdadera en \mathcal{M} si y sólo si los valores de los términos t_1, \dots, t_n como individuos de \mathcal{M} están en la relación del universo definida por P .
- Si A es una fórmula proposicional³, entonces usamos los criterios de verdad de la lógica proposicional.
- Si $A = \forall xB$ es una fórmula universal, entonces A es verdadera en \mathcal{M} si y sólo si B es verdadera en \mathcal{M} para todos los posibles valores de x como individuo de \mathcal{M} .
- Si $A = \exists xB$ es una fórmula existencial, entonces A es verdadera en \mathcal{M} si y sólo si B es verdadera en \mathcal{M} para algún valor de x como individuo de \mathcal{M} .

Esta definición es informal puesto que en el caso en que el universo de discurso sea infinito, no queda claro en general cómo mostrar que la fórmula $\forall xB$ es cierta para todos los valores posibles de x . La definición formal se estudia en cursos de lógica matemática o computacional y se conoce como definición de verdad de Tarski.

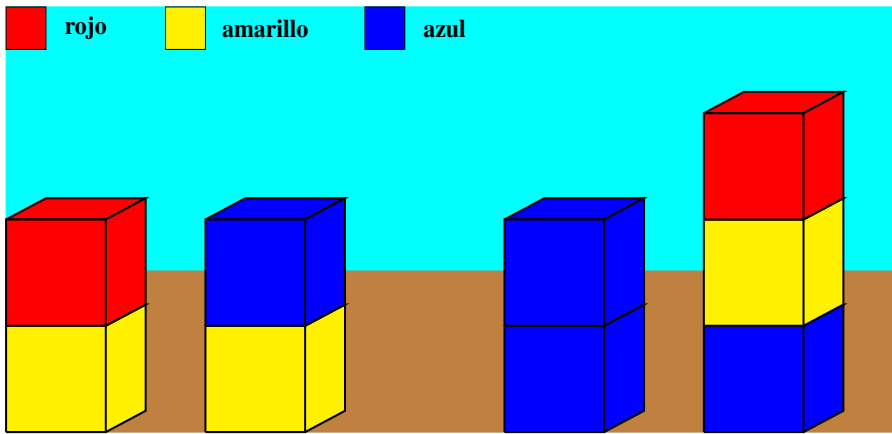
³Es decir, una fórmula que pertenece a algún esquema de la lógica proposicional, con predicados en lugar de variables proposicionales. Por ejemplo $\forall xP(x) \rightarrow \neg Q(x, y)$ que corresponde al esquema proposicional $A \rightarrow B$.

4.4.3. Verdad en micromundos

A continuación regresamos a nuestros dos micromundos particulares empezando con el mundo de los cubos para ejemplificar la definición informal de semántica que acabamos de enunciar. En los siguientes ejemplos nos referiremos al mundo particular que se encuentra en la figura 4.1.

Queremos ahora determinar la semántica de algunas fórmulas en este micromundo.

Figura 4.1. Micromundo de cubos



- Cualquier cubo rojo está libre:

$$\forall x (R(x) \rightarrow L(x)).$$

Verdadero, pues los cubos rojos en la primera y cuarta torre, que son todos los cubos rojos en este micromundo, están libres.

- Todos los cubos sobre el piso son azules:

$$\forall x (S(x, p) \rightarrow Az(x)).$$

Falso, pues la primera y segunda torre tienen cubos amarillos sobre el piso, por lo que no todos los cubos sobre el piso son azules.

- Cualquier cubo que esté sobre un cubo amarillo es rojo o azul:

$$\forall x (\exists y (A(y) \wedge S(x, y)) \rightarrow R(x) \vee Az(x)).$$

Cierto, ya que los cubos amarillos de la primera y cuarta torre tienen un cubo rojo encima, y el cubo amarillo de la segunda torre tiene encima un cubo azul.

- Hay un cubo rojo sobre un cubo rojo:

$$\exists x \exists y (R(x) \wedge R(y) \wedge S(x, y)).$$

Falso. Los dos cubos rojos, en la primera y cuarta torre, son libres, por lo que no tienen encima a ningún cubo, en particular a uno rojo.

- Hay un cubo amarillo libre sobre el piso:

$$\exists x (A(x) \wedge L(x) \wedge S(x, p)).$$

Falso. No hay ningún cubo libre sobre el piso, en particular que sea amarillo, por lo que la fórmula es falsa.

- Ningún cubo está sobre el piso:

$$\forall x \neg S(x, p).$$

Falso, pues el cubo amarillo en la primera torre sí está sobre el piso.

- Hay un cubo amarillo que está sobre uno azul y hay un cubo azul sobre él:

$$\exists x \exists y (A(x) \wedge Az(y) \wedge S(x, y) \wedge \exists w (Az(w) \wedge S(w, x))).$$

Falso. No hay una torre que contenga una secuencia de cubo azul, cubo amarillo y cubo azul.

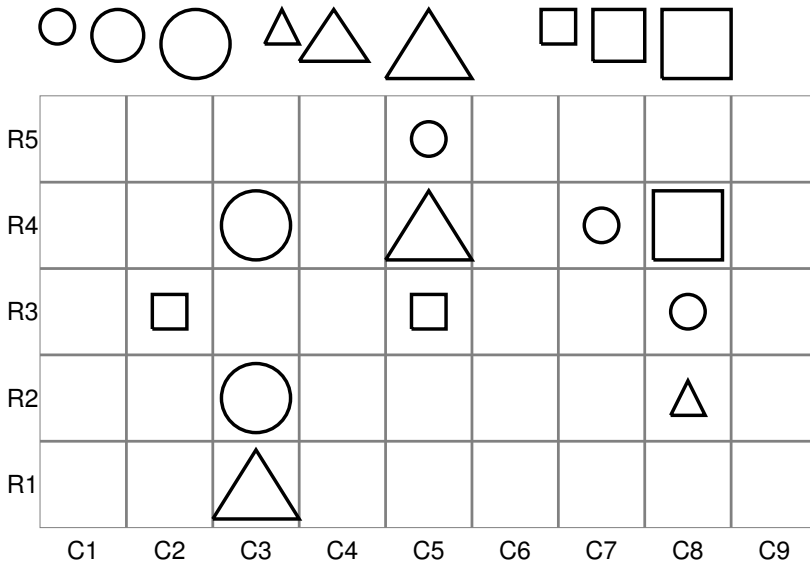
- Todos los cubos están sobre algo:

$$\forall x \exists y S(x, y).$$

Verdadera. Todos los cubos están o sobre el piso o sobre algún otro cubo.

Veamos ahora un micromundo particular de figuras geométricas en la figura 4.2, con los predicados que ya definimos para los mundos de figuras geométricas; tenemos que decidir cuáles de las fórmulas que le siguen son falsas o verdaderas. Arriba del mundo observamos los tres tamaños de figuras disponibles. Además usaremos coordenadas para renglón y columna, denotadas (RiCj), como un auxiliar para señalar cada cuadro en particular.

Figura 4.2. Micromundo de figuras geométricas



- Hay círculos medianos y cuadrados grandes:

$$\exists x (C(x) \wedge M(x)) \wedge \exists y (S(y) \wedge G(y)).$$

Falso, pues no hay ningún círculo mediano.

- No hay cuadrados pequeños:

$$\forall x(S(x) \rightarrow \neg P(x)).$$

En (R3C2) hay un cuadrado pequeño, por lo tanto la fórmula es falsa.

- Ningún cuadrado está al norte de un círculo grande:

$$\neg \exists x(S(x) \wedge \exists y(C(y) \wedge G(y) \wedge N(x, y))).$$

Falso, pues el cuadrado en (R3C5) sí está al norte del círculo grande en (R2C3).

- Todos los círculos medianos están al oeste de un mismo triángulo grande:

$$\exists x(T(x) \wedge G(x) \wedge \forall y(C(y) \wedge M(y) \rightarrow O(y, x))).$$

Como sí hay un triángulo grande en (R4C5) pero no hay círculos medianos, la implicación tiene antecedente falso y la fórmula se evalúa a verdadera.

- Todos los cuadrados pequeños están al sur de cualquier triángulo:

$$\forall x(S(x) \wedge P(x) \rightarrow \forall y(T(y) \rightarrow Su(x, y))).$$

Los cuadrados pequeños están en (R3), pero no están al sur del triángulo en (R2C8), por lo que la fórmula es falsa.

- Si dos cuadrados están en el mismo renglón, entonces cualquier triángulo al sur de ambos es mediano:

$$\forall x \forall y(S(x) \wedge S(y) \wedge R(x, y) \rightarrow \forall z(T(z) \wedge Su(z, x) \wedge Su(z, y) \rightarrow M(z))).$$

La fórmula es falsa, pues para los dos cuadrados en (R3), ninguno de los triángulos al sur es mediano.

- No hay dos triángulos medianos en la misma columna; y si un triángulo es grande, entonces hay un círculo pequeño al este de él:

$$\neg \exists x \exists y(T(x) \wedge T(y) \wedge x \neq y \wedge Co(x, y)) \\ \wedge \forall z(T(z) \wedge G(z) \rightarrow \exists w(C(w) \wedge P(w) \wedge E(w, z))).$$

El primer operando de la conjunción es verdadero porque en efecto al inspeccionar el micromundo no existen dos triángulos distintos en la misma columna; tanto para el triángulo grande en (R4C5) como el de (R1C3), hay dos círculos pequeños, (R4C7) y (R3C8), que se encuentran al este de cualquiera de ellos; por lo tanto, el segundo operando de la conjunción es verdadero y la fórmula completa también.

4.4.4. Algunas equivalencias lógicas

Con frecuencia un enunciado en español puede reescribirse de manera que la estructura lógica sea más clara, tal como lo hicimos en algunos ejemplos de la sección 4.3. En este caso ambos enunciados deben ser equivalentes, en el sentido de que cualquier conclusión obtenida con uno de ellos debe seguir siendo válida usando el otro. Esta situación se

formaliza mediante el concepto de equivalencia lógica, que ya estudiamos para la lógica proposicional, y que en la lógica de predicados tiene el mismo significado: dos fórmulas A y B son lógicamente equivalentes, denotado $A \equiv B$, si y sólo si ambas son verdaderas exactamente en los mismos mundos o interpretaciones. Debido a que no contamos con la definición formal de semántica no podemos mostrar la veracidad de estas equivalencias. Sin embargo, a continuación exponemos algunas equivalencias lógicas de utilidad.

Equivalencias proposicionales

Todas las equivalencias lógicas para la lógica proposicional siguen siendo válidas en la lógica de predicados y pueden usarse también dentro de una cuantificación.

Ejemplo 4.16. Las siguientes fórmulas son equivalentes debido al uso de una ley proposicional de equivalencia lógica.

- *Hay un círculo grande es equivalente a hay alguna figura grande que es círculo:*

$$\exists x(C(x) \wedge G(x)) \equiv \exists x(G(x) \wedge C(x)).$$

- *Cualquier figura o es triángulo o es mediana equivale a que toda figura que no es triángulo es mediana:*

$$\forall x(T(x) \vee M(x)) \equiv \forall x(\neg T(x) \rightarrow M(x)).$$

- *No es cierto que hay un cuadrado y que todas las figuras sean pequeñas significa lo mismo que o bien no hay cuadrados o bien no todas las figuras son pequeñas:*

$$\neg(\exists xS(x) \wedge \forall yP(y)) \equiv \neg\exists xS(x) \vee \neg\forall yP(y).$$

- *Si todas las figuras son cuadrados entonces no hay figuras grandes equivale a si existen figuras grandes entonces no todas son cuadrados:*

$$\forall xS(x) \rightarrow \neg\exists yG(y) \equiv \exists yG(y) \rightarrow \neg\forall xS(x).$$

Negación de cuantificadores

Volviendo a la idea de que una cuantificación puede entenderse como una conjunción o disyunción en el caso de un universo finito, podemos analizar de qué forma interactúan los cuantificadores con la negación. Por ejemplo, la fórmula $\neg\forall xC(x)$ (*no todos son círculos*) es cierta si y sólo si la negación de la conjunción de todos los objetos del universo, dada por $\neg(C(a_1) \wedge \dots \wedge C(a_1))$, es cierta; es decir, usando las leyes de De Morgan, $\neg C(a_1) \vee \dots \vee \neg C(a_1)$ es cierta, lo cual equivale a la fórmula existencial $\exists x\neg C(x)$; o, lo que es lo mismo, *existe algo que no es círculo*. Similarmente podemos analizar la negación de una fórmula existencial. Por esta razón es que las leyes de negación para cuantificaciones, que enunciamos enseguida, también se conocen como leyes de De Morgan generalizadas.

Leyes de negación:

$$\neg \forall x A \equiv \exists x \neg A \quad (4.30)$$

$$\neg \exists x A \equiv \forall x \neg A \quad (4.31)$$

Obsérvese que estas equivalencias permiten mover una negación hacia el alcance de un cuantificador, intercambiando cuantificadores.

Ejemplo 4.17. Mostramos aquí el uso de las leyes de negación para transformar una fórmula de manera que la negación se aplique únicamente a predicados.

- *No es cierto que si hay un triángulo entonces todas las figuras son medianas.*

$$\begin{aligned} \neg(\exists x T(x) \rightarrow \forall y M(y)) &\equiv \exists x T(x) \wedge \neg \forall y M(y) \\ &\equiv \exists x T(x) \wedge \exists y \neg M(y). \end{aligned}$$

Hay un triángulo y no todas las figuras son medianas, lo que equivale a hay un triángulo y hay una figura que no es mediana.

Ejemplo 4.18. En lo que sigue, el dominio de interpretación son los habitantes de la Ciudad de México, los lapsos de tiempo y los exámenes; utilizaremos los siguientes predicados:

$F(x)$: x es estudiante de la Facultad de Ciencias

$I(x)$: x es inteligente $A(x)$: x es alumno

$T(x)$: x es un tiempo $E(x, y)$: x estudia en el tiempo y

$R(x)$: x reprueba $C(x)$: el examen x fue calificado

$P(x)$: x es un examen

- *No es cierto que todos los estudiantes de la Facultad de Ciencias sean inteligentes:*

$$\begin{aligned} \neg \forall x (F(x) \rightarrow I(x)) &\equiv \exists x \neg (F(x) \rightarrow I(x)) \\ &\equiv \exists x (F(x) \wedge \neg I(x)) \end{aligned}$$

Hay un estudiante inscrito en la Facultad de Ciencias que no es inteligente.

- *No hay alumnos que estudien todo el tiempo:*

$$\begin{aligned} \neg \exists x (A(x) \wedge \forall y (T(y) \rightarrow E(x, y))) \\ &\equiv \forall x \neg (A(x) \wedge \forall y (T(y) \rightarrow E(x, y))) \\ &\equiv \forall x (\neg A(x) \vee \neg \forall y (T(y) \rightarrow E(x, y))) \\ &\equiv \forall x (\neg A(x) \vee \exists y \neg (T(y) \rightarrow E(x, y))) \\ &\equiv \forall x (\neg A(x) \vee \exists y (T(y) \wedge \neg E(x, y))) \\ &\equiv \forall x (A(x) \rightarrow \exists y (T(y) \wedge \neg E(x, y))) \end{aligned}$$

Para cualquier alumno hay un tiempo en el que no estudia.

- *No es cierto que o algún examen no se calificó o todos los alumnos reprobaron el curso:*

$$\begin{aligned}
 & \neg \left(\exists x (P(x) \wedge \neg C(x)) \vee \forall y (A(y) \rightarrow R(y)) \right) \\
 & \equiv \neg \exists x (P(x) \wedge \neg C(x)) \wedge \neg \forall y (A(y) \rightarrow R(y)) \\
 & \equiv \forall x \neg (P(x) \wedge \neg C(x)) \wedge \exists y \neg (A(y) \rightarrow R(y)) \\
 & \equiv \forall x (\neg P(x) \vee C(x)) \wedge \exists y (A(y) \wedge \neg R(y)) \\
 & \equiv \forall x (P(x) \rightarrow C(x)) \wedge \exists y (A(y) \wedge \neg R(y))
 \end{aligned}$$

Todos los exámenes se calificaron y algún alumno no reprobó.

Distributividad

Una vez que hemos visto cómo interactúan los cuantificadores con la negación, resulta natural preguntarse qué sucede con los demás conectivos proposicionales frente a los cuantificadores. Para esto presentamos algunas leyes distributivas entre cuantificadores y conectivos.

$$\forall x (A \wedge B) \equiv \forall x A \wedge \forall x B. \quad (4.32)$$

El lado izquierdo nos dice que para todo objeto x se cumple $A \wedge B$, lo cual equivale a que para todo individuo se cumplen tanto A como B . ¿Qué sucede si cambiamos la conjunción por disyunción?

Para el cuantificador existencial tenemos la siguiente equivalencia:

$$\exists x (A \vee B) \equiv \exists x A \vee \exists x B. \quad (4.33)$$

Si un individuo cumple $A \vee B$, entonces o cumple A o cumple B , de donde la disyunción de la derecha es válida. ¿Qué sucede si cambiamos la disyunción por conjunción?

Cuantificación vacua

Consideremos el siguiente enunciado: *para cualquier individuo, Berlín es la capital de Alemania*, el cual se especifica como $\forall x C(b, a)$; consideremos también el enunciado *existe un individuo tal que todos son leones*, representado con $\exists x \forall y L(y)$ o incluso con $\exists x \forall x L(x)$, donde la variable x de la cuantificación existencial es ocultada por la de la cuantificación universal, lo que hace que $L(x)$ haga referencia a la variable de la cuantificación universal. Este tipo de cuantificaciones, donde la variable cuantificada no figura libre en el alcance de la cuantificación, se conoce como cuantificación vacua o nula. Con respecto a su valor de verdad, de acuerdo a nuestra definición, $\forall x C(b, a)$ es verdadera si y sólo si $C(b, a)$ es verdadera para cualquier valor de x como un individuo particular, pero como x no figura en $C(b, a)$, basta mostrar la verdad de esta última fórmula; es decir, la cuantificación no aporta nada a la evaluación de la fórmula original y por lo tanto puede eliminarse mediante las equivalencias en la siguiente página.

Cuantificadores vacuos: si x no figura libre en A entonces

$$\forall x A \equiv A, \quad (4.34)$$

$$\exists x A \equiv A, \quad (4.35)$$

donde A puede ser, a su vez, una cuantificación con la misma variable cuantificadora o con una distinta. En particular, estas equivalencias permiten eliminar cuantificadores múltiples, puesto que $\forall x \forall x A \equiv \forall x A$ y $\exists x \exists x A \equiv \exists x A$.

Prenexación

El proceso de prenexación permite manipular un esquema proposicional binario, donde uno de los operandos es una cuantificación y la variable cuantificada en este operando no figura libre en el otro operando. El objetivo de la manipulación es “factorizar” el cuantificador, sumergiendo al operando proposicional en la cuantificación, de manera que la fórmula resultante ya no corresponde a un esquema proposicional sino a un esquema de cuantificación. Las equivalencias para el proceso de prenexación son:

Prenexación de cuantificadores: si x no figura libre en A , entonces

$$A \wedge \forall x B \equiv \forall x (A \wedge B) \quad (4.36)$$

$$A \vee \forall x B \equiv \forall x (A \vee B) \quad (4.37)$$

$$A \wedge \exists x B \equiv \exists x (A \wedge B) \quad (4.38)$$

$$A \vee \exists x B \equiv \exists x (A \vee B) \quad (4.39)$$

Prenexación de cuantificadores: si x no figura libre en A , entonces

$$A \rightarrow \forall x B \equiv \forall x (A \rightarrow B) \quad (4.40)$$

$$A \rightarrow \exists x B \equiv \exists x (A \rightarrow B) \quad (4.41)$$

Prenexación de cuantificadores: si x no figura libre en B , entonces

$$\forall x A \rightarrow B \equiv \exists x (A \rightarrow B) \quad (4.42)$$

$$\exists x A \rightarrow B \equiv \forall x (A \rightarrow B) \quad (4.43)$$

Obsérvese que en los dos últimos casos se intercambi6 el cuantificador al momento de prenexar.

Veamos un ejemplo del proceso de prenexación.

Ejemplo 4.19.

- *Para cualquier objeto, es azul o Múnchen es la capital de Baviera :*

$$\forall x (A(x) \vee C(m, b)) \equiv \forall x A(x) \vee C(m, b) \cdot$$

Todos los objetos son azules o Múnchen es la capital de Baviera.

- *Hay algo tal que los gorriones son bonitos y ese algo es la capital de Francia:*

$$\exists x (\forall y (G(y) \rightarrow B(y)) \wedge C(x, f)) \equiv \forall y (G(y) \rightarrow B(y)) \wedge \exists x C(x, f).$$

Los gorriones son bonitos y algo es la capital de Francia.

4.4.5. Algunos argumentos correctos

Ya hemos mencionado con anterioridad que nuestro propósito principal para estudiar lógica consiste en obtener métodos formales para mostrar la correctud de argumentos lógicos. Si bien podemos generalizar algunos de los métodos estudiados en la lógica de proposiciones para la lógica de predicados, estos métodos no son infalibles, debido a un importante resultado de la lógica matemática llamado *teorema de indecidibilidad*, demostrado por Alonzo Church, que nos dice que no puede existir un algoritmo para decidir si un argumento dado es correcto o no. A pesar de este resultado, el problema de analizar un argumento de la lógica de predicados para intentar decidir su correctud sigue siendo de gran importancia en la práctica, puesto que la lógica de predicados es una herramienta de gran importancia para la especificación formal en computación.

Si bien no hay un algoritmo general, la correctud de un argumento puede decidirse en muchos casos mediante métodos sintácticos o semánticos que quedan fuera del alcance de este libro. Sin embargo, dado que el proceso de argumentación es relevante en la práctica, tanto en matemáticas como en ciencias de la computación, enunciamos a continuación algunos argumentos correctos de la lógica de predicados, los cuales surgen naturalmente en matemáticas.

- **Generalización universal:** Sea A una fórmula y x una variable que no figura libre en la argumentación actual. Entonces, el siguiente argumento es correcto:

$$\frac{A}{\forall x A}$$

Este argumento permite concluir la validez de la fórmula $\forall x A$ al mostrar la validez de A , cerciorándonos de que x no figura libre en ninguna de las premisas usadas para llegar a A . Esta restricción implica que en la argumentación no se usó ninguna propiedad particular de x , por lo que ésta denota a cualquier individuo posible del universo de discurso, lo cual permite realizar la generalización. Este argumento es indispensable en pruebas por inducción, como se verá en el siguiente capítulo.

- **Instanciación universal:**

$$\frac{\forall x A}{A[x := t]}$$

La correctud de este argumento es intuitivamente clara: si la fórmula $\forall x A$ se supone verdadera, entonces uno debería poder concluir $A[x := t]$ para cualquier individuo particular del universo de discurso, denotado por el término t . Sin embargo, hay que tener cuidado, puesto que en A pueden figurar otros cuantificadores y en t otras variables, por lo que se corre el peligro de capturar alguna variable de t , que por supuesto estaba libre, mediante algún cuantificador de A , causando un problema semántico importante. Es por esta razón que la sustitución en lógica de predicados no es una

sustitución textual como la estudiada antes en este libro, sino que debe vigilar no cambiar presencias libres por ligadas y viceversa.

■ **Generalización existencial:**

$$\frac{A[x := t]}{\exists x A}$$

Nuevamente, la correctud de este argumento es intuitivamente clara: si sabemos que un individuo particular t cumple la propiedad A , entonces podemos concluir que alguien cumple A ; es decir, podemos concluir $\exists x A$.

■ **Instanciación existencial:** Sea A una fórmula y c una constante nueva en la argumentación actual. Entonces, el siguiente argumento es correcto:

$$\frac{\exists x A}{A[x := c]}$$

Este argumento permite concluir la validez de A para un individuo particular c del universo de discurso a partir de la verdad de $\exists x A$. La restricción acerca de que c sea una constante nueva se debe al hecho de que no es posible saber cuál individuo particular es el que cumple A a partir de la única información que tenemos, que es $\exists x A$. De esta forma estamos denotando al individuo que existe y cumple A mediante un nombre nuevo c para así asegurar, de manera sintáctica, que no se están suponiendo otras propiedades del mismo.

Ejercicios

4.4.1.- Si $P(x)$ denota al enunciado $x > 4$, diga cuál es el valor de

- a) $P(0)$ b) $P(4)$ c) $P(6)$.

4.4.2.- Sea $C(x, y)$ el enunciado x es la capital de y . Diga cuál es el valor de verdad de:

- a) $C(\text{Toluca}, \text{México})$ c) $C(\text{Grenoble}, \text{Francia})$
b) $C(\text{Quito}, \text{Bolivia})$ d) $C(\text{Cd. Juárez}, \text{Nuevo León})$

4.4.3.- Encuentre el valor de verdad de las siguientes fórmulas si el universo son los números reales \mathbb{R} y los predicados se interpretan como sigue:

- $Q(x)$ x es un número par
- $R(x)$ x es divisible por 6
- $L(x, y)$ x es menor que y
- $P(x)$ x es un número primo
- $G(x)$ x es menor o igual a 5

- a) $\exists x (R(x) \wedge P(x))$ d) $\forall x (R(x) \rightarrow \exists y (L(x, y) \wedge R(y)))$
b) $\exists x R(x) \wedge \exists x P(x)$ e) $\forall x \exists y (L(x, y) \wedge L(y, x))$
c) $\forall x (P(x) \rightarrow \neg Q(x))$ f) $\exists x P(x) \rightarrow \exists x (P(x) \wedge R(x))$

4.4.4.- Dé un micromundo de triángulos, cuadrados y círculos donde todas las fórmulas lógicas que siguen sean verdaderas:

- $\exists x \exists y \exists z \left(T(x) \wedge C(y) \wedge S(z) \wedge ((G(x) \wedge G(y) \wedge G(z)) \vee (M(x) \wedge M(y) \wedge M(z)) \vee (P(x) \wedge P(y) \wedge P(z))) \wedge R(x, y) \wedge R(y, z) \right)$
- $\exists x \exists y (C(x) \wedge P(x) \wedge C(y) \wedge M(y) \wedge E(x, y)).$
- $\exists x \exists y \exists z (T(x) \wedge P(x) \wedge T(y) \wedge M(y) \wedge T(z) \wedge G(z)).$

4.4.5.- Para cada una de las siguientes fórmulas, dé un micromundo de figuras donde estas fórmulas sean verdaderas o, en su defecto, justificar por qué no existe tal mundo.

- a) $\forall x \exists y (T(x) \rightarrow S(y) \wedge O(x, y)).$
- b) $\forall x \exists y (C(x) \wedge (T(y) \vee S(y)) \wedge (Co(x, y))).$
- c) $\exists x \exists y (R(x, y) \vee Co(x, y)).$
- d) $\exists x \exists y \exists z (C(x) \wedge M(x) \wedge N(z, x) \wedge O(y, x)).$
- e) $\forall x \exists y N(y, x) \wedge \exists z C(z).$

4.4.6.- Para cada fórmula proporcione dos micromundos de figuras, uno donde la fórmula sea verdadera y otro donde sea falsa.

- a) $\neg \forall x (C(x) \rightarrow G(x)) \wedge \exists z (P(z) \wedge \neg \exists y (T(y) \wedge O(y, z))).$
- b) $\forall x \forall y (T(x) \wedge C(y) \wedge N(x, y) \rightarrow \exists z (S(z) \wedge P(z) \wedge Z(z, x) \wedge Z(y, z))).$
- c) $\exists w (S(w) \vee G(w)) \wedge \forall x (T(x) \wedge M(x) \wedge \exists y Z(y, x) \rightarrow \exists z (G(z) \wedge N(z, x))).$
- d) $\forall w (G(w) \rightarrow \exists y (P(y) \wedge N(y, w))) \vee \exists x \exists z (T(z) \wedge M(x) \wedge O(z, x)).$
- e) $\exists x (T(x) \wedge \forall y (N(y, x) \rightarrow P(y) \vee S(y))) \wedge \forall w (C(w) \rightarrow \exists y (G(y) \wedge E(y, w))).$

4.4.7.- Formalice la siguiente afirmación y dibuje un micromundo de figuras donde sea falsa:

Existe un cuadrado grande tal que todos los objetos medianos al sur de él son círculos.

4.4.8.- Escriba el significado en español y dibuje un micromundo de figuras donde la siguiente afirmación sea verdadera:

$$\exists x (C(x) \wedge M(x) \wedge \forall y (S(y) \vee P(y) \rightarrow Z(x, y))) \wedge \forall z (G(z) \rightarrow T(z))$$

4.4.9.- Formalice la siguiente afirmación y dibuje dos micromundos de figuras, uno donde sea falsa y otro donde sea verdadera:

Hay un cuadrado grande tal que todos los círculos al norte de él son objetos medianos y x es un triángulo al oeste de dicho cuadrado.

4.4.10.- Considere la siguiente fórmula:

$$\begin{aligned} & \exists x \left(C(x) \wedge \forall y \forall z (T(y) \wedge S(z) \rightarrow N(x, y) \wedge Su(z, x)) \right) \\ & \vee \forall w \left(G(w) \rightarrow \exists v (P(v) \wedge C(v) \wedge E(v, w)) \right) \end{aligned}$$

- a) Escriba su significado en español.
- b) Dibuje un mundo donde sea verdadera y otro donde sea falsa.

4.4.11.- Dé un micromundo de cubos donde las siguientes fórmulas sean verdaderas al mismo tiempo.

- $\exists x \exists y \exists z (Az(x) \wedge L(x) \wedge A(y) \wedge L(y) \wedge R(z) \wedge L(z)).$
- $\exists x \exists y \exists z (Az(x) \wedge S(x, p) \wedge A(y) \wedge S(y, p) \wedge R(z) \wedge S(z, p)).$
- $\exists x \exists y (Az(x) \wedge A(y) \wedge S(x, y)) \wedge \exists x \exists y (R(x) \wedge Az(y) \wedge S(x, y)).$

4.4.12.- Considere las siguientes interpretaciones para los predicados:

- | | |
|-------------------------------------|------------------------------|
| • $F(x)$ x está fuera de servicio | • $O(x)$ x está ocupada |
| • $P(x)$ x se ha perdido | • $C(x)$ x está en la cola |
| • $I(x)$ x es impresora | • $T(x)$ x es trabajo |

Construya micromundos de impresoras y trabajos que hagan verdaderas a las fórmulas que siguen. La descripción de un micromundo puede ser mediante constantes y tablas de verdad para predicados.

- a) $\exists x (I(x) \wedge F(x) \wedge O(x)) \rightarrow \exists y (T(y) \wedge P(y)).$
- b) $\forall x (I(x) \rightarrow O(x)) \rightarrow \exists y (T(y) \wedge C(y)).$
- c) $\exists y (T(y) \wedge C(y) \wedge P(y)) \rightarrow \exists y (I(y) \wedge F(y)).$
- d) $\forall x (I(x) \rightarrow O(x)) \wedge \forall y (T(y) \rightarrow C(y)) \rightarrow \exists z (T(z) \wedge P(z)).$

4.4.13.- Dé las negaciones de las siguientes cuantificaciones de manera que el símbolo de negación sólo afecte a predicados. Por ejemplo, la negación de $\forall x (P(x) \wedge Q(x))$ es $\exists x (\neg P(x) \vee \neg Q(x))$, donde puede notar que no hay negación frente al cuantificador ni frente a una fórmula que consista de más de un predicado.

- a) $\forall x (x^2 > x).$
- b) $\exists x (x^2 = x).$
- c) $\forall x (P(x) \rightarrow Q(x)).$
- d) $\forall x (x^3 < x \rightarrow x < 0).$
- e) $\exists x (P(x) \wedge Q(x) \rightarrow R(x)).$

4.4.14.- Para los siguientes enunciados, diga cuál o cuáles son las negaciones correctas de los predicados:

- a) *A todo el mundo le gusta el helado.*
 - i. A nadie le gusta el helado

- ii. A todo mundo le disgusta el helado
 - iii. Alguien no adora el helado
 - b) *Algunas fotografías están viejas y deslavadas.*
 - i. Todas las fotografías no están viejas ni están deslavadas
 - ii. Algunas fotografías no están viejas o deslavadas
 - iii. Todas las fotografías no son viejas ni deslavadas
- 4.4.15.- Muestre que $\forall x(P(x) \wedge Q(x))$ y $\forall x(P(x)) \wedge \forall x(Q(x))$ son lógicamente equivalentes.
- 4.4.16.- Muestre que $\neg\forall x(P(x) \rightarrow Q(x))$ y $\exists x(P(x) \wedge \neg Q(x))$ son lógicamente equivalentes.
- 4.4.17.- Transforme las siguientes fórmulas mediante equivalencias lógicas, de manera que las negaciones sólo figuren frente a predicados.
- a) $\forall x\exists y\neg\forall z\exists w(P(x, w) \vee Q(z, y)) \rightarrow \neg\exists v\forall u\neg R(u, v).$
 - b) $\neg\forall x\exists y\neg\forall w\exists z(P(x, y) \vee \neg Q(x) \rightarrow \exists w\neg T(a, w)).$
 - c) $\neg\exists x\forall y\neg\exists w\forall z\neg(\neg P(x, y) \wedge Q(x) \rightarrow \forall wT(x, w)).$
 - d) $\neg(\neg\exists x\forall y(\neg T(y) \wedge R(z, x) \rightarrow G(x, z)) \rightarrow \forall w\neg\exists vP(v, a, w)).$
 - e) $\neg(\forall x\exists w\neg(\neg P(a, x) \vee R(c, w)) \wedge \exists z\neg\forall y(T(b, z) \wedge \neg Q(y, a))).$

4.5. Predicados y tipos

Frecuentemente un dominio de interpretación se compone de diversas clases bien determinadas de objetos; por ejemplo, círculos y cuadrados, alumnos y profesores, animales y vegetales. En estos casos, cuando se quiere especificar algo acerca de todos los individuos de cierta clase de objetos del universo, es conveniente hacer explícita su pertenencia a dicha clase particular mediante el uso de predicados llamados *calificadores* o *tipos*, los cuales denotan clases de objetos. Para expresar propiedades de un tipo específico de objeto se usa un juicio universal afirmativo. Por ejemplo, si el universo son los mamíferos y queremos hablar de una propiedad universal $P(x)$ de los felinos, como pudiese ser maullar o ser cuadrúpedo, la especificación $\forall xP(x)$ no da la suficiente información y es preferible usar un tipo $F(x)$ para felinos, con lo que la especificación sería $\forall x(F(x) \rightarrow P(x))$. Similarmente, si la especificación es existencial, utilizamos un juicio existencial afirmativo, como en *algunos felinos beben leche* que se formaliza con $\exists x(F(x) \wedge BL(x))$.

El uso de tipos permite restringir o dirigir el rango de valores de una variable dada mediante el uso de un juicio afirmativo. Sin embargo, dado que su uso resulta muy frecuente y útil, es conveniente introducir una notación especial para tipos, de la siguiente manera:

$\forall x:A. P(x)$ en lugar de $\forall x(A(x) \rightarrow P(x))$

$\exists x:A. P(x)$ en lugar de $\exists x(A(x) \wedge P(x))$

A esta notación la denominamos *tipos abreviados*. Por ejemplo, si el universo son los números reales, el enunciado *para todo número real existe un natural mayor que él*, puede expresarse como sigue:

- Sin tipos: $\forall x \exists y (x < y)$ (inconveniente pues no da suficiente información).
- Con juicios afirmativos: $\forall x (R(x) \rightarrow \exists y (N(y) \wedge x < y))$.
- Con tipos abreviados: $\forall x : R. \exists y : N. x < y$.

Cuando un lenguaje tiene reglas sintácticas que manejen los tipos de las variables decimos que tenemos un lenguaje *fuertemente tipificado o tipado*. Entre los lenguajes de programación que son fuertemente tipificados tenemos a PASCAL, C++, JAVA, C#, C y HASKELL. En general, un lenguaje fuertemente tipificado nos da reglas muy estrictas respecto a cómo podemos combinar expresiones de distintos tipos en una misma expresión. Otros lenguajes, como LISP, PYTHON, PROLOG y SCHEME (RACKET), son lenguajes que no observan el concepto de tipo, de manera que las variables pueden tener distintos tipos durante la ejecución, dependiendo del estado de las mismas. Se dice entonces que el tipado es *dinámico*.

El concepto de tipo que se presenta corresponde únicamente a los tipos primitivos de un lenguaje. La lógica de predicados que estudiamos aquí es una lógica sin tipos, en el sentido de que los símbolos funcionales y de predicado, que se interpretan como operadores o relaciones, no tienen tipos explícitos. En nuestro caso, el uso de tipos es un mecanismo de ayuda y simplificación en la escritura de ciertas especificaciones.

Terminamos esta sección con algunos ejemplos.

Ejemplo 4.20. Vamos a especificar el tipo de los números naturales $N(x)$ con sus operaciones más comunes.

- El cero es un número natural: $N(0)$ o bien $0:N$
- El sucesor de un natural es un natural: $\forall x:N. (N(s(x)))$ o bien $\forall x:N. (s(x):N)$
- La suma de dos naturales es un natural: $\forall x:N. \forall y:N. (N(x + y))$
- El producto de dos naturales es un natural: $\forall x:N. \forall y:N. (N(x \cdot y))$
- El sucesor es una función inyectiva: $\forall x:N. \forall y:N. (s(x) = s(y) \rightarrow x = y)$
- Hay un natural menor o igual que todos los naturales: $\exists y:N. \forall x:N. y \leq x$

Por último, un ejemplo más cercano a las especificaciones usuales en computación.

Ejemplo 4.21. Se desea especificar propiedades de un sistema de archivos de computadora. Consideremos que el universo consta de archivos y directorios para lo cual definimos los tipos $A(x)$ y $D(x)$ (o simplemente A y D), con el predicado $C(x, y)$ como *el objeto x está contenido en el objeto y* y la función $n(x)$ que devuelve el nombre del objeto.

- Ningún directorio se contiene a sí mismo:

$$\forall x:D. \neg C(x, x)$$

- Si un directorio está contenido en otro, entonces el segundo no puede estar contenido en el primero (es decir, no hay directorios cíclicos):

$$\forall x:D. \forall y:D. (C(x, y) \rightarrow \neg C(y, x))$$

- Existe un directorio que no está contenido en ningún otro directorio (el directorio raíz):

$$\exists x:D. \forall y:D. \neg C(x, y)$$

- Existe un directorio vacío:

$$\exists x:D. \forall y. \neg C(y, x)$$

- Todo archivo está contenido en algún directorio:

$$\forall x:A. \exists y:D. C(x, y)$$

- Si dos archivos están en el mismo directorio entonces deben tener nombres distintos.

$$\forall x:A. \forall y:A. (\exists z:D. (C(x, z) \wedge C(y, z)) \rightarrow n(x) \neq n(y))$$

Ejercicios

4.5.1.- Formalice las siguientes especificaciones acerca de un tipo $A(x)$ y el tipo de listas de elementos de A , denotado $L(x)$. Debe agregar cualquier predicado, función o constante necesaria.

- La lista vacía es una lista de elementos de A .
- La operación de agregar un elemento de A al inicio de una lista dada es nuevamente una lista.
- La concatenación de dos listas es nuevamente una lista.
- La cabeza de una lista es un elemento de A .
- La cola de una lista es nuevamente una lista.
- La longitud de una lista es un número natural.

4.5.2.- Formalice las siguientes especificaciones acerca de un tipo $A(x)$ y el tipo de pilas de elementos de A , denotado $P(x)$. Debe agregar cualquier predicado, función o constante necesaria.

- Hay una pila vacía.
- La operación de agregar un elemento de A al tope de una pila es una pila.
- El tope de la pila es un elemento de A .
- La operación de eliminar el elemento en el tope de la pila devuelve una pila.

Parte II

Inducción y recursión

Inducción y recursión

5

Existen muchos universos o dominios que contienen un número ilimitado de elementos que sin embargo pueden ser contados. Por ejemplo, el universo de números naturales, el dominio de expresiones lógicas (tomadas con las variables proposicionales de un alfabeto), el dominio de programas escritos en ciertos lenguajes de programación y, en general, todas las estructuras comunes en computación como son las listas o los árboles, conocidas como estructuras discretas. Estos universos se conocen como conjuntos infinitos numerables y son de gran utilidad en Ciencias de la Computación y Matemáticas Discretas. *Numerable* en este caso significa que se pueden contar, en el sentido que dado un elemento del conjunto, es posible determinar cuál es el elemento siguiente. Sin embargo, por ser infinitos, no es posible describirlos elemento por elemento pues nunca terminaríamos, ni tampoco podemos probar alguna propiedad acerca de ellos tratando de mostrarla para cada elemento particular. En este capítulo tratamos dos técnicas muy relacionadas entre sí, la inducción y la recursión, las cuales sirven para probar y definir propiedades sobre dominios infinitos numerables.

Iniciamos el capítulo definiendo de manera formal a los números naturales, mostrando algunas definiciones recursivas de funciones sobre los mismos, y discutiendo el llamado *método de inducción matemática* y algunas de sus variantes. Posteriormente nos ocuparemos de las definiciones recursivas de conjuntos y funciones en un ámbito más general. Estas definiciones recursivas son generalizaciones de las utilizadas en números naturales a cualquier dominio infinito numerable y que además esté bien fundado¹. En la última sección nos ocupamos en generalizar el principio de inducción matemática mediante la llamada inducción estructural en algunas estructuras discretas muy necesarias en programación, como lo son los árboles, y las cadenas o listas finitas.

¹*dominio bien fundado* se refiere, en términos muy generales, a un conjunto tal que cualquiera de sus subconjuntos tiene un primer elemento.

5.1. Los números naturales

El conjunto de números naturales² $\mathbb{N} = \{0, 1, 2, \dots\}$ es quizás el ejemplo más sencillo de un conjunto infinito numerable. Pero siendo infinito, ¿cómo podemos justificar su construcción y manejo en computación?

Empecemos con su construcción. En la vida diaria utilizamos los símbolos $0, \dots, 9$ para representar los primeros diez números naturales, llamados dígitos, mientras que los siguientes números se definen a partir de los dígitos mediante ciertas reglas. Formalmente sólo utilizaremos el dígito 0 ya que los demás números se construirán utilizando la función *sucesor*. El sucesor de un número n , escrito $s(n)$, es simplemente el número que le sigue a n en la sucesión de números naturales o, equivalentemente, $s(n) = n + 1$ (como aún no definimos la suma evitaremos su uso). Obsérvese que la función *sucesor* es general y no depende del dominio de los números naturales; por ejemplo, los días y meses tienen sucesor.

La definición de números naturales será nuestro primer ejemplo de definición recursiva.

- 0 es un número natural.
- Si n es un número natural, entonces el sucesor de n , denotado $s(n)$, es un número natural.
- Éstos y sólo éstos son números naturales.

Esta definición es recursiva, pues en la segunda cláusula se está usando a n , que suponemos es un natural, para poder concluir que $s(n)$ también lo es; es decir, estamos usando lo definido en la misma definición; en la siguiente sección trataremos con detalle este tipo de definiciones. La tercera cláusula puede parecer extraña y con frecuencia se omite en las definiciones. Sin embargo es necesaria para garantizar que un objeto es un número natural únicamente si fue construido usando las cláusulas anteriores. Esto es imprescindible para que funcionen los principios de inducción.

Según la definición anterior el conjunto de números naturales es

$$\mathbb{N} = \{0, s(0), s(s(0)), \dots\}$$

De esta manera hemos construido un conjunto infinito en el sentido que siempre podremos construir cualquier número de sus elementos y, en particular, cualquier elemento.

Diferencia entre sintaxis y semántica

Estructuralmente es claro que el conjunto de números naturales recién definido es infinito; sin embargo, si le damos cierto significado a la función *sucesor* pudiera darse el caso de que los elementos $s(s(\dots s(n) \dots))$ no sean todos distintos. Por ejemplo, si hablamos de los días de la semana,

$$s(s(s(s(s(s(\text{lunes})))))) = \text{lunes}.$$

²La inclusión del 0 en los naturales no es aceptada universalmente, especialmente por matemáticos; sin embargo, aquellos académicos que cultivan la investigación en lógica, postulan que $0 \in \mathbb{N}$.

Para indicar que el conjunto es infinito es necesario postular dos propiedades más que garanticen que todos los naturales son distintos.

- $\forall n (s(n) \neq 0)$.
- $\forall n \forall m (s(n) = s(m) \rightarrow n = m)$.

Estas dos propiedades aseguran que el 0 no es sucesor de nadie y que la función sucesor es inyectiva.

A continuación nos gustaría definir las operaciones básicas suma y producto; esto se hará nuevamente usando recursión. Para la suma tenemos la siguiente definición:

- $\forall n (n + 0 = n)$.
- $\forall n \forall m (m + s(n) = s(m + n))$.

La importancia de una definición recursiva es que podemos extraer de ella un programa para calcular dicha función; veamos un ejemplo sencillo:

$$\begin{aligned}
 3 + 2 &= s(s(s(0))) + s(s(0)) \\
 &= s(s(s(s(0))) + s(0)) \\
 &= s(s(s(s(s(0))) + 0)) \\
 &= s(s(s(s(s(0))))) \\
 &= 5
 \end{aligned}$$

Finalmente, el producto de dos naturales se define recursivamente como sigue:

- $\forall n (n \times 0 = 0)$.
- $\forall n \forall m (n \times s(m) = n \times m + n)$.

Más adelante daremos más ejemplos de funciones definidas recursivamente.

5.1.1. Axiomas de Peano

Las fórmulas lógicas definidas a continuación constituyen los llamados *axiomas de Peano*; éstos fueron propuestos por el matemático italiano Giuseppe Peano en 1889 y conforman una definición abstracta del conjunto de los números naturales. A continuación los resumimos.

- 0 es un número natural. (P-1)
- Si n es un número natural entonces $s(n)$ es un número natural. (P-2)
- $\forall n (s(n) \neq 0)$. (P-3)
- $\forall m \forall n ((s(n) = s(m)) \rightarrow (n = m))$. (P-4)

También contamos, en este mismo formato, con las definiciones recursivas de las operaciones de suma y producto de los números naturales recién discutidas y que recapitulamos a continuación:

$$\bullet \forall m(m + 0 = m). \quad (\text{D-1})$$

$$\bullet \forall m \forall n(m + s(n) = s(m + n)). \quad (\text{D-2})$$

$$\bullet \forall m(m \times 0 = 0). \quad (\text{D-3})$$

$$\bullet \forall m \forall n(m \times s(n) = m \times n + m). \quad (\text{D-4})$$

El último axioma de Peano es el llamado *axioma de inducción* y nos dice que para cualquier predicado P la siguiente expresión es válida:

$$\bullet P(0) \wedge \forall n(P(n) \rightarrow P(s(n))) \rightarrow \forall n(P(n)). \quad (\text{P-5})$$

Esta expresión formaliza el principio de inducción para números naturales. Este principio es muy conocido y de gran importancia en matemáticas discretas y ciencias de la computación y, en general, en todas las matemáticas. A continuación discutimos su validez y desarrollamos algunos ejemplos de su uso.

5.2. Inducción en los números naturales

Sea P una propiedad acerca de números naturales, tal que para cualquier natural n , al suponer que $P(n)$ ha sido probada, es fácil cerciorarse de la validez de $P(s(n))$; es decir, se prueba la validez de la misma propiedad para el siguiente número. Si además podemos probar $P(0)$, entonces el axioma (P-5) nos permite concluir que nuestra propiedad es válida para todos los números naturales. Esto se justifica al existir para cada número natural n_0 una derivación de $P(n_0)$ construida como se ve enseguida (usando $1, 2, 3, \dots$ en lugar de $s(0), s(s(0)), s(s(s(0))), \dots$):

- | | | |
|-------|---------------------------------------|------------------------------|
| 1. | $P(0)$ | Hipótesis. |
| 2. | $\forall n(P(n) \rightarrow P(s(n)))$ | Hipótesis. |
| 3. | $P(0) \rightarrow P(1)$ | Instanciación $n := 0$ en 2. |
| 4. | $P(1)$ | Modus Ponens 1, 3. |
| 5. | $P(1) \rightarrow P(2)$ | Instanciación $n := 1$ en 2. |
| 6. | $P(2)$ | Modus Ponens 4, 5. |
| 7. | $P(2) \rightarrow P(3)$ | Instanciación $n := 2$ en 2. |
| 8. | $P(3)$ | Modus Ponens 6, 7. |
| | \vdots | |
| k . | $P(n_0)$ | |

Estas derivaciones generan la siguiente regla de inferencia, la cual también se deriva del axioma (P-5), que se muestra en la siguiente página.

$$\frac{P(0) \quad \forall n(P(n) \rightarrow P(s(n)))}{\forall n(P(n))}$$

donde P es un predicado acerca de números naturales.

Veamos algunos ejemplos de su uso.

Ejemplo 5.1. Mostrar que 0 es identidad por la izquierda de la suma; esto es

$$\forall n(0 + n = n).$$

Demostración.

Base: Demostrar $P(0)$: $(0 + 0) = 0$.

Esto se cumple por (P-2).

Hipótesis de inducción: Suponemos $P(n)$: $0 + n = n$.

Paso inductivo: Demostrar $P(s(n))$: $0 + s(n) = s(n)$.

$$\begin{aligned} 0 + s(n) &= s(0 + n) && \text{(D-2)} \\ &= s(n) && \text{(hipótesis de inducción)} \end{aligned}$$

Ejemplo 5.2. Mostrar que la suma es conmutativa, esto es:

$$\forall m(\forall n(n + m = m + n)).$$

Demostración. Demostrar $n + m = m + n$.

En este caso hay dos variables, m y n , y podríamos hacer inducción sobre cualquiera de ellas. Sin embargo, es necesario escoger una sola para la inducción, dejando a la otra como un parámetro fijo durante toda la prueba, ya que no debemos hacer inducción sobre ambas variables. Aquí elegimos hacer inducción sobre m .

Base: Demostrar $P(0)$: $0 + n = n + 0$.

$$\begin{aligned} 0 + n &= n && \text{(ejemplo (5.2))} \\ &= n + 0 && \text{(D-1)} \end{aligned}$$

Hipótesis de inducción: Suponemos $P(m)$: $m + n = n + m$.

Paso inductivo: Demostrar $P(s(m))$: $s(m) + n = n + s(m)$.

Tomando el lado derecho:

$$\begin{aligned} n + s(m) &= s(n + m) && \text{(D-2)} \\ &= s(m + n) && \text{(hipótesis de inducción)} \end{aligned}$$

Quisiéramos que el siguiente paso fuera

$$s(m + n) = s(m) + n.$$

Pero esto no es consecuencia de los axiomas ni de resultados anteriores. Por lo tanto, lo tenemos que demostrar. Lo haremos usando inducción natural, ahora sobre n .

La propiedad a probar, con m fija, es:

$$\forall m \forall n (s(m+n) = s(m) + n)$$

Base: Demostrar $P(0)$: $s(m+0) = s(m) + 0$.

$$\text{Esto se cumple porque } s(m+0) = s(m) = s(m) + 0. \quad (\text{D-1})$$

Hipótesis de inducción: Suponemos $P(n)$: $s(m+n) = s(m) + n$.

Paso inductivo: Demostrar $P(s(n))$: $s(m+s(n)) = s(m) + s(n)$.

$$s(m+s(n)) = s(s(m+n)) \quad (\text{D-2})$$

$$= s(s(m) + n) \quad (\text{hipótesis de inducción})$$

$$= s(m) + s(n) \quad (\text{D-2})$$

Generalización Universal: $\forall n (s(m+s(n)) = s(m) + s(n))$

Generalización universal sobre m : $\forall m \forall n (m+n = n+m)$.

Ejemplo 5.3. Sea $H_n = 0$ para $n = 0$; $H_{n+1} = 1 + 2H_n$ para $n > 0$. Demostrar que

$$\forall n (H_n = 2^n - 1).$$

Demostración. Verificamos primero para la base, que en este caso es 0:

Base: Demostrar, usando la definición dada, $P(0)$, o sea, $H_0 = 2^0 - 1$.

$$H_0 = 0 \quad (\text{por la definición de } H_n \text{ con } n = 0)$$

$$2^0 - 1 = 1 - 1 \quad (\text{por aritmética})$$

$$= 0 \quad \checkmark$$

Hipótesis de inducción: Suponemos $P(n)$: $H_n = 2^n - 1$.

Paso inductivo: Verificar que $H_{n+1} = 2^{n+1} - 1$.

$$H_{n+1} = 1 + 2H_n \quad (\text{definición de } H_{n+1})$$

$$= 1 + 2(2^n - 1) \quad (\text{hipótesis de inducción})$$

$$= 1 + 2 \cdot 2^n - 2 \cdot 1 \quad (\text{aritmética})$$

$$= 1 + 2^{n+1} - 2 \quad (\text{aritmética})$$

$$= 2^{n+1} - 1 \quad (\text{aritmética}) \quad \checkmark$$

Ejemplo 5.4. Mostrar que para toda n , $2(n+2) \leq (n+2)^2$.

Demostración.

Base: Para $n = 0$, $P(0)$,

$$2(0+2) = 2(2) = 4 \leq 4 = 2^2 = (0+2)^2$$

Hipótesis de inducción: Suponemos $P(n)$: $2(n+2) \leq (n+2)^2$.

Paso inductivo: Corroborar que se cumple $P(n+1)$. Empezamos por el lado izquierdo:

$$\begin{aligned}
 2((n+1)+2) &= 2n+2+4 && \text{(aritmética)} \\
 &= 2(n+2)+2 && \text{(aritmética)} \\
 &\leq (n+2)^2+2 && \text{(hipótesis de inducción)} \\
 &= n^2+4n+4+2 && \text{(aritmética)} \\
 &< n^2+4n+4+2n+5 && n > 0 \text{ (por lo que al agregarlo se} \\
 & && \text{mantiene la desigualdad)} \\
 &= n^2+6n+9 && \text{(aritmética)} \\
 &= (n+3)^2 && \text{(aritmética)} \\
 &= ((n+1)+2)^2 && \checkmark
 \end{aligned}$$

Ejemplo 5.5. Demuestre que $n^3 + 2n$ es divisible entre 3.

Demostración. Que $n^3 + 2n$ sea divisible entre 3 quiere decir que se puede expresar como $n^3 + 2n = 3 \cdot k$ para algún natural k .

Base: Para $n = 0$, $0^3 + 2 \cdot 0 = 0 + 0 = 0 = 3 \cdot 0$, por lo que $0^3 + 2(0)$ es divisible por 3.

Hipótesis de inducción: Suponemos $P(n)$: $n^3 + 2n = 3 \cdot k$ para alguna k .

Paso inductivo: Tomemos $n+1$ y veamos cómo se expresa $(n+1)^3 + 2(n+1)$.

$$\begin{aligned}
 (n+1)^3 + 2(n+1) &= n^3 + 3n^2 + 3n + 1 + 2n + 2 && \text{(álgebra)} \\
 &= (n^3 + 2n) + 3n^2 + 3n + 3 && \text{(asociatividad y} \\
 & && \text{conmutatividad)} \\
 &= 3 \cdot k + 3(n^2 + n + 1) && \text{(hipótesis de inducción)} \\
 &= 3 \cdot k' && \text{(con } k' = k + n^2 + n + 1)
 \end{aligned}$$

Conclusión: De esto, $\forall n (n^3 + 2n \text{ es divisible entre } 3)$.

5.2.1. Cambio de la base de la inducción

En algunos casos la base de la inducción no es necesariamente el cero o el uno; esto no es una falla en el método de inducción, sino que la propiedad utilizada es válida a partir de cierto número n_0 , lo cual genera un principio similar, presentado aquí como regla de inferencia:

$$\frac{P(n_0) \quad \forall n (n \geq n_0 \rightarrow P(n) \rightarrow P(s(n)))}{\forall n (n \geq n_0 \rightarrow P(n))}$$

Ejemplo 5.6. Mostrar que $2^n < n!$, para $n \geq 4$.

Demostración.

Base: $P(4)$: $2^4 = 16 < 24 = 4!$.

Hipótesis de inducción: Suponer $P(n)$: $2^n < n!$.

Paso inductivo: Demostrar $P(n+1)$: $2^{n+1} < (n+1)!$ para $n \geq 4$.

$$\begin{aligned}
 2^{n+1} &= 2 \times 2^n && \text{(aritmética)} \\
 &< 2 \times n! && \text{(hipótesis de inducción)} \\
 &< (n+1) \times n! && 2 < n+1, \text{ (pues } n \geq 4) \\
 &= (n+1)! && \text{(definición de } (n+1)! \text{)}
 \end{aligned}$$

Ejemplo 5.7. Mostrar que cualquier cantidad mayor a 3 pesos puede pagarse usando únicamente monedas de 2 y 5 pesos.

Demostración.

Base: $P(4)$: $4 = 2 \cdot 2$ de manera que \$4 puede pagarse con dos monedas de \$2.

Hipótesis de inducción: $P(n)$: Suponemos que \$ n pueden pagarse con monedas de \$2 y \$5.

Paso inductivo: $P(n+1)$: Demostrar que \$($n+1$) pueden pagarse con monedas de \$2 y \$5.

Por la hipótesis de inducción tenemos que $\$n = k \cdot 2 + m \cdot 5$. Es decir, \$ n se pagan con k monedas de \$2 y m monedas de \$5. Tenemos dos casos:

- $m = 0$. Es decir, \$ n se pagaron solamente con monedas de \$2. En este caso,

$$n+1 = k \cdot 2 + 1 = (k-2) \cdot 2 + 2 \cdot 2 + 1 = (k-2) \cdot 2 + 5,$$

de donde si \$ n se pagaron con k monedas de \$2, tenemos que \$($n+1$) se pagan con $k-2$ monedas de \$2 y una moneda de \$5. Obsérvese que estamos separando dos monedas de \$2 para completar \$5; esto puede hacerse debido a que $k \geq 2$ ya que $n \geq 4$ y $m = 0$ (no tenemos más que monedas de \$2 y al menos dos).

- $m > 0$. Es decir, \$ n se pagaron con al menos una moneda de \$5.

$$\begin{aligned}
 n+1 &= k \cdot 2 + m \cdot 5 + 1 = k \cdot 2 + (m-1) \cdot 5 + 5 + 1 = \\
 &= (k+3) \cdot 2 + (m-1) \cdot 5
 \end{aligned}$$

de donde \$($n+1$) se pagan con $k+3$ monedas de \$2 y $m-1$ monedas de \$5. Obsérvese que separamos una moneda de \$5 para obtener \$6 que se pagan con tres monedas de \$2; esto puede hacerse pues $m \geq 1$.

De los ejemplos anteriores podemos obtener un esquema general para una prueba por inducción:

1. Enunciar el uso del principio de inducción. De esta manera el lector comprenderá de qué tipo de prueba se trata.

2. Definir un predicado apropiado $P(n)$, de manera que la meta a probar sea $\forall n P(n)$. Con frecuencia este predicado puede extraerse de la afirmación matemática o en español que se desea probar.
3. Mostrar que la base de la inducción $P(0)$ (o $P(n_0)$) es cierta.
4. Enunciar la hipótesis de inducción $P(n)$.
5. Probar la implicación $P(n) \rightarrow P(n+1)$; esto se conoce como paso inductivo.
6. Invocar el principio de inducción y concluir que $\forall n P(n)$.

Cualquier prueba por inducción debe tener todos estos pasos y en este orden.

5.2.2. Inducción completa

Si pensamos en una prueba por inducción de acuerdo al principio original (P-5) y a la derivación lógica dada en la página 216 para justificar el método, al probar $P(m)$ para un número cualquiera m tuvimos que probar antes $P(0), P(1), \dots, P(m-1)$, es decir, la propiedad P tuvo que verificarse para todos los números anteriores a m . Esta información podría ser útil y necesaria para probar $P(m+1)$, ya que en algunos casos no basta con la información inmediata anterior $P(m)$. Esta observación da lugar al *principio de inducción fuerte* o *completa* que enunciamos aquí como regla de inferencia.

$$\frac{\forall n \left(\forall m (m < n \rightarrow P(m)) \rightarrow P(n) \right)}{\forall n (P(n))}$$

Obsérvese que en este caso no hay una base explícita de la inducción. Si instanciamos $n = 0$ entonces la premisa de la regla resulta equivalente a $P(0)$, puesto que la fórmula $\forall m (m < 0 \rightarrow P(m))$ es cierta al tratarse de una implicación con antecedente falso ($m < 0$) con $m \in \mathbb{N}$. Al probar el paso inductivo para $n = 0$ no hay hipótesis disponible para usarse, por lo que $P(0)$ debe ser probado como en casos anteriores. Sin embargo, esto no es necesario en la mayoría de los casos.

Este principio permite partir la prueba del paso inductivo en dos o más casos más pequeños, cualesquiera que éstos sean.

Ejemplo 5.8. Sea d elemento neutro del operador binario \circ ; es decir $\forall x (x \circ d = d \circ x = x)$. Mostrar que cualquier expresión e de la forma $e = e_1 \circ e_2 \circ \dots \circ e_k$ que contenga una o más presencias de d debe ser igual a d .

Por demostrar: Sea $P(n)$ la proposición que cualquier expresión con n presencias de \circ y al menos una presencia de d es igual a d .

Base: Veamos cuáles son las posibles expresiones de la forma requerida con una presencia de \circ y al menos una presencia de d :

- a) $x \circ d$ b) $d \circ x$.

Por la definición del operador \circ tenemos

$$\forall x(x \circ d = d \circ x = d).$$

por lo que $P(1)$ se cumple.

Hipótesis de inducción: Supongamos $P(m)$ para $m < n$. Es decir, cualquier expresión de la forma requerida con $m < n$ presencias de \circ y al menos una presencia de d es igual a d .

Paso inductivo: Sea x una expresión de la forma requerida con $n > 0$ operadores \circ que contiene al menos una presencia de d ; entonces $x = x_1 \circ x_2$ donde x_1, x_2 son expresiones con menos de n operadores \circ y alguna de x_1, x_2 contiene una presencia de d , digamos que es x_1 . En tal caso, por la hipótesis de inducción se tiene $x_1 = d$, de donde tenemos $x = x_1 \circ x_2 = d \circ x_2 = d$, lo cual completa el paso inductivo. Obsérvese que la prueba es totalmente análoga si es x_2 quien contiene una presencia de d .

Ejemplo 5.9. Demostrar que cualquier $n \geq 2$ es primo o producto de primos.

Por demostrar: Sea $P(n)$ la proposición: n es primo o producto de primos. Queremos probar que

$$\forall n(n \geq 2 \rightarrow P(n)).$$

Base: $P(2)$: Para $n = 2$ tenemos que 2 es primo, por lo que se cumple $P(2)$.

Hipótesis de inducción: Supongamos $P(m)$ para $m < n$. Es decir, cualquier número $m < n$ es primo o producto de primos.

Paso inductivo: Si n es primo hemos terminado. Si no lo es, n se puede escribir como $n = m \cdot q$ con $1 < m < n$, $1 < q < n$ y por la hipótesis de inducción, ambos, m y q , son primos o producto de primos, de donde $n = m \cdot q$ también lo es.

Obsérvese que en este ejemplo se combinan la inducción completa y el cambio de base, al iniciar en $n = 2$.

Ejercicios

5.2.1.- Demuestre, usando las definiciones de suma y producto dadas al inicio de esta sección, que $s(0)$ es la identidad para la multiplicación; esto es

$$\forall m(m \times s(0) = m)$$

5.2.2.- Demuestre las siguientes propiedades de la suma y el producto:

- Asociatividad de la suma: $\forall m \forall n \forall k(m + (n + k) = (m + n) + k)$.
- Asociatividad del producto: $\forall m \forall n \forall k(m \times (n \times k) = (m \times n) \times k)$.
- Neutro izquierdo del producto: $\forall n(0 \times n = 0)$.
- Conmutatividad del producto: $\forall m \forall n(m \times n = n \times m)$.

5.2.3.- Demuestre, usando las definiciones de suma y producto dadas al inicio de esta sección, $\forall m \forall n (s(m) \times s(n) = m \times n + s(m) + n)$.

5.2.4.- Demuestre que para toda n ,
$$\sum_{k=1}^n k = \frac{n(n+1)}{2}.$$

5.2.5.- Demuestre que para toda n ,
$$\sum_{k=1}^n k^3 = \left(\frac{n(n+1)}{2} \right)^2.$$

5.2.6.- Demuestre que para toda n ,
$$5+8+11+\cdots+(3n+2) = \frac{1}{2} (3n^2 + 7n).$$

5.2.7.- Use inducción, así como las leyes de conmutatividad y asociatividad, para demostrar que

$$a_1 + (a_2 + (a_3 + \dots + (a_{n-1} + a_n) \dots)) = a_n + (a_{n-1} + (\dots + (a_2 + a_1) \dots))$$

5.2.8.- Sea $n > 3$ un número natural. Sea m el entero mayor que es menor o igual que $(n+2)/2$; es decir, $m = \lfloor (n+2)/2 \rfloor$. Veamos una pequeña tabla con los valores de n y m a continuación.

n	$\lfloor (n+2)/2 \rfloor$	m
2	2	2
5	3	3
6	4	4
7	4	4

Demuestre que dados más de m enteros en el conjunto $\{1, 2, \dots, n\}$, tres de los enteros en este conjunto tienen la propiedad de que alguno de los tres es la suma de los otros dos.

5.2.9.- Demuestre que para toda $n \geq 0$,
$$\sum_{k=0}^n 9 \cdot 10^k = 10^{n+1} - 1.$$

5.2.10.- Use inducción matemática para demostrar que para todo entero n , $n < 2^n$.

5.2.11.- Demuestre que para todo entero positivo n , existe un entero positivo con n dígitos, que es divisible entre 5^n y tal que todos sus dígitos son impares. Que un entero p sea divisible entre otro entero q , denotado $p \mid q$, quiere decir que al dividir p entre q , el residuo es 0. Dicho de otra manera,

$$\text{dados } p, q \in \mathbb{Z}^+, p \mid q \rightarrow \exists m \in \mathbb{Z}^+ \text{ tal que } p = q \cdot m.$$

Veamos algunos ejemplos de la proposición en la tabla a continuación:

n	Entero con n dígitos	$p = q \cdot m$
$P(1)$	5	$5 = 5^1(1)$
$P(2)$	75	$75 = 5^2(3) = 25 \cdot 3$
$P(3)$	375	$375 = 5^3(3) = 125 \cdot 3$
$P(4)$	9375	$5^4(15) = 625 \cdot 15$

5.2.12.- Demuestre que para todo entero $n \geq 0$ y $z \neq 1$, $\sum_{k=0}^n z^k = \frac{z^{n+1} - 1}{z - 1}$

5.2.13.- Demostrar que para todo entero $n > 6$, $3^n < n!$.

5.2.14.- Demuestre que para todo natural n , $\sum_{k=1}^n k(k!) = (n+1)! - 1$

5.2.15.- Demuestre que cualquier número natural es par o impar, es decir:

$$\forall n \in \mathbb{N} (\exists k \in \mathbb{N} (n = 2k) \vee \exists r \in \mathbb{N} (n = 2r + 1))$$

5.2.16.- Demuestre que la suma de los cubos de tres números naturales consecutivos es divisible entre 9

5.2.17.- Después de transcurrir n meses en un experimento de invernadero, el número p_n de plantas de un tipo particular satisface las ecuaciones:

$$p_0 = 3 \quad p_1 = 7 \quad p_{n+2} = 3p_{n+1} - 2p_n \quad n \in \mathbb{N}$$

Demuestre que para cualquier $n \in \mathbb{N}$, $p_n = 2^{n+2} - 1$.

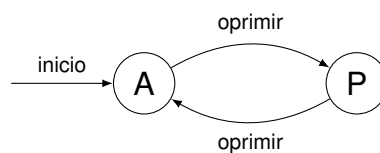
5.2.18.- Demuestre que para cualquier $n \in \mathbb{N}$, el número $2^{2n} - 1$ es múltiplo de 3.

5.2.19.- Los autómatas finitos son un modelo muy útil para dispositivos en software o hardware. Un autómata finito es un dispositivo que se puede encontrar, en un momento dado, en alguno de entre un número finito de *estados*. El objetivo de los estados es *recordar* una porción relevante de la historia del sistema. Como sólo hay un número finito de estados, la historia completa no puede ser registrada, por lo que se deberá diseñar con cuidado para recordar los aspectos relevantes.

En cada estado, el autómata recibe posibles señales, que lo pueden hacer cambiar de estado. El autómata inicia siempre en un estado designado como *inicial*, y dependiendo del estado en el que está, puede emitir una señal.

Podemos modelar un apagador muy sencillo con un autómata finito. El autómata tiene dos estados, el de *apagado* y el de *prendido*, que es lo que el autómata tiene que recordar. Cuando se oprime el apagador, dependiendo en cuál de los dos estados esté, va a pasar al otro: Si está en *apagado* pasa a *prendido* y si está en *prendido* pasa a *apagado*. El estado inicial es *apagado*. Podemos modelar el autómata con lo que se conoce como un *diagrama de transiciones*, como se muestra en la figura 5.1 a continuación. Como se puede ver en ella, los estados están representados por círculos, mientras que el resultado de oprimir el apagador, que corresponde a una *transición*, se representa con una flecha que va de un estado al otro. El estado inicial es al que llega la flecha identificada con inicio.

Figura 5.1. Autómata correspondiente a un apagador



Deberá demostrar que los siguientes enunciados para describir el comportamiento del autómata se cumplen:



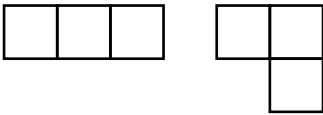
- $S_1(n)$: El autómata está en el estado **A** (de *apagado*) después de haber oprimido el botón n veces, si y sólo si n es par.
- $S_2(n)$: El autómata está en el estado **P** (de *prendido*) si y sólo si n es impar.

Se tiene que hacer una demostración doble de inducción, ya que hay que hacer inducción sobre los dos casos posibles de la definición.

5.2.20.- Un *poliominó* es una pieza formada por cuadrados iguales unidos entre sí por al menos una línea (se excluyen los que estén unidos sólo por un vértice). Los poliominós se clasifican según el número de cuadrados que los forman; así, tenemos los monominós formados por un cuadrado, los dominós por dos cuadrados, triminós por tres, tetraminós por cuatro, pentaminós con cinco, los hexaminós con seis, los heptaminós con siete, En ciencias de la computación a la propiedad de este tipo de uniones, donde se requiere que cuadrados adyacentes compartan un lado, se conoce también como 4-conectividad.

Al número de cuadrado que tiene el poliominó se le llama el *orden* de la figura. Según el número de cuadrados en el poliominó tendremos un número distinto de figuras con ese número de cuadrados. En la siguiente tabla consideraremos el número de poliominós *libres*, donde dos poliominós son diferentes si uno no es el reflejo, la rotación o la traslación del otro. En la tabla 5.1 mostramos los poliominós libres de órdenes 1 a 5.

Tabla 5.1. Poliominós libres de órdenes 1 a 5

Nombre	Número	Figuras
monominó	1	
dominós	1	
triminós	2	

(Continúa en la siguiente página)

Tabla 5.1. Poliomínos libres de órdenes 1 a 5 (Continúa...)

Nombre	Número	Figuras
tetraminós	5	
pentaminós	12	

Consideremos el triminó en forma de L. Consideremos un tablero de $2^n \times 2^n$ cuadros en el que eliminamos un cuadro. Demostrar que el resto del tablero puede ser cubierto con triminós en forma de L.

- 5.2.21.- Considere el siguiente juego: tenemos una fila de monedas dispuesta sobre una mesa. Ganamos si conseguimos eliminar todas las monedas de la fila de acuerdo con el siguiente procedimiento: podemos empezar eliminando cualquier moneda que muestre la cara (es decir, sol), pero en ese caso tenemos que dar la vuelta a las monedas (una o dos) que están al lado de la que acabamos de eliminar. Después buscamos otra moneda con sol y usamos el mismo procedimiento. La pregunta es: ¿Cómo tiene que ser una fila de monedas para poder ganar el juego? Examinando casos con un número pequeño de monedas defina una propiedad que dé la respuesta y demuéstrela usando inducción fuerte.

5.3. Definiciones recursivas

Una definición recursiva es aquella en la cual el concepto definido figura en la definición misma. Esto puede parecer problemático –de hecho, introduce problemas matemáticos profundos si dicho uso o autorreferencia³ se utiliza sin cuidado–. Sin embargo, usado bajo ciertas restricciones, este método de definición, al que llamaremos en adelante *recursión*, proporciona una herramienta sumamente útil tanto en matemáticas como en ciencias de la computación. En particular, la gran mayoría de los tipos de datos usuales en programación como listas o árboles, así como diversas funciones sobre los mismos, pueden definirse recursivamente.

Para que una definición recursiva sea válida, en el sentido que genere tipos de datos o funciones que no causen ciclos infinitos de evaluación, debe constar de dos partes:

- Un conjunto de casos base, los cuales son casos simples donde la definición se da directamente, es decir, sin usar autorreferencia.
- Un conjunto de reglas recursivas donde se define un nuevo elemento de la definición en términos de elementos anteriores (más pequeños) ya definidos.

Además de estas dos partes, la definición debe constar de una cláusula que asegure que las dos anteriores son las únicas formas de obtener el concepto, objeto o función definida. Esta cláusula puede omitirse en el entendido de que siempre está presente.

La definición en los casos base nos da un punto de partida, al proporcionar una definición directa, mientras que las reglas recursivas nos permiten construir nuevos casos a partir de los básicos de una manera iterativa.

Es importante observar que las únicas definiciones recursivas que consideramos válidas son aquellas donde las reglas recursivas se definen en términos de *elementos anteriores*. Por ejemplo, la siguiente definición de una función:

$$\begin{aligned}f(0) &= 1 \\f(n+1) &= f(n+2)\end{aligned}$$

no es válida, puesto que la definición en $n+1$ está dada en términos de un elemento *posterior* a $n+1$, a saber $n+2$. En particular, f resulta indefinida en cualquier valor distinto de cero. Definiciones como la anterior se llaman *recursivas generales* y por lo general causan ciclos infinitos en programación.

Ya hemos visto definiciones recursivas del conjunto de números naturales, así como de algunas funciones sobre este mismo tipo de datos, como la suma o el producto. Veamos algunos ejemplos más en las páginas que siguen.

Ejemplo 5.10. Dada una persona x , la relación *ser descendiente de* x en el dominio de las personas se define como se muestra en la siguiente página:

³Palabra con un significado técnico preciso que no aparece en el diccionario.

- i. Si y es hijo de x entonces y es descendiente de x .
 - ii. Si y es descendiente de x y z es hijo de y entonces z es descendiente de x .
 - iii. Nadie más es descendiente de x .
-

Ejemplo 5.11. Dados dos números naturales n y m , la relación n es menor que m , denotada $n < m$, se define como sigue:

- i. $0 < s(m)$.
 - ii. Si $n < m$, entonces $s(n) < s(m)$,
 - iii. Ningún otro par de números está en la relación $<$.
-

Obsérvese que en el ejemplo anterior, la recursión se hace sobre el número n dejando a m fijo y declarándolo explícitamente como un número sucesor $s(k)$, puesto que la relación $n < 0$ no sucede nunca.

Ejemplo 5.12. El conjunto de *fórmulas bien construidas* de la lógica proposicional⁴ se define como sigue:

- i. Una variable proposicional es una *fórmula bien construida*.
 - ii. Las constantes lógicas true y false son *fórmulas bien construidas*.
 - iii. Si A y B son *fórmulas bien construidas*, entonces $(\neg A)$, $(A \vee B)$, $(A \wedge B)$ y $(A \rightarrow B)$ son *fórmulas bien construidas*.
 - iv. Ninguna expresión que no sea construida con estas reglas es una *fórmula bien construida*.
-

Ejemplo 5.13. El conjunto de *expresiones aritméticas* se define como sigue:

- i. Todos los enteros y todos los nombres de variables son *expresiones aritméticas*.
 - ii. Si A y B son *expresiones aritméticas* entonces $(-A)$, $(A + B)$, $(A - B)$, $(A \times B)$ y $(A \div B)$ son *expresiones aritméticas*⁵.
 - iii. Éstas y sólo éstas son *expresiones aritméticas*.
-

Ejemplo 5.14. El tipo de datos de *listas finitas* $[a_1, \dots, a_n]$ con elementos a_i en un conjunto A (o simplemente *listas*) se define de la siguiente forma:

- i. La lista vacía es una *lista* y se denota con $[]$ o *nil*.
 - ii. Si $a \in A$ y ℓ es una *lista*, entonces $\text{cons}(a, \ell)$ es una *lista*. A a se le llama la *cabeza* y a ℓ la *cola* de la lista.
 - iii. Sólo éstas son *listas*.
-

Frecuentemente se usa la notación $(a : \ell)$ para $\text{cons}(a, \ell)$. Por ejemplo, si consideramos al conjunto $A = \{1, 3, 6, 10, 15, 21, 28\}$, la lista $[10, 6, 1, 6]$ que contiene a los elementos 10, 6, 1, 6, en ese orden, se representa como se muestra a continuación.

$(10 : (6 : (1 : (6 : []))))$

Es conveniente notar que en las listas se admiten repeticiones e importa el orden, a diferencia de lo que sucede con los conjuntos. Por lo tanto, una lista formada con elementos de un

⁴Se les conoce también como *wff*, del inglés *well formed formulas*.

⁵El operador \div entrega cociente entero, lo que permitiría definir expresiones aritméticas en los enteros.

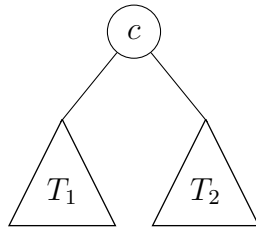
conjunto finito puede ser infinita (nótese que este último tipo de listas no está en nuestra definición).

Tenemos varias opciones para representar a una lista con un único elemento. $cons(a, [])$ corresponde a una lista con un primer elemento a y donde la cola de la lista es la lista vacía. También podemos denotar a una lista con un solo elemento, de manera abreviada, como $[a]$, lo que haremos en adelante.

Ejemplo 5.15. El tipo de datos *árboles binarios* con todos los nodos etiquetados por elementos de un conjunto A se define como sigue:

- i. Un árbol vacío es un *árbol binario* y se denota por $void$.
- ii. Si T_1 y T_2 son *árboles binarios* y c es un elemento de A , entonces $tree(T_1, c, T_2)$ es un *árbol binario*, donde T_1 es el *subárbol izquierdo* y T_2 es el *subárbol derecho*. Al nodo etiquetado con c se le llama la *raíz* del árbol.
- iii. Nada más es un árbol binario.

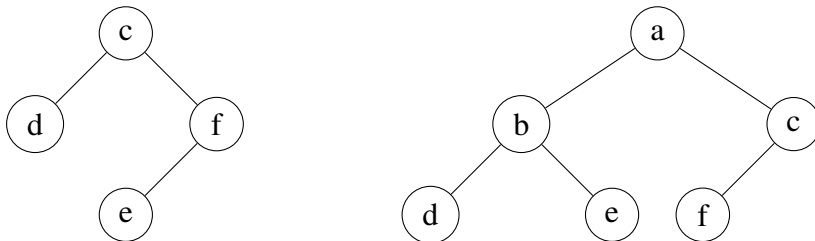
Por ejemplo, la expresión $tree(T_1, c, T_2)$ corresponde a la siguiente figura:



En el caso en que $T_1 = T_2 = void$, tenemos un árbol de la forma $tree(void, c, void)$, el cual se conoce como una *hoja*.

Por otro lado, cuando tenemos la figura del árbol podemos extraer de ella la expresión. ¿A qué expresión corresponde cada uno de los árboles en la figura 5.2?

Figura 5.2. Gráficas de árboles binarios



Los ejemplos anteriores muestran definiciones recursivas de tipos de datos usuales en computación como números naturales, expresiones lógicas o aritméticas, listas o árboles, o bien de relaciones como el orden usual entre números. En la siguiente sección mostramos definiciones recursivas de funciones que involucran a los tipos de datos recién definidos.

5.3.1. Definición de funciones recursivas

La definición recursiva de tipos de datos permite definir funciones sobre los mismos utilizando la técnica de *casa o apareamiento de patrones*⁶: cada cláusula de la definición del tipo de datos introduce un patrón, el cual es un esquema sintáctico bien definido que se utiliza para definir un caso de la función en cuestión. Listamos a continuación los patrones básicos de cada tipo de datos definido previamente:

- Números naturales: $0, s(n)$.
- Fórmulas bien construidas: $p, \text{true}, \text{false}, \neg A, A \wedge B, A \vee B, A \rightarrow B$.
- Expresiones aritméticas: $n, x, (-A), (A + B), (A - B), (A \times B), (A \div B)$
- Listas: $[], (a : \ell)$
- Árboles binarios: $\text{void}, \text{tree}(T_1, c, T_2)$

De esta manera, para definir, por ejemplo, una función f sobre las listas, es suficiente definir los casos para $f([])$ y para $f((a : \ell))$.

Veamos a continuación algunos ejemplos de funciones definidas sobre los tipos de datos recién presentados y cuyas implementaciones se dan mediante apareamiento de patrones. En cada caso se da primero una especificación que proporciona una definición directa, seguida de una implementación mediante una función recursiva f definida mediante patrones. En algunos ejemplos nos puede resultar claro que la definición recursiva de f cumple con la especificación dada en cada caso. Sin embargo, debemos mostrar esto formalmente para cada ejemplo, proceso que discutiremos en la siguiente sección.

Ejemplo 5.16. Exponenciación de números naturales.

Especificación: $\text{pot}(n, m) = n^m$

Implementación recursiva:

- $f(n, 0) = 1$
 - $f(n, s(m)) = f(n, m) \cdot n$
-

Ejemplo 5.17. Factorial de un número natural.

Especificación: $\text{fac}(0) = 1; \text{fac}(n) = n \cdot (n - 1) \cdot \dots \cdot 2 \cdot 1$.

Implementación recursiva:

- $f(0) = 1$
 - $f(s(n)) = s(n) \cdot f(n)$
-

Ejemplo 5.18. Suma de los elementos de una lista de números.

Especificación: $\text{suml}([a_1, \dots, a_n]) = a_1 + a_2 + \dots + a_n$

Implementación recursiva:

- $f([]) = 0$
 - $f((a : \ell)) = a + f(\ell)$
-

⁶En inglés *pattern matching*, También se usa acoplamiento.

Ejemplo 5.19. Producto de los elementos de una lista de números.

Especificación: $prodl([a_1, \dots, a_n]) = a_1 \cdot a_2 \cdot \dots \cdot a_n$

Implementación recursiva:

- $f([]) = 1$
 - $f((a : \ell)) = a \cdot f(\ell)$
-

Ejemplo 5.20. Longitud de una lista.

Especificación: $long([a_1, \dots, a_n]) = n$

Implementación recursiva:

- $f([]) = 0$
 - $f((a : \ell)) = 1 + f(\ell)$
-

Ejemplo 5.21. El operador binario \sqcup devuelve la concatenación de dos listas.

Especificación: $[a_1, \dots, a_k] \sqcup [b_1, \dots, b_j] = [a_1, \dots, a_k, b_1, \dots, b_j]$

Implementación recursiva:

- $f([], \ell_2) = \ell_2$
 - $f((a : \ell_1), \ell_2) = (a : f(\ell_1, \ell_2))$
-

Ejemplo 5.22. Reversa de una lista.

Especificación: $rev([a_1, \dots, a_k]) = [a_k, \dots, a_1]$

Implementación recursiva:

- $f([]) = []$
 - $f((a : \ell)) = f(\ell) \sqcup [a]$
-

En algunos casos, la especificación de una función no puede darse de forma directa mediante una ecuación como en los casos anteriores, sino que tiene que darse con palabras como en los siguientes ejemplos.

Ejemplo 5.23. Conectivos de una fórmula proposicional.

Especificación: nc es la función que calcula el número de conectivos en una fórmula de la lógica proposicional. Por ejemplo $nc(p \rightarrow \neg q \vee r) = 3$.

Implementación recursiva:

- $f(p) = 0$
 - $f(\text{true}) = f(\text{false}) = 0$
 - $f(\neg A) = 1 + f(A)$
 - $f(A \wedge B) = 1 + f(A) + f(B)$
 - $f(A \vee B) = 1 + f(A) + f(B)$
 - $f(A \rightarrow B) = 1 + f(A) + f(B)$
-

Ejemplo 5.24. Intercambio de conectivos lógicos.

Especificación: *ccd* es la función que recibe una fórmula proposicional *A* y devuelve la fórmula obtenida a partir de *A* al intercambiar los conectivos \wedge y \vee en *A*. Por ejemplo $ccd(\neg p \vee q \rightarrow r \wedge s) = \neg p \wedge q \rightarrow r \vee s$.

Implementación recursiva:

- $f(p) = p$
 - $f(\text{true}) = \text{true}$
 - $f(\text{false}) = \text{false}$
 - $f(\neg A) = \neg f(A)$
 - $f(A \wedge B) = f(A) \vee f(B)$
 - $f(A \vee B) = f(A) \wedge f(B)$
 - $f(A \rightarrow B) = f(A) \rightarrow f(B)$
-

Ejemplo 5.25. Número de presencias atómicas en una fórmula.

Especificación: *at* es la función que calcula el número de presencias de fórmulas atómicas que figuran en una fórmula. Por ejemplo,

$$at(q \wedge \neg p \rightarrow r \vee p) = 4; at(q \wedge (\text{true} \vee \neg(r \rightarrow \text{false} \wedge t))) = 5.$$

Implementación recursiva:

- $f(p) = 1$
 - $f(\text{true}) = f(\text{false}) = 1$
 - $f(\neg A) = f(A)$
 - $f(A \wedge B) = f(A) + f(B)$
 - $f(A \vee B) = f(A) + f(B)$
 - $f(A \rightarrow B) = f(A) + f(B)$
-

Ejemplo 5.26. Número de nodos en un árbol binario.

Especificación: *nn* es la función que recibe un árbol binario *t* y calcula el número de nodos que hay en *t*.

Implementación recursiva:

- $f(\text{void}) = 0$
 - $f(\text{tree}(T_1, c, T_2)) = 1 + f(T_1) + f(T_2)$
-

El ejemplo que sigue involucra a dos tipos de datos, el de las fórmulas proposicionales y el de listas de fórmulas.

Ejemplo 5.27. Lista de subfórmulas en una fórmula proposicional.

Especificación: *sf* es la función que devuelve la lista de subfórmulas de una fórmula proposicional *A*. Por ejemplo,

$$sf(\neg(p \rightarrow q) \wedge r) = [\neg(p \rightarrow q) \wedge r, \neg(p \rightarrow q), p \rightarrow q, p, q, r].$$

$$sf(p \vee \neg(p \rightarrow s)) = [p \vee \neg(p \rightarrow s), p, \neg(p \rightarrow s), p \rightarrow s, p, s].$$

Implementación recursiva:

- $f(p) = [p]$
 - $f(\text{true}) = [\text{true}]$
 - $f(\text{false}) = [\text{false}]$
 - $f(\neg A) = (\neg A : f(A))$
 - $f(A \wedge B) = ((A \wedge B) : f(A) \sqcup f(B))$
 - $f(A \vee B) = ((A \vee B) : f(A) \sqcup f(B))$
 - $f(A \rightarrow B) = ((A \rightarrow B) : f(A) \sqcup f(B))$
-

Ejemplo 5.28. Altura o profundidad de un árbol binario.

Especificación: la profundidad o altura de un nodo x en un árbol binario T se define como la distancia (número de líneas) existente entre x y la raíz de T en la representación gráfica de T . La profundidad o altura de un árbol T se define como la altura máxima entre los nodos de T más uno. ht es la función que calcula la profundidad de un árbol binario.

Implementación recursiva:

- $f(\text{void}) = 0$
 - $f(\text{tree}(T_1, c, T_2)) = 1 + \text{máx}\{f(T_1), f(T_2)\}$
-

Como ya mencionamos, en cada caso debemos cerciorarnos formalmente que la definición recursiva dada por la función correspondiente realmente cumple con la especificación dada. Para el caso de funciones que involucren a los números naturales, esto puede lograrse mediante el principio de inducción matemática. Como ejemplo, veamos que la definición recursiva del factorial en verdad cumple la especificación.

Ejemplo 5.29. Demostrar que la definición recursiva de f en el ejemplo 5.17 es correcta. Es decir, para todo número natural n , se cumple $f(n) = fac(n)$.

Base: $n = 0$. Tenemos $f(0) = 1 = fac(0)$.

Hipótesis de inducción: $f(n) = fac(n)$.

Paso inductivo: queremos demostrar que $f(s(n)) = fac(s(n))$.

$$\begin{aligned}
 f(s(n)) &= s(n) \cdot f(n) && \text{(definición de } f\text{)} \\
 &= s(n) \cdot fac(n) && \text{(hipótesis de inducción)} \\
 &= (n+1) \cdot (n \cdot (n-1) \cdot \dots \cdot 2 \cdot 1) && \text{(definición de } fac(n) \text{ y asociatividad)} \\
 &= fac(s(n)) && \text{(definición de } fac(s(n)))
 \end{aligned}$$

En conclusión $f(n) = fac(n)$ para todo número natural n .

Ahora bien, para el caso de funciones definidas sobre otro tipo de datos, ¿cómo podemos probar que la especificación se satisface con la implementación recursiva?

De las definiciones y pruebas por inducción de las operaciones de suma y producto, así como de la prueba del ejemplo anterior, se observa una fuerte relación entre el principio de inducción matemática y las definiciones recursivas que involucren números naturales.

Cada propiedad de la definición recursiva, como cumplir con una especificación dada, puede mostrarse mediante el principio de inducción. Esta relación puede generalizarse a distintas estructuras o tipos de datos definidos recursivamente, lo que haremos en la siguiente sección.

Ejercicios

5.3.1.- Para cada una de las siguientes definiciones recursivas de una función f , calcule

$f(1)$, $f(2)$, $f(3)$, $f(4)$ y $f(5)$,

a) $f(n+1) = f(n) + 2$

b) $f(n+1) = 3 * f(n)$

c) $f(n+1) = 2^{f(n)}$

d) $f(n+1) = f(n)^2 + f(n) + 1$

donde, en todos los casos, $f(0) = 1$.

5.3.2.- Un conjunto T de números naturales se define recursivamente con las siguientes condiciones:

a) 2 pertenece a T .

b) Si x pertenece a T , entonces también $x + 3$ y $2x$ pertenecen a T .

c) Solamente estos números pertenecen a T

¿Cuáles de los números 6, 7, 12, 19 pertenecen a T ? Justifique su respuesta

5.3.3.- Defina recursivamente cada uno de los siguientes conjuntos de números naturales.

a) El conjunto I de números impares.

b) El conjunto D de múltiplos de 3.

c) El conjunto $S = \{4, 7, 10, 13, \dots\} \cup \{3, 6, 9, 12, \dots\}$.

5.3.4.- Defina recursivamente cada uno de los siguientes conjuntos de listas sobre un conjunto fijo A .

a) El conjunto de listas de longitud par.

b) El conjunto de listas de longitud impar.

c) El conjunto de listas de longitud arbitraria pero que tienen un mismo elemento. Por ejemplo $[a]$, $[a, a]$, \dots , $[a, a, \dots, a]$.

5.3.5.- Defina recursivamente cada uno de los siguientes conjuntos de listas sobre

$A = \{0, 1\}$.

a) El conjunto T de listas que alternan sus elementos. Por ejemplo, $[\]$, $[0]$, $[1]$, $[0, 1]$, $[1, 0, 1]$, $[0, 1, 0, 1, 0]$ son todas elementos de T .

b) El conjunto O de listas de longitud impar de tal forma que la única lista que empieza con 0 es $[0]$.

- 5.3.6.- Defina recursivamente una función *toma* que reciba un número n y una lista ℓ y devuelva la lista que contiene los primeros n elementos de ℓ . Por ejemplo,

`toma 3 [a, b, c, d, e, f, g] = [a, b, c]`

- 5.3.7.- Defina recursivamente una función *quita* que reciba un número n y una lista ℓ , y devuelva la lista que quita los primeros n elementos de ℓ . Por ejemplo,

`quita 3 [a, b, c, d, e, f, g] = [d, e, f, g]`.

- 5.3.8.- Defina una función recursiva *replica* que toma un elemento a y un número n como entrada, y devuelve la lista que contiene n veces a a como elemento, es decir, que cumpla la siguiente especificación:

`replica(a, n) = [a, a, a, ..., a]`

donde a aparece n veces.

- 5.3.9.- Demuestre la siguiente propiedad de las funciones *suml* (véase el ejemplo 5.18) y *replica*, del ejercicio anterior: para cualesquiera $k, n \in \mathbb{N}$, se cumple

`suml(replica(k, n)) = k · n`

- 5.3.10.- Defina recursivamente el predicado *elem* cuya especificación formal es la siguiente:

$elem\ x\ \ell$ si y sólo si x es elemento de la lista ℓ .

- 5.3.11.- Considere la siguiente función misteriosa:

`mist [] l2 = []`
`mist (a:l1) l2 = if elem a l2 then mist l1 l2`
`else (a:mist l1 l2)`

- a) Muestre la evaluación de `mist [1,2,3] [4,2,5,1]` y
`mist [3,7] [2,4]`.

- b) Dé una especificación formal para *mist*.

- 5.3.12.- Defina recursivamente las funciones *last*, que devuelve el último elemento de una lista no vacía, y *elast*, que devuelve la lista resultante al borrar el último elemento de una lista no vacía.

- 5.3.13.- Considere la siguiente función misteriosa:

`mist [] = []`
`mist (a:l) = if elem a l then mist l`
`else (a:mist l)`

- a) Muestre la evaluación de `mist [1,2] ymist [3,7,8,7,3]`.

- b) Dé una especificación formal para *mist*

- 5.3.14.- La función *eln* recibe una fórmula A y devuelve la fórmula que resulta de reemplazar en A cada negación $\neg B$ por $B \rightarrow \text{false}$. Por ejemplo,

$eln(\neg p \wedge \neg(q \vee r)) = (p \rightarrow \text{false}) \wedge (q \vee r \rightarrow \text{false})$.

Defina la función *eln* recursivamente.

- 5.3.15.- La función quita-implicaciones qi toma una fórmula proposicional A y devuelve una fórmula B donde no figura el conectivo implicación. Por ejemplo,

$$qi((p \rightarrow q) \rightarrow \neg s) = \neg(\neg p \vee q) \vee \neg s.$$

Defina recursivamente a qi .

- 5.3.16.- Defina una función $leaf$ que dado un elemento a , construya el árbol binario cuyo único elemento es a .

- 5.3.17.- Defina recursivamente una función $espejo$ que tome un árbol binario t y devuelva el árbol obtenido al intercambiar los subárboles izquierdo y derecho, propagando esta misma operación a los subárboles de los subárboles, y así sucesivamente. Esta operación obtiene el reflejo de t en un espejo.

Por ejemplo, si $t = tree(tree(leaf(4), 3, leaf(5)), 1, leaf(2))$ entonces

$$espejo(t) = tree(leaf(2), 1, tree(leaf(5), 3, leaf(4))).$$

- 5.3.18.- Defina recursivamente una función hmi que devuelva la hoja más a la izquierda en un árbol binario.

- 5.3.19.- Sea $L = \{ (,), +, -, x, y \}$. Dé una definición recursiva del lenguaje EA de expresiones aritméticas totalmente parentizadas sobre L . Por ejemplo, las siguientes son expresiones de EA : $x, (x + y), (x - x), (x + (y - x))$; mientras que las siguientes no lo son: $x + y, x - y, x + (x - y)$

- 5.3.20.- Usando la definición de EA del ejercicio anterior, dé una definición recursiva de las siguientes funciones:

- nv que cuenta el número de presencias de variables en una expresión.
- npi, npd que cuentan, respectivamente, el número de paréntesis izquierdos y derechos en una expresión.
- $prefs$ que devuelve una lista con todos los prefijos de una expresión. Observe que un prefijo no es necesariamente una expresión válida. Por ejemplo $(x +$ es prefijo de $(x + y)$ y no es una EA válida.

5.4. Inducción estructural

Para demostrar propiedades acerca de estructuras definidas recursivamente en el sentido descrito en la página 227, es posible recurrir a la inducción matemática, definiendo una medida en la estructura en cuestión, lo que se hace mediante un número natural. Entre las medidas que podemos mencionar están la longitud de una lista, la profundidad de un árbol o el número de conectivos en una fórmula proposicional. Esto es posible debido a que las reglas recursivas de la definición en cuestión se dan en términos de elementos estructuralmente más simples, por lo que su medida será menor, y la hipótesis de inducción podrá emplearse. Sin embargo, en la mayoría de los casos, el uso de una medida complica las pruebas, además de que la elección de una medida incorrecta podría resultar en una

prueba fallida. Otra posibilidad es generalizar el principio de inducción completa mediante la definición de un orden en los tipos de datos, el cual debe ser bien fundado, es decir, no debe contener sucesiones descendentes infinitas. Sin embargo, el problema de decidir si una estructura particular es bien fundada no siempre es fácil de resolver.

En lugar de las alternativas anteriores es posible utilizar los llamados *principios de inducción estructural*, basados en las reglas base y recursivas de la definición de un tipo de datos, así como en el análisis de los patrones básicos introducidos por aquéllas.

El esquema general del principio de inducción estructural es el siguiente: Sean A un conjunto o tipo de datos definido recursivamente y P una propiedad acerca de los elementos de A . Para probar $P(x)$ válida para todo elemento de A , deben seguirse los siguientes pasos:

- **Base de la inducción:** Si a es un elemento de A generado por una regla básica, entonces debemos probar directamente la validez de $P(a)$.

Si x es un elemento de A construido mediante alguna regla recursiva a partir de elementos anteriores⁷ x_1, \dots, x_n , entonces procedemos como sigue:

- **Hipótesis de inducción:** Suponer $P(x_1), \dots, P(x_n)$.
- **Paso inductivo:** Probar $P(x)$.

En este caso, el principio de inducción estructural permite concluir $\forall x P(x)$.

La validez de este principio se sigue de un principio más general de la teoría de los conjuntos, llamado de *inducción bien fundada (inducción B-F)*, cuya discusión cae fuera de los objetivos de este libro.

En nuestro caso, el principio de inducción estructural debe adaptarse a cada conjunto o tipo de datos particular. En las siguientes secciones lo ejemplificamos para los casos de listas, árboles y fórmulas proposicionales.

5.4.1. Inducción en listas

El tipo de datos `lista` es uno de los más comunes en ciencias de la computación. Este tipo de datos se definió recursivamente en el ejemplo 5.14 y genera el siguiente principio de inducción estructural:

Sea P una propiedad acerca de listas; si se desea probar $P(xs)$ para toda lista xs , basta proceder como sigue:

Base de la inducción: Probar $P([])$ directamente.

Hipótesis de inducción: Suponer $P(xs)$.

Paso inductivo: Probar $P((a : xs))$ para cualquier a .

Si este es el caso, el principio de inducción para listas permite concluir $P(xs)$ para cualquier lista xs .

⁷Es decir, elementos estructuralmente más simples.

Para ilustrar el uso de la inducción en listas probamos enseguida algunas propiedades de las operaciones en listas.

Proposición 5.1 La función recursiva f dada en el ejemplo 5.21 calcula la concatenación de dos listas. Es decir, para cualesquiera listas xs, ys , $f(xs, ys) = xs \sqcup ys$.

Demostración. Inducción sobre xs .

Base de la inducción: $xs = []$. Tenemos $f([], ys) = ys = [] \sqcup ys$.

Hipótesis de inducción: $f(xs, ys) = xs \sqcup ys$.

Paso inductivo: Debemos mostrar que para cualquier a , $f((a:xs), ys) = (a:xs) \sqcup ys$.

$$\begin{aligned} f((a:xs), ys) &= (a : f(xs, ys)) && \text{(definición recursiva de } f) \\ &= (a : (xs \sqcup ys)) && \text{(hipótesis de inducción)} \\ &= (a : xs) \sqcup ys && \text{(especificación de } \sqcup) \end{aligned}$$

En conclusión $f(xs, ys) = xs \sqcup ys$ para cualesquiera listas xs, ys . □

De manera similar podemos probar la correctud de todas las definiciones recursivas de listas dadas en los ejemplos de la sección 5.3.1.

Como ya probamos que la implementación de la concatenación \sqcup es correcta, podemos usarla de ahora en adelante, como en el caso de la siguiente proposición.

Proposición 5.2 La operación de concatenación \sqcup en listas cumple las siguientes propiedades:

- **Asociatividad:** $xs \sqcup (ys \sqcup zs) = (xs \sqcup ys) \sqcup zs$
- **Longitud:** $long(xs \sqcup ys) = long(xs) + long(ys)$

Demostración. Probamos la asociatividad mediante inducción sobre la lista xs .

Sea $P(xs)$ la propiedad $xs \sqcup (ys \sqcup zs) = (xs \sqcup ys) \sqcup zs$.

Base de la inducción: $xs = []$; debemos mostrar que $[] \sqcup (ys \sqcup zs) = ([] \sqcup ys) \sqcup zs$.

Procedemos, saliendo del lado izquierdo de la igualdad:

$$\begin{aligned} [] \sqcup (ys \sqcup zs) &= ys \sqcup zs && \text{(definición recursiva de } \sqcup) \\ &= ([] \sqcup ys) \sqcup zs && \text{(definición recursiva de } \sqcup, \text{ } ys = [] \sqcup ys) \end{aligned}$$

Hipótesis de inducción: $xs \sqcup (ys \sqcup zs) = (xs \sqcup ys) \sqcup zs$.

Paso inductivo: Sea a un elemento de A ; debemos mostrar que

$$(a : xs) \sqcup (ys \sqcup zs) = ((a : xs) \sqcup ys) \sqcup zs.$$

Procedemos, partiendo nuevamente del lado izquierdo de la igualdad:

$$\begin{aligned} (a : xs) \sqcup (ys \sqcup zs) &= \left(a : (xs \sqcup (ys \sqcup zs)) \right) && \text{(definición recursiva de } \sqcup) \\ &= \left(a : ((xs \sqcup ys) \sqcup zs) \right) && \text{(hipótesis de inducción)} \\ &= (a : (xs \sqcup ys)) \sqcup zs && \text{(definición recursiva de } \sqcup) \\ &= ((a : xs) \sqcup ys) \sqcup zs && \text{(definición recursiva de } \sqcup) \end{aligned}$$

Así que por el principio de inducción para listas, concluimos que la operación \sqcup es asociativa. \square

La propiedad de longitud se demuestra similarmente.

Proposición 5.3 La operación reversa rev en listas cumple las siguientes propiedades:

- **Longitud:** $long(rev(xs)) = long(xs)$
- **Concatenación:** $rev(xs \sqcup ys) = rev(ys) \sqcup rev(xs)$
- **Involución:**⁸ $rev(rev(xs)) = xs$

Demostración. Mostramos la propiedad de involución mediante inducción sobre la lista xs , dejando las restantes como ejercicio.

Base de la inducción: $xs = []$. Como $rev([]) = []$ entonces

$$rev(rev([])) = rev([]) = [].$$

Hipótesis de inducción: $rev(rev(xs)) = xs$

Paso inductivo: Sea a un elemento de A ; mostraremos que

$$rev(rev((a : xs))) = (a : xs).$$

$$\begin{aligned}
 rev(rev((a : xs))) &= rev(rev(xs) \sqcup [a]) && \text{(definición recursiva de } rev\text{)} \\
 &= rev([a]) \sqcup rev(rev(xs)) && \text{(propiedad de concatenación de } rev\text{)} \\
 &= rev([a]) \sqcup xs && \text{(hipótesis de inducción)} \\
 &= [a] \sqcup xs && (rev([a]) = [a]) \\
 &= (a : []) \sqcup xs && ([a] = (a : [])) \\
 &= a : ([] \sqcup xs) && \text{(definición recursiva de } \sqcup\text{)} \\
 &= (a : xs) && \text{(definición recursiva de } \sqcup\text{)}
 \end{aligned}$$

Conclusión: Así que por el principio de inducción para listas se cumple

$$rev(rev(xs)) = xs \text{ para toda lista } xs. \quad \square$$

Pasamos ahora a ilustrar la inducción estructural en fórmulas proposicionales.

5.4.2. Inducción en fórmulas

El conjunto de fórmulas de la lógica proposicional se definió ya mediante una gramática, así como mediante la definición recursiva del ejemplo 5.12. Esta última forma de definirlo habilita un principio de inducción estructural de gran utilidad en lógica matemática. El principio de inducción estructural para fórmulas es el siguiente:

⁸En matemáticas se dice que una función $f : A \rightarrow A$ es una *involución* si $\forall x \in A$ se cumple $F(f(x)) = x$.

Sea P una propiedad acerca de fórmulas proposicionales. Si se desea probar $P(A)$ para toda fórmula A , basta proceder como sigue:

Base de la inducción: probar $P(q)$ directamente para cada variable proposicional q ; probar $P(\text{true})$ y probar $P(\text{false})$

Hipótesis de inducción: suponer $P(A)$ y $P(B)$.

Paso inductivo: probar $P(\neg A)$, $P(A \wedge B)$, $P(A \vee B)$ y $P(A \rightarrow B)$.

Conclusión: En tal caso el principio de inducción para fórmulas permite concluir $P(A)$ para cualquier fórmula A .

En este caso, debido a nuestros conocimientos de equivalencias lógicas, el paso inductivo puede simplificarse a probar $P(\neg A)$ y alguno de los casos para un operador binario, el cual se elige dependiendo de la propiedad P particular.

Demostramos a continuación algunas propiedades de las fórmulas proposicionales.

Proposición 5.4 Sea comp la siguiente función recursiva:

$$\text{comp}(p) = \neg p \quad (\text{i})$$

$$\text{comp}(\text{true}) = \text{false} \quad (\text{ii})$$

$$\text{comp}(\text{false}) = \text{true} \quad (\text{iii})$$

$$\text{comp}(\neg A) = \neg \text{comp}(A) \quad (\text{iv})$$

$$\text{comp}(A \vee B) = \text{comp}(A) \wedge \text{comp}(B) \quad (\text{v})$$

$$\text{comp}(A \wedge B) = \text{comp}(A) \vee \text{comp}(B) \quad (\text{vi})$$

Entonces, para toda fórmula C , se cumple $\text{comp}(C) \equiv \neg C$.

Demostración. Inducción sobre las fórmulas (el número de operadores en las fórmulas o el número de subfórmulas).

Base: C es atómica (no tiene operadores). Si $C = p$ entonces hay que mostrar $\text{comp}(p) \equiv \neg p$.

$$\text{comp}(p) = \neg p \quad (\text{por (i)})$$

$$\equiv \neg p \quad (\text{reflexividad de } \equiv)$$

Los casos para $C = \text{true}$ y $C = \text{false}$ son similares.

Hipótesis de inducción Supongamos que $\text{comp}(A) \equiv \neg A$ y $\text{comp}(B) \equiv \neg B$.

Paso inductivo: Dado nuestro conocimiento de las equivalencias lógicas (leyes de De Morgan), basta mostrar la propiedad para $\neg A$ y $A \wedge B$, que son fórmulas que tienen más operadores que A y B .

$$\text{comp}(\neg A) = \neg \text{comp}(A) \quad (\text{por (iv)})$$

$$\equiv \neg \neg A \quad (\text{hipótesis de inducción y equivalencia lógica})$$

$$\text{comp}(A \wedge B) = \text{comp}(A) \vee \text{comp}(B) \quad (\text{por (vi)})$$

$$\equiv \neg A \vee \neg B \quad (\text{hipótesis de inducción y equivalencia lógica})$$

$$\equiv \neg(A \wedge B) \quad (\text{De Morgan})$$

Conclusión: Por el principio de inducción para fórmulas, podemos concluir que $comp(C) \equiv \neg C$, para cualquier fórmula C . \square

Por último, demostramos una propiedad que relaciona a las funciones definidas en los ejemplos 5.20, 5.23 y 5.25.

Proposición 5.5 Si A es una fórmula proposicional, entonces la longitud de la lista de subfórmulas de A es igual a la suma del número de presencias de variables proposicionales de A con el número de conectivos que figuran en A . Es decir,

$$long(sf(A)) = at(A) + nc(A).$$

Demostración. Inducción sobre la fórmula A .

Base de la inducción: Sea $A = p$. Tenemos en A una presencia de la fórmula atómica p y ningún conectivo. Por lo tanto,

$$long(sf(p)) = long([p]) = 1 = 1 + 0 = at(p) + nc(p).$$

Para $A = \text{true}$ o $A = \text{false}$, la prueba es similar.

Hipótesis de inducción: Supongamos que

$$long(sf(A)) = at(A) + nc(A) \quad long(sf(B)) = at(B) + nc(B).$$

Paso inductivo: Probamos la propiedad para $\neg A$ y $A \rightarrow B$.

$$\begin{aligned} long(sf(\neg A)) &= long((\neg A : sf(A))) && \text{(definición de } sf) \\ &= 1 + long(sf(A)) && \text{(definición recursiva de } long) \\ &= 1 + (at(A) + nc(A)) && \text{(hipótesis de inducción)} \\ &= at(A) + (1 + nc(A)) && \text{(aritmética)} \\ &= at(\neg A) + (1 + nc(A)) && \text{(definición recursiva de } at) \\ &= at(\neg A) + nc(\neg A) && \text{(definición recursiva de } nc) \end{aligned}$$

$$\begin{aligned} long(sf(A \rightarrow B)) &= long(((A \rightarrow B) : sf(A) \sqcup sf(B))) && \text{(definición de } sf) \\ &= 1 + long(sf(A) \sqcup sf(B)) && \text{(definición recursiva de } long) \\ &= 1 + long(sf(A)) + long(sf(B)) && \text{(propiedad de } long \text{ de } \sqcup) \\ &= 1 + (at(A) + nc(A)) + (at(B) + nc(B)) && \text{(hipótesis de inducción)} \\ &= (at(A) + at(B)) + (1 + nc(A) + nc(B)) && \text{(aritmética)} \\ &= at(A \rightarrow B) + (1 + nc(A) + nc(B)) && \text{(definición recursiva de } at) \\ &= at(A \rightarrow B) + nc(A \rightarrow B) && \text{(definición recursiva de } nc) \end{aligned}$$

Conclusión: Por lo tanto, por el principio de inducción para fórmulas, para cualquier fórmula A se cumple

$$long(sf(A)) = at(A) + nc(A). \quad \square$$

Para finalizar este capítulo discutimos el principio de inducción para árboles binarios.

5.4.3. Inducción en árboles

La definición recursiva del tipo de datos de *árboles binarios* dada en el ejemplo 5.15, genera la siguiente versión del principio de inducción estructural:

Sea P una propiedad acerca de árboles binarios. Si se desea probar $P(T)$ para todo árbol T , basta proceder como sigue:

Base de la inducción: Probar $P(\text{void})$ directamente.

Hipótesis de inducción: Suponer $P(T_1)$ y $P(T_2)$.

Paso inductivo: Probar $P(\text{tree}(T_1, c, T_2))$.

Conclusión: En este caso, el principio de inducción para árboles permite concluir $P(T)$ para cualquier árbol T .

Veamos a continuación un par de ejemplos de pruebas mediante este principio de inducción.

Proposición 5.6 Cualquier árbol binario T con n nodos contiene exactamente $n + 1$ subárboles vacíos.

Demostración. Inducción sobre T .

Base: $T = \text{void}$. El número de nodos de T es 0; el número de subárboles vacíos es 1, pues T mismo es un subárbol binario vacío, por lo que tenemos que se cumple la fórmula $0 + 1 = 1$ subárboles vacíos.

Hipótesis de inducción: Si los árboles binarios T_1 y T_2 tienen n_1 y n_2 nodos respectivamente, entonces tienen $n_1 + 1$ y $n_2 + 1$ subárboles vacíos respectivamente.

Paso inductivo: Sea $T = \text{tree}(T_1, c, T_2)$ un árbol binario no vacío. El número de nodos de T es $n_1 + n_2 + 1$ (agregamos al nodo c). Queremos demostrar que T tiene $(n_1 + n_2 + 1) + 1 = n_1 + n_2 + 2$ árboles vacíos.

Es claro que los subárboles vacíos de T son subárboles de T_1 o de T_2 , por lo que se tiene que el número de subárboles vacíos de T es igual a la suma de los números de subárboles vacíos de T_1 y de T_2 ; pero por la hipótesis de inducción dicha suma es igual a $(n_1 + 1) + (n_2 + 1) = n_1 + n_2 + 2$.

Conclusión: Todos los árboles binarios con n nodos tienen $n + 1$ subárboles vacíos. \square

Proposición 5.7 Si T es un árbol binario con altura n , entonces tiene a lo más $2^n - 1$ nodos. Es decir, $nn(T) \leq 2^n - 1$.

Demostración. Inducción sobre T .

Base: $T = \text{void}$. En este caso la altura de T es 0 y $nn(T) = 0$, pues T no tiene nodos; por otro lado, $2^0 - 1 = 1 - 1 = 0$, con lo que queda demostrada la base de la inducción.

Hipótesis de inducción: Si el árbol T_i tiene altura n_i , entonces tiene a lo más $2^{n_i} - 1$ nodos, donde $i = 1, 2$.

Paso inductivo: Sea $T = \text{tree}(T_1, c, T_2)$. Recordemos que la altura de T es igual a $\max\{n_1, n_2\} + 1$. Sea $m = \max\{n_1, n_2\}$. Debemos demostrar que el máximo número de nodos en T es $2^{m+1} - 1$, es decir que $nn(T) \leq 2^{m+1} - 1$.

$$\begin{aligned}
 nn(T) &= nn(T_1) + nn(T_2) + 1 && \text{(definición recursiva de } nn) \\
 &\leq (2^{n_1} - 1) + (2^{n_2} - 1) + 1 && \text{(hipótesis de inducción)} \\
 &\leq 2^m - 1 + 2^m - 1 + 1 && \text{(aritmética: } 2^{n_1}, 2^{n_2} \leq 2^m) \\
 &= 2 \cdot 2^m - 1 && \text{(aritmética)} \\
 &= 2^{m+1} - 1 && \text{(leyes de exponentes)}
 \end{aligned}$$



Este resultado particular es muy utilizado en computación.

De los resultados y ejemplos de esta sección podemos concluir que cualquier estructura definida recursivamente tendrá asociado un principio de inducción, llamado el principio de inducción estructural, el cual debe utilizarse preferentemente para demostrar propiedades relacionadas a la estructura.

Ejercicios

5.4.1.- Para cada ejemplo de la sección 5.3.1 demuestre, mediante el principio de inducción estructural correspondiente, que la función especificada cumple con la implementación recursiva.

5.4.2.- Considere las siguientes especificaciones de dos funciones *spar* y *simp*, cuyo dominio y codominio son los números naturales, definidas como sigue:

$$spar(n) = 2 + 4 + 6 + \dots + 2n \quad simp(n) = 1 + 3 + 5 + \dots + (2n + 1).$$

- Proponga implementaciones recursivas f y g para *spar* y *simp*.
- Muestre que $f(n) = n(n + 1)$.
- Muestre que $g(n) = (n + 1)^2$.

5.4.3.- Definimos al conjunto de cadenas de la forma $a^m b a^m$ de la siguiente manera:

- b , la cadena representada por $a^0 b a^0$, está en el conjunto.
- Si w es una cadena en este conjunto, entonces awa también está en el conjunto.
- Éstas son las únicas formas de construir cadenas que cumplan con ser de la forma $a^m b a^m$.

Demuestre que todas las cadenas que pertenecen a este conjunto tienen un número impar de caracteres, utilizando los siguientes métodos:

- Inducción sobre la longitud de las cadenas.
- Definiendo y utilizando un principio de inducción estructural adecuado.

5.4.4.- Considere la definición recursiva de las expresiones aritméticas dada en el ejemplo 5.13. Enuncie el principio de inducción estructural correspondiente y utilícelo para demostrar que toda expresión aritmética tiene el mismo número de paréntesis izquierdos que derechos.

5.4.5.- Una cadena de caracteres es palíndroma si es de la forma ww^R , donde w^R es w escrita de atrás hacia adelante. Algunos ejemplos son 0110 y *aabaabaa*. Defina al conjunto de las cadenas palíndromas en forma recursiva y demuestre, mediante inducción estructural, que todas las cadenas palíndromas de este tipo tiene un número par de símbolos.

5.4.6.- Demuestre, mediante inducción para listas, lo siguiente:

- La propiedad de longitud enunciada en la proposición 5.2.
- Las propiedades de concatenación e idempotencia para la reversa, enunciadas en la proposición 5.3.

5.4.7.- La función *snoc* en listas se define como sigue:

$$\text{snoc } c \ [x_1, \dots, x_n] = [x_1, \dots, x_n, c].$$

- Dé una definición recursiva para *snoc*.
- Demuestre, usando la definición recursiva, que:

$$\text{snoc } c \ (xs \sqcup ys) = xs \sqcup (\text{snoc } c \ ys).$$

- Demuestre la siguiente propiedad que relaciona a *snoc* con la operación reversa *rev*:

$$\text{rev}(\text{snoc } c \ xs) = c : (\text{rev } xs).$$

5.4.8.- Considere la siguiente función misteriosa *mist*:

$$\begin{aligned} \text{mist} [] \ ys &= ys \\ \text{mist } (x:xs) \ ys &= \text{mist } xs \ (x:ys) \end{aligned}$$

- ¿Qué hace *mist*?
- Muestre que $\text{rev } xs = \text{mist } xs \ []$

5.4.9.- Demuestre las siguientes propiedades de la función *elem*:

- Si *elem a l* entonces *elem a (b : l)*.
- Si *elem a (b : l)* entonces $a = b$ o *elem a l*.
- elem a (l₁ \sqcup l₂)* si y sólo si *elem a l₁* o *elem a l₂*.
- Si *elem a l* entonces $\exists l_1, l_2 (l = l_1 \sqcup (a : l_2))$.

5.4.10.- Demuestre la siguiente propiedad de las funciones *toma* y *quita*, definidas en los ejercicios 5.3.6 y 5.3.7:

$$\forall \ell \forall n (\ell = \text{toma } n \ \ell \sqcup \text{quita } n \ \ell)$$

5.4.11.- Demuestre que la función *eln* del ejercicio 5.3.14 es idempotente. Es decir, demuestre que para toda fórmula *A*, $\text{eln}(\text{eln}(A)) = A$.

5.4.12.- Demuestre que la función *qi* del ejercicio 5.3.15 cumple que para toda fórmula *A*, $A \equiv qi(A)$.

5.4.13.- Considere la siguiente función misteriosa *mist*:

```

mist x [ ] = 0
mist x (y:ys) = if x=y then 1+(mist x ys)
                  else mist x ys

```

a) ¿Qué hace *mist*?

b) Pruebe que $\text{mist } x \ (xs ++ ys) = (\text{mist } x \ xs) + (\text{mist } x \ ys)$.

5.4.14.- La función *filter* en listas toma un predicado p y una lista xs y devuelve la lista de aquellos elementos de xs que cumplen con el predicado p . Por ejemplo, si p es la propiedad de “ser menor que 4” y $xs = [1, 0, 4, 2, 7, 3, 1]$, entonces *filter* $p \ xs$ devuelve $[1, 0, 2, 3, 1]$.

a) Dé una definición recursiva de la función *filter*.

b) Demuestre las siguientes propiedades de *filter* mediante inducción estructural.

i. $\text{filter } p \ (xs ++ ys) = \text{filter } p \ xs ++ \text{filter } p \ ys$

ii. $\text{rev } (\text{filter } p \ xs) = \text{filter } p \ (\text{rev } xs)$

5.4.15.- La función *fusl* en listas se especifica como sigue:

$\text{fusl } [x_1, \dots, x_n] [y_1, \dots, y_m] = [y_1, \dots, y_m, x_1, \dots, x_n]$

a) Dar una definición recursiva para *fusl*.

b) Demuestre, por inducción sobre listas, que:

$\text{rev } (\text{fusl } xs \ ys) = \text{fusl } (\text{rev } ys) \ (\text{rev } xs)$

donde *rev* es la función reversa de listas. Puede usar sin demostrar cualquier propiedad de la reversa, pero cualquier propiedad adicional de *fusl* que pretenda usar también debe ser demostrada.

5.4.16.- Este ejercicio concierne a la operación de sustitución textual para las fórmulas de la lógica proposicional.

a) Defina recursivamente la operación de sustitución textual $A[p := B]$.

b) Demuestre las siguientes propiedades mediante inducción para fórmulas:

- Si p no figura en A , entonces $A[p := B] = A$.

- Si $p \neq q$ y p no figura en C , entonces

$$A[p := B][q := C] = A[q := C][p := B[q := C]].$$

- Si $p \neq q$ y p no figura en B , entonces

$$A[q, p := B, C] = A[q := B][p := C].$$

5.4.17.- Sea A una fórmula de la lógica proposicional cuyos únicos conectivos son \wedge, \vee, \neg . Construimos la fórmula dual de A , denotada A_D , intercambiando \wedge con \vee y reemplazando cada variable p por su negación $\neg p$. Por ejemplo, si $A = (r \vee p) \wedge \neg q$, entonces $A_D = (\neg r \wedge \neg p) \vee \neg \neg q$.

a) Defina recursivamente una función *dual* tal que $dual(A) = A_D$.

b) Muestre que $\neg A \equiv A_D$ mediante inducción sobre las fórmulas.

5.4.18.- Defina recursivamente al conjunto de términos de la lógica de predicados y enuncie el principio de inducción estructural correspondiente.

5.4.19.- Defina recursivamente al conjunto de fórmulas de la lógica de predicados y enuncie el principio de inducción estructural correspondiente. Observe que este principio debe incluir al dado en la sección 5.4.2 para la lógica de proposiciones.

5.4.20.- Defina recursivamente las siguientes funciones para términos de la lógica de predicados:

a) $ctes(t)$ que devuelva el conjunto de constantes que figuran en t . Por ejemplo, $ctes(f(a, g(x, b))) = \{a, b\}$.

b) $vars(t)$ que devuelva el conjunto de variables que figuran en t . Por ejemplo, $vars(g(x, f(y), h(b))) = \{x, y\}$.

c) $funcs(t)$ que devuelva el conjunto de símbolos de función que figuran en t . Por ejemplo, $funcs(f(a, g(x, b))) = \{f, g\}$.

5.4.21.- La operación de sustitución textual puede extenderse a los términos de la lógica de predicados. Si t y r son términos y x es una variable entonces $t[x := r]$ denota a la sustitución textual de x por r en t . Por ejemplo,

$$f(a, x, g(y, x))[x := h(w)] = f(a, h(w), g(y, h(w))).$$

Realice lo siguiente:

a) Formule una definición recursiva de $t[x := r]$.

b) Muestre que si $x \notin vars(t)$, entonces $t[x := r] = t$.

c) Muestre que $vars(t[x := r]) = (vars(t) \setminus \{x\}) \cup vars(r)$.

5.4.22.- Defina recursivamente las siguientes funciones para fórmulas de la lógica de predicados:

a) $fv(A)$ que devuelva el conjunto de variables libres de A . Por ejemplo,

$$fv(\forall x P(x, y) \wedge \exists w Q(z, w)) = \{y, z\}$$

b) $bv(A)$ que devuelva el conjunto de variables ligadas de A . Por ejemplo,

$$bv(\forall x P(x, y) \wedge \exists w Q(z, w)) = \{x, w\}$$

c) $nq(A)$ que devuelva el número de cuantificadores que figuran en A .

Por ejemplo,

$$bv(\forall x \exists y P(x, y) \wedge \exists w \forall z Q(z, w)) = 4$$

5.4.23.- Reformule la prueba de la proposición 5.6 definiendo y utilizando una función recursiva nsu que calcule el número de subárboles vacíos de un árbol dado.

5.4.24.- Reformule la prueba de la proposición 5.7 sirviéndose de la función ht del ejemplo 5.28.

5.4.25.- Demuestre que el mínimo número de nodos en un árbol de altura n es n .

5.4.26.- Demuestre que el número máximo de hojas en un árbol de altura n es 2^{n-1} y que el máximo número de nodos internos es $2^{n-1} - 1$

5.4.27.- Demuestre que el siguiente recorrido en un árbol binario reporta a todos los nodos del árbol y siempre termina.

Reglas para reportar un árbol binario:

- a) Si el árbol es un árbol vacío, reporte void y regrese.
- b) Si el árbol es $tree(A, c, B)$, donde A y B son árboles binarios, entonces:
 - i. Reporte c .
 - ii. Reporte A .
 - iii. Reporte B .
 - iv. Termine.

5.4.28.- Defina recursivamente una función *aplana* que tome un árbol binario y devuelva la lista de sus nodos empezando por la raíz y siguiendo con los nodos del subárbol izquierdo y derecho recursivamente.

Por ejemplo, si

$$T = tree(tree(hoja(1), 6, hoja(2)), 5, tree(hoja(4), 9, void)),$$

donde $hoja(n) = tree(void, n, void)$, entonces $aplana(T) = [5, 6, 1, 2, 9, 4]$.

Muestre que para cualquier árbol t , se cumple $nn(t) = long(aplana(t))$.

5.4.29.- Queremos representar árboles binarios cuyos únicos nodos etiquetados son las hojas. Para eso tenemos la siguiente definición:

- Si $a \in A$, entonces $hoja(a)$ es un árbol.
- Si t_1, t_2 son árboles, entonces $mk(t_1, t_2)$ es un árbol.
- Éstos y sólo éstos son árboles.

Observe que en esta definición no existe el árbol vacío.

- a) Defina funciones recursivas nh y nni que calculen el número de hojas y el número de nodos internos de un árbol (es decir los nodos que no son hojas).
- b) Enuncie el principio de inducción estructural correspondiente y utilícelo para mostrar que:

$$nh(t) = nni(t) + 1.$$

5.4.30.- Considere nuevamente al conjunto de expresiones aritméticas parentizadas EA del ejercicio 5.3.19. Demuestre, usando las funciones del ejercicio 5.3.20, que para toda expresión x , cualquiera de sus prefijos contiene por lo menos tantos paréntesis izquierdos como derechos.

Parte III

Relaciones

Relaciones | 6

El mundo de las ciencias de la computación abunda en relaciones útiles. Por ejemplo aquellas que nos vienen de matemáticas, como son la relación existente entre un número entero y sus divisores; la relación definida por la propiedad de que dos números sean primos entre sí o la relación entre tres números, existente si el último es la suma de los dos primeros. Sin embargo existen muchas relaciones útiles que aparentemente no involucran directamente a las matemáticas. Por ejemplo podemos relacionar a los lenguajes de programación con los programas escritos en ellos; a los individuos y la información que tenemos sobre los mismos, definiendo relaciones familiares, o bien la relación existente entre dos ciudades si existe un vuelo directo entre ellas.

En este capítulo estudiaremos el concepto matemático de relación, sus propiedades y algunas de sus operaciones. En particular nos centraremos en dos clases importantes, las relaciones de equivalencia y las relaciones de orden. Para nuestra exposición nos serviremos de diversos conceptos de la teoría intuitiva de conjuntos, los cuales suponemos conocidos. El lector interesado puede revisar estos conceptos en cualquier libro de Álgebra Superior.

6.1. Producto cartesiano

Si A y B son conjuntos cualesquiera, su *producto cartesiano* –denotado por $A \times B$ – es el conjunto que consiste de todas las parejas ordenadas (a, b) , tales que $a \in A$ y $b \in B$. Es decir,

$$A \times B = \{(a, b) \mid a \in A, b \in B\}$$

Es importante observar que las parejas que son elementos de un producto cartesiano se llaman ordenadas porque el orden de sus componentes es relevante ya que, en general,

⁰Usaremos el término *relación* de manera informal, con su significado usual, hasta que lo definamos matemáticamente.

$(x, y) \neq (y, x)$. Más aún si $x, u \in A$ y $y, v \in B$ entonces $(x, y) = (u, v)$ si y sólo si $x = u$ e $y = v$. Es decir, dos parejas ordenadas son iguales si y sólo si lo son componente a componente. Por ejemplo $(18/2, 4 - 3) = (10 - 1, -1 + 2)$ pero $(3, 6 + 5) \neq (5 - 2, 7)$, pues $6 + 5 \neq 7$.

Veamos algunos ejemplos de productos cartesianos.

Ejemplo 6.1. Si $A = \{0, 5, 2, 8\}$ y $B = \{1, 3, 4, 7\}$, entonces

$$A \times B = \{(0, 1), (0, 3), (0, 4), (0, 7), (5, 1), (5, 3), (5, 4), (5, 7), \\ (2, 1), (2, 3), (2, 4), (2, 7), (8, 1), (8, 3), (8, 4), (8, 7)\}$$

Ejemplo 6.2. El producto cartesiano de los números naturales consigo mismos es:

$$A \times A = \mathbb{N} \times \mathbb{N} = \{(1, 1), (1, 2), (2, 1), (2, 2), (1, 3), \\ (2, 3), (3, 1), (3, 2), (3, 3), (1, 4), \dots \\ \}$$

En este caso, como \mathbb{N} es un conjunto infinito, el producto cartesiano también es infinito¹. Para describirlo formalmente usamos la notación por comprensión:

$$A = \mathbb{N} \times \mathbb{N} = \{(a, b) \mid a, b \in \mathbb{N}\}.$$

Ejemplo 6.3. Si pensamos en parejas formadas por los invitados a una fiesta, donde ponemos en un conjunto a los hombres y en otro a las mujeres:

$$H = \{Juan, Pedro, Mario, Arturo\} \quad M = \{Ana, Rosa, Margarita\}.$$

Las posibles parejas de hombre y mujer que pueden bailar en la fiesta corresponden al producto cartesiano $H \times M$ y son:

$$H \times M = \{(Juan, Ana), (Juan, Rosa), (Juan, Margarita), \\ (Pedro, Ana), (Pedro, Rosa), (Pedro, Margarita), \\ (Mario, Ana), (Mario, Rosa), (Mario, Margarita), \\ (Arturo, Ana), (Arturo, Rosa), (Arturo, Margarita) \\ \}$$

Insistimos en recalcar que el orden en cada pareja es particularmente importante pues el primer elemento de la pareja es de un conjunto y el segundo de otro diferente. Aun cuando los conjuntos sean iguales el orden es importante. Consideremos por ejemplo los pares

¹Tal vez una manera más “natural” habría sido listar primero las parejas que tienen como primer componente al 1, seguir con las que tienen como primer componente al 2 y así sucesivamente. Como los números naturales son infinitos no terminaríamos de listar a las primeras parejas que mencionamos. Por eso se elige un *orden canónico*, que está dado por la suma de los elementos de la pareja.

(n, m) , (m, n) del conjunto $\mathbb{N} \times \mathbb{N}$. Si pensamos que el par (n, m) representa la relación $n < m$ entonces claramente (m, n) no debe considerarse igual a (n, m) .

Podemos extender el producto cartesiano a cualquier número de conjuntos. Veamos un ejemplo para tres conjuntos. En este caso a cada elemento del producto cartesiano le llamamos *terna* en lugar de pareja.

Ejemplo 6.4. Queremos organizar una competencia entre equipos que tengan dos adultos y un niño (identificaremos a cada uno por su inicial). Tenemos el conjunto de los adultos y el conjunto de los niños:

$$A = \{M, J, P, A\}, \quad N = \{m, j, p\}.$$

Los equipos estarán formados por un adulto, que será el capitán, otro adulto, que será el suplente, y un niño; los equipos serán tomados del producto cartesiano $A \times A \times N$ —ignoraremos por el momento el problema de tener dos veces al mismo adulto—. Nuevamente, el orden en que aparecen los jugadores dentro del equipo es importante, pues de eso depende el papel que van a tomar dentro del mismo y de cuál de los conjuntos provienen. El número inicial de equipos es 48, que resulta de asociar a cada adulto con cada uno de los elementos del conjunto de adultos ($4 \times 4 = 16$ parejas) y a cada una de estas parejas con cada uno de los niños ($16 \times 3 = 48$ equipos).

$$\begin{aligned} \text{Equipos} = A \times A \times N = \{ & (M, M, m), (M, J, m), (M, P, m), (M, A, m), (J, M, m), \\ & (J, J, m), (J, P, m), (J, A, m), (P, M, m), (P, J, m), \\ & (P, P, m), (P, A, m), (A, M, m), (A, J, m), (A, P, m), \\ & (A, A, m), (M, M, j), (M, J, j), (M, P, j), (M, A, j), \\ & (J, M, j), (J, J, j), (J, P, j), (J, A, j), (P, M, j), \\ & (P, J, j), (P, P, j), (P, A, j), (A, M, j), (A, J, j), \\ & (A, P, j), (A, A, j), (M, M, p), (M, J, p), (M, P, p), \\ & (M, A, p), (J, M, p), (J, J, p), (J, P, p), (J, A, p), \\ & (P, M, p), (P, J, p), (P, P, p), (P, A, p), (A, M, p), \\ & (A, J, p), (A, P, p), (A, A, p) \\ & \} \end{aligned}$$

Hay varias ternas que no son válidas, ya que el mismo adulto aparece como capitán y suplente. Si ponemos restricciones a la manera de formar las parejas de adultos, entonces invalidamos varias parejas por lo que el conjunto de equipos válidos no es todo el producto cartesiano, sino un subconjunto de éste. Por esta clase de problemas diremos que *una re-*

lación es un subconjunto del producto cartesiano. Si usamos el criterio de que un mismo adulto sólo puede aparecer una vez en cada terna, tendremos a cada adulto formando pareja con los tres restantes ($4 \times 3 = 12$ parejas, ya que eliminamos a las cuatro parejas formadas con el mismo adulto) y a cada una de estas parejas con cada uno de los niños ($12 \times 3 = 36$ equipos), por lo que del conjunto anterior de equipos eliminamos 12 ternas, que es la cuarta parte de las originales.

Ejemplo 6.5. Supongamos ahora que tenemos un conjunto formado con las asignaturas obligatorias de la licenciatura en Matemáticas (M) y definimos una relación $S \subseteq M \times M$, donde $(m_1, m_2) \in S$ si la asignatura m_1 es prerequisite directo para la asignatura m_2 . En este contexto no es válido que tanto (m_1, m_2) como (m_2, m_1) estén en S —¿ven por qué?—. Podemos tener una asignatura m_1 relacionada con dos o más materias distintas, o a varias materias distintas relacionadas con una misma asignatura m_2 .

Por ejemplo, Álgebra Superior I ($AS1$) es prerequisite de Cálculo Diferencial e Integral II ($CDI2$) y de Álgebra Superior II ($AS2$), pero también Cálculo Diferencial e Integral I ($CDI1$) y Geometría Analítica I ($GA1$) son prerequisites para Cálculo Diferencial e Integral II. De lo anterior tenemos las siguientes parejas que pertenecen a la relación S :

$$S \supseteq \{ (AS1, AS2), (AS1, CDI2), (CDI1, CDI2), (GA1, CDI2) \}$$

Podemos generalizar el producto cartesiano a un número arbitrario de conjuntos donde los elementos serán tuplas o n -adas (*eneadas*) ordenadas donde el i -ésimo elemento pertenece al i -ésimo conjunto:

$$C = A_1 \times A_2 \times \dots \times A_n = \{ (a_1, a_2, \dots, a_n); a_i \in A_i \}$$

De los ejemplos anteriores debe quedar claro que los conjuntos A_i pueden o no ser distintos entre sí. Si $n = 2$ los elementos del producto cartesiano son parejas; con $n = 3$ los elementos son triplas (o ternas); y así sucesivamente, usando el término genérico de n -adas para los elementos del producto cartesiano de n conjuntos.

Ejercicios

6.1.1.- Sean $A = \{0, 1\}$, $B = \{1, 2\}$. Determine los siguientes conjuntos:

a) $A \times B$ y $B \times A$

d) $A \times \{1\} \times B$

b) $A \times A$ y $B \times B$.

e) $(A \times A) \times B$

c) $A \times B \times A$ y $B \times A \times B$.

f) $(B \times A) \times (B \times A)$

6.1.2.- Consideremos el producto cartesiano entre profesores que pueden impartir Álgebra I y II y los grupos que se abren para estas asignaturas en ciertos horarios. Tenemos la restricción de que un mismo profesor no puede dar dos asignaturas en el mismo

horario. Identificaremos a los profesores con una letra por su apellido, a las asignatura de manera similar a como lo hicimos en el ejemplo 6.5 y a los horarios con la hora en la que inicia la asignatura. Enseguida mostramos a los conjuntos que intervienen en el producto cartesiano:

$$P = \{M, P, N, O, R, A\}, A = \{AS1, AS2\}, H = \{9, 10, 11\}.$$

¿Cuántas y cuáles ternas válidas hay, de acuerdo a la restricción dada, en el producto cartesiano $P \times A \times H$?

6.1.3.- En relación a los mismos conjuntos que en el ejercicio anterior, tenemos un alumno que tiene que llevar Álgebra Superior I y Álgebra Superior II, pero no puede hacerlo en el mismo horario. Si $S = P \times A \times H$ entonces liste el subconjunto de $S \times S$ que es válido para este estudiante.

6.1.4.- Consideremos al conjunto de nombres

$$N = \{Juan, Pedro, Pablo, José, Javier, Jaime, Arturo, César\}.$$

¿Cuál es la cardinalidad del conjunto de parejas de nombres, $N \times N$? ¿Y del subconjunto de $N \times N$ de aquellas parejas que no repitan la primera letra del nombre?

6.1.5.- Supongamos que tenemos un conjunto E de estudiantes de Música y un conjunto I de instrumentos musicales. Los estudiantes de Música son ocho y los instrumentos son cinco. Consideremos el producto cartesiano $E \times I \times I$ entre los estudiantes de Música y los instrumentos de manera doble. La escuela de Música tiene la restricción de que cada estudiante únicamente puede aprender a tocar dos de los instrumentos y estos instrumentos tienen que ser distintos. ¿Cuál es la cardinalidad de este subconjunto de $E \times I \times I$?

6.1.6.- Sean $A = \{1, 2, 4, 8, 16\}$, $B = \{1, 2, 3, 4, 5, 6, 7\}$. Si $(2 - x, 5), (4, y - 2) \in A \times B$, ¿cuáles son las parejas para las cuales $(2 - x, 5) = (4, y - 2)$?

6.1.7.- Este ejercicio requiere recordar las definiciones de las operaciones básicas de conjuntos (unión \cup , intersección \cap , diferencia \setminus , contención o inclusión \subseteq , etcétera, las cuales pueden consultarse en un libro de Álgebra Superior). Demuestre lo siguiente.

a) Si $A \subseteq B$ entonces $A \times C \subseteq B \times C$.

b) Si $A \subseteq B$ entonces $C \times A \subseteq C \times B$.

c) Si $A \subseteq B$ y $C \subseteq D$ entonces $A \times C \subseteq B \times D$.

d) $(A \cup B) \times C = (A \times C) \cup (B \times C)$

e) $(A \cap B) \times C = (A \times C) \cap (B \times C)$

f) $(A \cap B) \times (A \cap B) = (A \times B) \cap (B \times A)$

g) $A \times (B \setminus C) = A \times B \setminus A \times C$

6.2. Relaciones

Enunciamos ahora la definición formal de relación siguiendo las ideas de los ejemplos y ejercicios anteriores, de acuerdo a los cuales una *relación* es un subconjunto de un producto cartesiano. Es decir, un conjunto de n -adas ordenadas en la que cada n -ada cumple con una cierta propiedad, que es la que define a la relación, y que posee cada uno de los miembros de la relación.

Definición 6.1 (relación) Una *relación* R sobre los conjuntos A_1, \dots, A_n es un subconjunto del producto cartesiano $A_1 \times \dots \times A_n$.

Si $(a_1, \dots, a_n) \in R$ decimos que a_1, \dots, a_n están relacionados según R . En particular, si $n = 2$ y $(a, b) \in R$ decimos que a está relacionado con b según R y, en este caso, a veces utilizamos la notación infija aRb en vez de $(a, b) \in R$.

Una manera usual de clasificar las relaciones es mediante el número de conjuntos en el producto cartesiano. Esta clasificación se define a continuación.

Definición 6.2 Decimos que una relación es *binaria* si es subconjunto del producto cartesiano de *dos* conjuntos; es *terciaria* si es subconjunto del producto cartesiano de tres conjuntos; en general, es *n -aria* si es subconjunto del producto cartesiano de n conjuntos.

El caso en que $n = 1$ no se considera aquí, pues no es usual hablar de relaciones unarias, sino de propiedades o predicados, cosa que ya hemos hecho en otros capítulos.

Ejemplo 6.6. Denotemos a $R \subseteq \mathbb{N} \times \mathbb{N}$, $R = \{(a, b) \mid a < b\}$. Esta relación consiste de todas las parejas de naturales en los que el primer elemento es menor que el segundo.

Ejemplo 6.7. Sea A un conjunto de animales. Definimos la relación $R \subseteq A \times A$ como $(a, b) \in R$ si y sólo si el animal a come al animal b . Por ejemplo, $(León, Gacela), (Lobo, Oveja), (Gato, Ratón) \in R$ pero $(Elefante, Ballena), (Mosca, Lagartija) \notin R$.

Ejemplo 6.8. El conjunto A consiste de todos aquellos individuos que tienen López como primer o segundo apellido. Consideremos $F \subseteq A \times A$, $F = \{(p, h) \mid p \text{ es padre de } h\}$. En esta relación, si $(Juan, Pedro) \in F$, entonces $(Pedro, Juan) \notin F$.

Ejemplo 6.9. Consideremos el producto cartesiano $\mathbb{N} \times \mathbb{N} \times \mathbb{N}$ y la relación $\text{suma} \subseteq \mathbb{N} \times \mathbb{N} \times \mathbb{N}$, $\text{suma} = \{(r, s, t) \mid r + s = t\}$. Algunas de las ternas que están en suma son $(1, 2, 3), (4, 2, 6), (2, 4, 6), (10, 10, 20)$. Recordemos que la segunda y tercera ternas son distintas aún cuando juntas representan el hecho de que $4 + 2 = 2 + 4$.

Ejemplo 6.10. Sean T el conjunto de continentes, P el conjunto de países y C el conjunto de ciudades del mundo, definimos la relación $R \subseteq T \times P \times C$ como $(c, p, s) \in R$ si y sólo si

la ciudad s está en el país p y p está en el continente c . Por ejemplo, $(Asia, Japón, Kioto)$, $(América, México, Aguascalientes)$, $(Europa, Alemania, Colonia)$ están en R , pero $(Oceanía, Australia, Wellington) \notin R$ y $(Europa, Francia, Verona) \notin R$.

El siguiente ejemplo será de particular importancia más adelante.

Ejemplo 6.11. Sean \mathbb{Z} el conjunto de números enteros y $a, b \in \mathbb{Z}$. Decimos que a divide a b , denotado $a \mid b$, si y sólo si existe $k \in \mathbb{Z}$ tal que $b = k \cdot a$. Definimos $R \subseteq \mathbb{Z} \times \mathbb{Z}$ como $R = \{(a, b) \mid a \mid b\}$. Por ejemplo $(5, 30)$, $(-3, 39)$ y $(12, 72) \in R$ puesto que $30 = 5 \cdot 6$, $39 = (-3) \cdot 13$ y $72 = 12 \cdot 6$. Por otra parte $(3, 8)$, $(7, 23) \notin R$ puesto que $3 \nmid 8$ y $7 \nmid 23$.

La relación R se conoce como relación de divisibilidad. Si $a \mid b$ también decimos que a es divisor de b o que b es divisible entre a .

Veamos ahora algunas operaciones que generan nuevas relaciones a partir de relaciones dadas.

6.2.1. Operaciones con relaciones

Dado que las relaciones son una clase especial de conjuntos, todas las operaciones de conjuntos tienen sentido entre relaciones. Veamos algunos ejemplos.

- **Unión de relaciones:** la unión de R y S , denotada $R \cup S$, consiste de todas las tuplas que están en R o en S sin que sea necesario que ambas sean relaciones con el mismo número de argumentos.
 - La relación ser suma o producto de dos números.
 - La relación ser menor o mayor que un número dado.
 - La relación ser el máximo o el mínimo de dos números dados.
 - La relación entre el número de cuenta de un alumno, su carrera y su promedio o bien entre el número de cuenta y su última materia inscrita.
- **Intersección de relaciones:** la intersección de R y S , denotada $R \cap S$, consiste de todas las tuplas que están en R y en S . En este caso R y S sí deben tener el mismo número de argumentos.
 - La relación ser divisible entre dos y entre tres.
 - La relación ser menor que y ser consecutivos.
- **Diferencia de relaciones:** la diferencia de R y S , denotada $R \setminus S$, consiste de todas las tuplas que están en R y que no están en S . En este caso R y S sí deben tener el mismo número de argumentos.
 - La relación ser menor o igual que y no ser igual.
 - La relación ser ciudad de un país y no ser la capital del mismo.

- **O exclusivo:** la unión exclusiva de R y S , denotada $R \oplus S$, consiste de todas las tuplas que están en R o en S pero no en ambas.
 - Si R es la relación de parejas (e, m) tal que el estudiante e ya cursó la materia m y S es la relación de parejas (e, r) tal que el estudiante e requiere cursar la materia r para graduarse entonces la unión exclusiva de R y S consiste de aquellas parejas (e, s) tal que el estudiante e ha cursado la materia s pero ésta no es necesaria para graduarse o bien el estudiante e requiere a la materia s para graduarse pero no la ha cursado aún.

Si bien las definiciones formales de estas operaciones son las dadas por las operaciones conjuntistas, las definimos aquí nuevamente para el caso de relaciones binarias.

Sean $R_1, R_2 \subseteq A \times B$. Definimos las siguientes operaciones:

Unión: $R_1 \cup R_2 = \{(x, y) \mid (x, y) \in R_1 \vee (x, y) \in R_2\}.$

Intersección: $R_1 \cap R_2 = \{(x, y) \mid (x, y) \in R_1 \wedge (x, y) \in R_2\}.$

Diferencia: $R_1 \setminus R_2 = \{(x, y) \mid (x, y) \in R_1 \wedge (x, y) \notin R_2\}.$

O exclusivo: $R_1 \oplus R_2 = \{(x, y); ((x, y) \in R_1 \cup R_2) \wedge ((x, y) \notin R_1 \cap R_2)\}.$

Ejemplo 6.12. Sean R y S las relaciones siguientes:

$$R = \{(1, 1), (1, 2), (2, 1)\}$$

$$S = \{(1, 1), (1, 2), (1, 4), (2, 1), (2, 2), (3, 3), (4, 1), (4, 4)\}$$

Calcular $R \oplus S$.

Solución: $R \oplus S = \{\cancel{(1, 1)}, \cancel{(1, 2)}, \cancel{(2, 1)}\} \oplus$
 $\oplus \{\cancel{(1, 1)}, \cancel{(1, 2)}, (1, 4), \cancel{(2, 1)}, (2, 2), (3, 3), (4, 1), (4, 4)\}$
 $= \{(1, 4), (2, 2), (3, 3), (4, 1), (4, 4)\}.$

Enseguida, debido a su gran importancia, estudiamos de manera más amplia a las relaciones binarias.

Ejercicios

6.2.1.- Una terna (x, y, z) de números enteros se dice que es *pitagórica* si cumple la propiedad: $x^2 + y^2 = z^2$. Dar los elementos de la siguiente relación:

$$R = \{(x, y, z) \mid x, y \leq 5 \text{ y } (x, y, z) \text{ es pitagórica}\}.$$

6.2.2.- Sea C el conjunto de ciudades del mundo y P el conjunto de países del mundo. Dar al menos dos ejemplos de relaciones en $C \times P$.

6.2.3.- Respecto a los conjuntos C, P del ejercicio anterior, defina alguna relación ternaria $R \subseteq P \times P \times P$ y alguna relación ternaria $S \subseteq C \times C \times C$.

6.2.4.- Sean $A = \{0, 1, 2, 3, 4\}$, $B = \{1, 3, 7, 12\}$, $C = \{0, 1, 2, 4, 8, 16, 32\}$ y $D = \{-3, -2, -1, 0, 1, 2, 3\}$.

a) Sea $R \subseteq A \times B \times C \times D$ definida como $R = \{(x, y, z, w) \mid xyzw = 0\}$. Liste todas las cuaternas que pertenecen a R .

b) Sea $R \subseteq A \times B \times C \times D$ definida como $R = \{(x, y, z, w) \mid xyzw < 0\}$. Decir cuántas son y listar todas las cuaternas que pertenecen a R .

6.2.5.- Liste las 16 relaciones binarias diferentes sobre el conjunto $\{f, t\}$. ¿En cuáles de estas relaciones binarias figura el par (f, t) ?

6.2.6.- Considere las siguientes relaciones binarias:

$$R = \{(1, 2), (2, 3), (3, 4)\},$$

$$S = \{(1, 1), (1, 2), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3), (3, 4)\}.$$

Calcular las siguientes relaciones.

a) $R \cup S$, $R \cap S$.

b) $R \setminus S$, $S \setminus R$.

c) $R \oplus S$.

6.3. Relaciones binarias

Es común trabajar con relaciones R que consisten de parejas sobre dos conjuntos A y B , es decir $R \subseteq A \times B$. Como ya mencionamos, estas son *relaciones binarias*. Listamos a continuación algunos ejemplos:

- La relación *edad* entre personas y números naturales.
- La relación *signo del Zodiaco* entre personas y signos del Zodiaco dados por su fecha de nacimiento.
- En el mundo de figuras sobre un panel, la relación *estar al norte de*, *estar al oeste de*, *estar junto a*, entre otras.
- La relación de conversión entre un natural y su representación binaria.
- La relación entre listas y su longitud.
- La relación entre árboles y la lista de sus elementos.
- En programación orientada a objetos, la relación entre dos clases si la primera extiende a la segunda.
- En programación orientada a objetos, la relación entre un método y una clase si el método está declarado en esa clase.

Una clase particular de relaciones binarias son aquellas definidas sobre un mismo conjunto A , es decir, las relaciones R tales que $R \subseteq A \times A$. Estas relaciones se presentan con mucha frecuencia y suelen ser importantes, especialmente en matemáticas. Antes de ver algunos ejemplos aclaremos el significado de ciertas operaciones entre enteros.

Definición 6.3 Sean $a, b \in \mathbb{Z}$ y $q, r \in \mathbb{N}, q \geq 0, 0 \leq r < b$. Si $a = qb + r$, definimos el cociente entero y el módulo de a con respecto a b , denotados $a \div b$ y $a \bmod b$ respectivamente como

$$a \div b = q \quad a \bmod b = r$$

Esta definición es correcta debido al llamado algoritmo de la división que garantiza la existencia de enteros únicos q, r tales que $a = bq + r$ con $0 \leq r < |b|$, el último término se refiere al valor absoluto de b .

Ejemplo 6.13. Si tomamos al producto cartesiano $\mathbb{Z} \times \mathbb{Z}$, podemos listar las siguientes relaciones:

$$R_1 = \{(a, b) \mid a, b \in \mathbb{Z}, a < b\}$$

$$R_2 = \{(a, b) \mid a, b \in \mathbb{Z}, a \leq b\}$$

$$R_3 = \{(a, b) \mid a, b \in \mathbb{Z}, a = b\}$$

$$R_4 = \{(a, b) \mid a, b \in \mathbb{Z}, (a \bmod c) = (b \bmod c)\}$$

$$R_5 = \{(a, b) \mid a, b \in \mathbb{Z}, c > 1, (a \div c) = (b \div c)\}$$

$$R_6 = \{(a, b) \mid a, b \in \mathbb{Z}, a \text{ divide a } b (a \mid b)\}$$

$$R_7 = \{(a, b) \mid a, b \in \mathbb{Z}, a \text{ no es múltiplo de } b\}$$

En los casos de R_4 y R_5 las operaciones de módulo (\bmod) y cociente entero (\div) resultan familiares como operadores en lenguajes de programación. Para evitar confusiones es que dimos su definición formal.

Daremos algunos ejemplos de parejas que se encuentran en las relaciones que acabamos de anotar; sin embargo, como todas estas relaciones son infinitas, no podremos más que dar una pequeña lista, terminando la descripción con puntos sucesivos indicando que siguen un número infinito de parejas que están en la relación definida.

$$R_1 = \{ \dots (-1, 0), (-2, -1), (-2, 0), (1, 2), (1, 3), (1, 4), \dots, (2, 3), (2, 4), \dots \}$$

$$R_2 = R_1 \cup \{(1, 1), (2, 2), (3, 3), \dots\}$$

$$R_3 = \{(1, 1), (2, 2), (3, 3), \dots\}$$

Si consideramos, por ejemplo, $c = 5$, entonces

$$R_4 = \{(5, 25), (7, 12), (2, 7), (127, 20422), \dots\}$$

Si ahora consideramos $c = 3$, entonces

$$R_5 = \{(15, 15), (15, 16), (15, 17), (9, 9), (9, 10), (9, 11), \dots\}$$

$$R_6 = \{(2, 2), (2, 4), (2, 6), \dots, (3, 3), (3, 6), (3, 9), \dots\}$$

$$R_7 = \{(4, 3), (7, 11), \dots, (8, 5), \dots (35, 23), \dots\}$$

En esta ocasión utilizamos para denotar a cada relación (de cardinalidad infinita) sub-listas de los elementos que pertenecen a la misma. Es una pregunta común en exámenes sicométricos para calcular el coeficiente intelectual de una persona, dar sucesiones de este tipo y preguntar cuál es el número que sigue, obligándonos a recuperar las relaciones especificadas en la primera parte de este ejemplo.

Es importante mencionar dos relaciones binarias especiales, la relación *identidad* y la relación *universal* definidas sobre A , un conjunto cualquiera:

Definición 6.4 (relación identidad) La *relación identidad* sobre un conjunto A , denotada por I_A , es la relación de igualdad definida como $I_A = \{(a, b) \mid a, b \in A, a = b\}$, es decir, todo elemento está relacionado únicamente consigo mismo.

Definición 6.5 (relación universal) La *relación universal* sobre un conjunto A , denotada por U_A , es la relación definida como $U_A = \{(a, b) \mid a, b \in A\}$. Esto es, cada elemento está relacionado con todos los demás, en particular consigo mismo. Obsérvese que $U_A = A \times A$.

Enseguida presentamos distintas maneras de representar relaciones binarias, las cuales pueden resultar más convenientes, en algunos casos, que la definición mediante conjuntos de parejas.

6.3.1. Representación de relaciones binarias

Una relación binaria se puede representar, como lo hemos hecho hasta ahora, con la notación usual de conjuntos, ya sea listando todos sus elementos —esto es posible sólo en los conjuntos finitos—, indicando con puntos sucesivos a un conjunto infinito cuyos elementos siguen el patrón dado o describiendo la propiedad que deben cumplir; sin embargo, esta notación es poco operativa cuando queremos manipular de alguna manera a la relación. Tenemos otras opciones para el caso en que los conjuntos en los que se define la relación sean finitos, digamos $R \subseteq A_1 \times A_2$. La primera de ellas es una tabla con un renglón para cada elemento de A_1 y una columna para cada elemento de A_2 . Se marcan aquellos elementos que estén en la relación (con una cruz, un 1 o con el valor booleano *verdadero*). En el caso de conjuntos infinitos, la tabla se puede dejar indicada listando únicamente algunos de sus elementos.

Ejemplo 6.14. Usemos los conjuntos A y N del ejemplo 6.4. Queremos formar las parejas posibles donde el primer elemento de la pareja es un adulto y el segundo un niño. La relación está definida por aquellas parejas cuyos elementos tengan distinto nombre. Esta representación se muestra en la tabla 6.1, a continuación.

Tabla 6.1. Un adulto y un niño cuyos nombres empiezan distinto

Adultos	Niños		
	m	j	p
M		×	×
J	×		×
P	×	×	
A	×	×	×

Ejemplo 6.15. Si regresamos a la relación R_3 del ejemplo 6.13, la relación quedaría indicada como se muestra en la tabla 6.2.

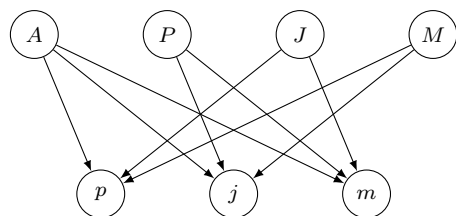
Tabla 6.2. Relación identidad sobre los números naturales

$\mathbb{N} \times \mathbb{N}$	1	2	3	4	5	...
1	×					
2		×				
3			×			
4				×		
5					×	
⋮						

Otra forma de representar relaciones es mediante una gráfica dirigida. Una *gráfica dirigida* o *digráfica* $G = (V, E)$ consta de un conjunto V cuyos elementos se llaman vértices, nodos o puntos, y una relación binaria $A \subseteq V \times V$, llamada la relación de adyacencia, tal que $(x, y) \in E$ indica que el primer vértice x es adyacente al segundo vértice y . Una gráfica se muestra visualmente mediante círculos o puntos representando a los elementos de V y si $(v_1, v_2) \in E$ dibujamos una flecha (*arco*) que sale de v_1 y llega a v_2 –la forma particular de la flecha o arco no importa, excepto que tiene que mostrar conexión y dirección–. En el caso de una relación binaria $R \subseteq A \times B$, el conjunto de vértices es $V = A \cup B$ –cada elemento de cada uno de los conjuntos corresponde a un vértice de la gráfica– y cada arco en E , corresponde a una pareja $(a, b) \in R$. Cuando la relación tiene alguna pareja (a, a) , se dibuja un arco que sale y llega al mismo vértice, a lo que llamamos un *lazo*.

Ejemplo 6.16. La gráfica correspondiente al ejemplo 6.14 se muestra en la figura 6.1, a continuación.

Figura 6.1. Gráfica correspondiente a la relación del ejemplo 6.14



Adicionalmente, parecido a las tablas pero más manejables matemáticamente hablando, podemos representar a una relación con una matriz booleana: en cada posición (i, j) de la matriz se pone un 1 si los elementos a_i y a_j están en la relación y 0 si no. Al igual que en las tablas, los conjuntos infinitos pueden quedar indicados. Es importante mencionar que si el producto cartesiano corresponde a una relación binaria sobre el mismo conjunto, o si ambos conjuntos tienen el mismo número de elementos, estas matrices son cuadradas.

La matriz definida para I_A es la que tiene 1 en la diagonal y 0 en el resto de las posiciones, mientras que la gráfica dirigida correspondiente a esta relación tiene tantos vértices como elementos hay en el conjunto y cada vértice tiene un lazo, pero no hay arcos que no sean lazos.

La matriz definida para U_A tiene un 1 en todas las posiciones, mientras que la gráfica tiene un arco entre cualesquiera dos vértices, además de un lazo para cada vértice.

Ejemplo 6.17. La matriz booleana correspondiente al ejemplo 6.14 se encuentra a continuación. Corresponde a la gráfica de la figura 6.1.

$$\begin{array}{c} \\ M \\ J \\ P \\ A \end{array} \begin{pmatrix} & m & j & p \\ \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \end{pmatrix}$$

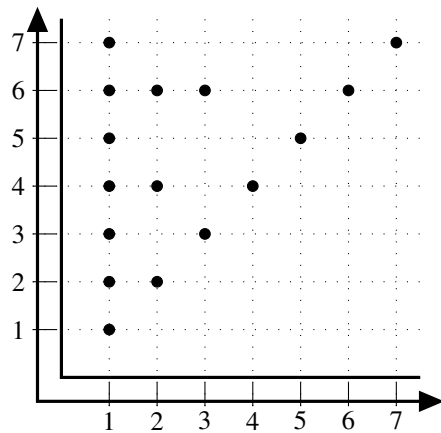
Diagramas de coordenadas

Cuando tenemos relaciones binarias entre conjuntos de números, particularmente \mathbb{N} o \mathbb{Z}^+ , podemos representar la relación como puntos en un plano, donde los ejes del plano están etiquetados a distancias constantes por los elementos del conjunto listados en cierto orden, usualmente en orden creciente.

Ejemplo 6.18. Supongamos tenemos el conjunto $A = \{1, 2, 3, 4, 5, 6, 7\}$ y la relación de divisibilidad $|$ sobre $A \times A$. Podemos representar esta relación en un plano cartesiano cuyos ejes están etiquetados con los enteros $1, 2, \dots, 7$ y los elementos que están relacionados

entre sí se representan con un punto en las coordenadas (a, b) , como podemos observar en la figura 6.2.

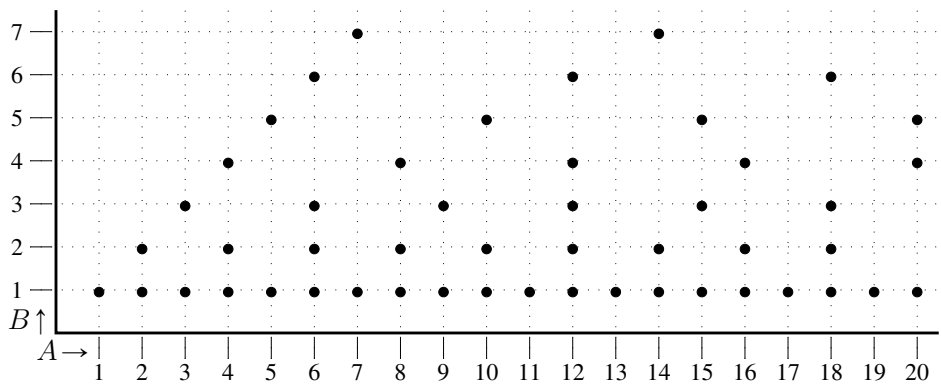
Figura 6.2. Diagrama de coordenadas para la relación en el ejemplo 6.18



Este tipo de diagramas se puede utilizar para cualquier clase de relación, siempre que se elija un orden fijo para los elementos de cada uno de los conjuntos de la misma, para que uno se coloque en el eje horizontal y el otro en el eje vertical.

Ejemplo 6.19. Supongamos que tenemos los conjuntos $A = \{1, 2, 3, 4, 5, 6, \dots, 19, 20\}$, $B = \{1, 2, 3, 4, 5, 6, 7\}$ y una relación R donde $R = \{(a, b) \mid a \text{ es múltiplo de } b\}$. El diagrama de coordenadas se encuentra en la figura 6.3.

Figura 6.3. Relación a es múltiplo de b



En el caso de relaciones binarias sobre conjuntos infinitos, como la relación identidad entre los naturales, los ejes se extienden con puntos suspensivos a partir de cierto número. Sin embargo, en relaciones arbitrarias los puntos no siempre tienen un patrón a seguir, por lo que este tipo de diagramas suelen ser poco útiles.

6.3.2. Operaciones con relaciones binarias

En las relaciones binarias, además de las operaciones heredadas de conjuntos, contamos con otras operaciones importantes discutidas a continuación. Empecemos con unos ejemplos

- A partir de la relación *ser padre de* podemos definir fácilmente la relación *ser hijo de* simplemente observando que a es padre de b si y sólo si b es hijo de a .
- La relación *comerse a* entre animales permite definir la relación *ser comido por* de la misma manera.
- La relación *ser abuelo de* entre personas se define a partir de la relación *ser padre de* observando que a es abuelo de c si y sólo si existe un b (el padre) tal que a es padre de b y b es padre de c .
- La relación *haber vuelo* entre ciudades se define similarmente a partir de la relación *haber vuelo directo*.

Describiremos estas ideas de forma general mediante operaciones formales.

Inversa: La inversa de una relación R sobre $A \times B$ se denota con R^{-1} y es una relación definida sobre $B \times A$:

$$R^{-1} = \{(b, a) \mid (a, b) \in R\}.$$

Composición: Dadas dos relaciones R sobre $A \times B$ y S sobre $B \times C$, definimos la composición de R con S , denotada $R \circ S$ de la siguiente manera:

$$R \circ S = \{(x, y) \mid \text{existe } z \in B, (x, z) \in R \wedge (z, y) \in S\}$$

Potencia: Dada una relación R sobre $A \times A$ y un número natural $n \in \mathbb{N}$, definimos la potencia n -ésima de la relación R , denotada R^n , como la relación obtenida al componer R consigo misma n -veces. Recursivamente las potencias de R se definen como:

$$R^0 = \emptyset$$

$$R^1 = R$$

$$R^{n+1} = R^n \circ R$$

Obsérvese que para que la composición consigo misma sea posible, si R es una relación sobre $A \times B$, es necesario $A = B$. Es decir, para que R admita la operación *potencia*, debe ser una relación sobre $A \times A$.

Es importante notar que en todas estas operaciones nunca se afectan las relaciones originales sino que se produce una nueva relación, un nuevo conjunto de parejas, que puede o no coincidir con alguna de las relaciones originales. A este tipo de aplicación de operadores se le conoce como *funcional*. Veamos algunos ejemplos con estas operaciones.

Ejemplo 6.20. Calculamos la relación inversa de la relación dada.

- Si $R = \{(a, b), (c, d), (e, f), (g, h)\}$ entonces $R^{-1} = \{(b, a), (d, c), (f, e), (h, g)\}$.
 - Si R es la relación *ser hijo de* sobre el conjunto de seres humanos entonces R^{-1} es la relación *ser padre de* puesto que si a es hijo de b entonces claramente b es padre de a .
 - Si R es la relación \leq sobre los enteros \mathbb{Z} entonces R^{-1} es la relación \geq puesto que si $n \leq m$ entonces $m \geq n$.
-

En el caso de la relación de divisibilidad, la inversa es la relación de multiplicidad. Veamos los detalles.

Ejemplo 6.21. Sea R la relación de divisibilidad, denotada por el símbolo $|$ y definida sobre $\mathbb{Z} \times \mathbb{Z}$. Se cumple $(a, b) \in R^{-1}$ si y sólo si $(b, a) \in R$, lo que sucede si y sólo si $b | a$; esto es, si y sólo si existe $k \in \mathbb{Z}$ tal que $a = k \cdot b$, lo que define el que a es múltiplo de b . Por lo tanto R^{-1} es la relación *ser múltiplo de*:

$$R^{-1} = \{(b, a) \mid a = k \cdot b, \quad k \in \mathbb{Z}\}.$$

Mostramos ahora algunos ejemplos de la operación de composición.

Ejemplo 6.22. Calcular $R = R_1 \circ R_2$ con $R_1 = \{(1, 1), (1, 4), (2, 3), (3, 1), (3, 4)\}$ y $R_2 = \{(1, 0), (2, 0), (3, 1), (3, 2), (4, 1)\}$.

Solución: Como dijimos antes, se trata de construir una nueva relación con aquellas parejas en las que coincida el segundo elemento de las parejas en R_1 con el primero en las parejas de R_2 :

- Se tiene $(1, 1) \in R_1, (1, 0) \in R_2$, por lo que la pareja $(1, 0) \in R_1 \circ R_2$. No hay ninguna otra pareja que tenga a 1 como primer elemento en R_2 .
- La pareja $(1, 4) \in R_1$ junto con la pareja $(4, 1) \in R_2$ agregan a R la pareja $(1, 1)$.
- La pareja $(2, 3) \in R_1$ con las parejas $(3, 1)$ y $(3, 2) \in R_2$ agregan a R las parejas $(2, 1)$ y $(2, 2)$.
- La pareja $(3, 1) \in R_1$ con la pareja $(1, 0) \in R_2$ agrega a R la pareja $(3, 0)$.
- La pareja $(3, 4) \in R_1$ con la pareja $(4, 1) \in R_2$ agregan a R la pareja $(3, 1)$.
- Ya no hay mas parejas que agregar pues ya agotamos a R_1 .

De esto, $R = R_1 \circ R_2 = \{(1, 0), (1, 1), (2, 1), (2, 2), (3, 0), (3, 1)\}$.

Ejemplo 6.23. Sea $R = \{(1, 1), (2, 1), (3, 2), (4, 3)\}$. Encontrar R^2, R^3, \dots

Solución:

$$R^2 = R \circ R = \{(1, 1), (2, 1), (3, 1), (4, 2)\}.$$

$$R^3 = R^2 \circ R = \{(1, 1), (2, 1), (3, 1), (4, 1)\}.$$

$$R^4 = R^3 \circ R = \{(1, 1), (2, 1), (3, 1), (4, 1)\}.$$

Noten que en este punto, $R^3 = R^4$. A esto es a lo que llamamos un *punto fijo*; ya no tiene caso calcular R^5, R^6, \dots , pues si $R^i = R^j$ con $i \geq 3, j > i$, vamos a obtener el mismo resultado. Por ejemplo $R^5 = R^4 \circ R = R^3 \circ R = R^3$.

Ejemplo 6.24. Sean A el conjunto de ciudades del mundo y R la relación sobre A tal que $(x, y) \in R$ si y sólo si hay un vuelo directo de la ciudad x a la ciudad y . Entonces, la relación S definida como $(x, y) \in S$ si y sólo si hay forma de volar de x a y , no es más que la unión de las potencias de R , es decir $S = \bigcup_{i=1}^{\infty} R^i$. Esto es intuitivamente claro puesto que hay forma de volar de x a y si y sólo si hay vuelos directos entre $n-1$ ciudades conectando a x con y , lo que se conoce como un vuelo con $n-1$ escalas. $(x, y) \in R^n$ significa precisamente que hay un vuelo con $n-1$ escalas de x a y . Por ejemplo, si $(\text{México}, \text{París}), (\text{París}, \text{Munich}) \in R$ entonces $(\text{México}, \text{Munich}) \in R^2$, ya que se hace una escala en París.

6.3.3. Propiedades de las relaciones binarias

Existen diversas propiedades particulares de una relación binaria que permiten clasificarlas o utilizarlas de manera más eficiente. Por ejemplo aquellas donde un individuo está relacionado consigo mismo o aquellas donde el sentido de la relación es irrelevante. Entre estas últimas tenemos que si $(x, y) \in H$, siendo H la relación de ser hermanos, entonces también $(y, x) \in H$. Discutimos aquí algunas de estas propiedades.

Para ir verificando las definiciones nos servimos de la siguiente serie de relaciones entre enteros.

$$R_1 = \{(1, 1), (1, 2), (2, 1), (2, 2), (3, 4), (4, 1), (4, 4)\}.$$

$$R_2 = \{(1, 1), (1, 2), (2, 1)\}.$$

$$R_3 = \{(1, 1), (1, 2), (1, 4), (2, 1), (2, 2), (3, 3), (4, 1), (4, 4)\}.$$

$$R_4 = \{(2, 1), (3, 1), (3, 2), (4, 1), (4, 2), (4, 3)\}.$$

$$R_5 = \{(1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 3), (2, 4), (3, 3), (3, 4), (4, 4)\}.$$

$$R_6 = \{(3, 4)\}.$$

Al mismo tiempo de enunciar las definiciones examinaremos cada una de estas relaciones para ver cuáles de ellas cumplen con la propiedad recién definida.

Reflexividad: Una relación $R \subseteq A \times A$ es *reflexiva* si cada elemento de A está relacionado consigo mismo, es decir, si se cumple que $\forall a \in A$, sucede que $(a, a) \in R$.

- R_1 no es reflexiva. $A = \{1, 2, 3, 4\}$ y la pareja $(3, 3) \notin R_1$, aunque $3 \in A$.
- Para R_2 tenemos $A = \{1, 2\}$, pero $(2, 2) \notin R_2$, por lo que tampoco es reflexiva.
- R_3 se define en $A = \{1, 2, 3, 4\}$ y cumple con ser reflexiva ya que las parejas $(1, 1), (2, 2), (3, 3)$ y $(4, 4)$ están en R_3 .

- R_4 se define en $A = \{1, 2, 3, 4\}$ pero no es reflexiva porque no contiene a ninguna pareja (a, a) .
- R_5 está definida en $A = \{1, 2, 3, 4\}$ y cumple con ser reflexiva.
- R_6 está definida en $A = \{3, 4\}$ y no es reflexiva.

Antirreflexividad: Una relación $R \subseteq A \times A$ es *antirreflexiva* si ningún elemento $a \in A$ está relacionado consigo mismo, es decir, si $\forall a \in A$ sucede que $(a, a) \notin R$.

- R_1, R_2, R_3 y R_5 no son antirreflexivas pues contienen al menos a la pareja $(1, 1)$.
- R_4 y R_6 son antirreflexivas porque ninguna contiene parejas donde el primer elemento sea igual al segundo.

Simetría: Una relación $R \subseteq A \times A$ es *simétrica* si $\forall a, b \in A$ sucede que si $(a, b) \in R$ entonces $(b, a) \in R$. Es decir, siempre que a está relacionado con b , también b está relacionado con a .

- R_1 no es simétrica pues está la pareja $(3, 4)$ pero no la $(4, 3)$.
- R_2 sí es simétrica, pues tenemos la pareja $(2, 1)$ y $(1, 2)$. La pareja $(1, 1)$ resulta igual al voltear sus elementos por lo que también cumple con la propiedad.
- R_3 sí es simétrica pues para cada pareja (a, b) se encuentra en la relación la pareja (b, a) — $(1, 4)$ y $(4, 1)$; $(1, 2)$ y $(2, 1)$ y las parejas de la forma (a, a) —.
- R_4 no es simétrica. Es más, para ninguna pareja (a, b) , tenemos a (b, a) en la relación.
- R_5 tampoco es simétrica; por ejemplo $(2, 4) \in R_5$ pero $(4, 2) \notin R_5$.
- R_6 no es simétrica pues $(3, 4) \in R_6$ pero $(4, 3) \notin R_6$.

Antisimetría: Una relación $R \subseteq A \times A$ es *antisimétrica* si $\forall a, b \in A$ se tiene que $((a, b) \in R \wedge (b, a) \in R) \rightarrow (a = b)$.

Es decir, siempre que a está relacionado con b y también b está relacionado con a , entonces $a = b$.

- R_1 no es antisimétrica, pues se cumple que $(1, 2) \in R_1$ y $(2, 1) \in R_1$ pero $1 \neq 2$.
- R_2 tampoco es antisimétrica por la misma razón que R_1 .
- R_3 tampoco es antisimétrica por la misma razón que R_1 .
- R_4 es antisimétrica *por vacuidad*. Como no sucede que tanto $(a, b) \in R$ como $(b, a) \in R$, entonces el antecedente de la implicación que define a la antisimetría es falso y por lo tanto dicha implicación es verdadera. Entonces R_4 es antisimétrica.
- R_5 es antisimétrica porque, en efecto, con a y b tales que $(a, b) \in R$ y $(b, a) \in R$ es porque $a = b$.
- R_6 es antisimétrica por la misma razón que R_4 , por vacuidad.

Asimetría: Una relación $R \subseteq A \times A$ es *asimétrica* si $\forall a, b \in A$ tenemos

$$((a, b) \in R) \rightarrow ((b, a) \notin R).$$

Es decir, si a está relacionado con b entonces no sucede que b esté relacionado con a .

- R_1, R_2, R_3 y R_5 no son asimétricas ya que contienen parejas (a, a) que invalidan la propiedad de asimetría.
- R_4 es asimétrica ya que no contiene a ninguna pareja (b, a) , donde $(a, b) \in R_4$.
- R_6 es asimétrica, ya que contiene a la pareja $(3, 4)$ pero no a la pareja $(4, 3)$.

Transitividad: Una relación $R \subseteq A \times A$ es *transitiva* si

$$\forall a, b, c \in A, ((a, b) \in R \wedge (b, c) \in R) \rightarrow (a, c) \in R.$$

Es decir, si siempre que a está relacionado con b y además b está relacionado con c , entonces a está relacionado con c .

Esta propiedad nos recuerda a la operación de composición de relaciones sólo que aquella es una operación que crea una nueva relación mientras que ésta es una propiedad de una relación dada.

- R_1 no es transitiva. Tenemos $(3, 4) \in R_1$ y $(4, 1) \in R_1$ pero $(3, 1) \notin R_1$, por ejemplo.
- R_2 no es transitiva, pues si tomamos a las parejas $(2, 1) \in R_2$ y $(1, 2) \in R_2$, la pareja $(2, 2)$ que resulta de la transitividad, no está en la relación.
- R_3 es transitiva, lo cual puede verificarse por inspección directa. Por ejemplo: $(1, 4) \in R_3, (4, 1) \in R_3$ y $(1, 1) \in R_3$;
- R_4 es transitiva. Dejamos como ejercicio verificar por inspección la propiedad.
- R_5 es transitiva. Dejamos como ejercicio corroborar esta afirmación.
- R_6 es transitiva, dado que sólo hay una pareja la propiedad se cumple por que el antecedente de la definición es falso.

Algunas de las propiedades vistas también se pueden determinar observando su representación como gráfica dirigida o matriz booleana. Veamos esto mediante ejemplo.

En el caso de la gráfica dirigida, para que la relación sea reflexiva cada nodo (elemento del conjunto) debe tener un lazo.

Ejemplo 6.25. Dada la relación

$$R_3 = \{(1, 1), (1, 2), (1, 4), (2, 1), (2, 2), (3, 3), (4, 1), (4, 4)\},$$

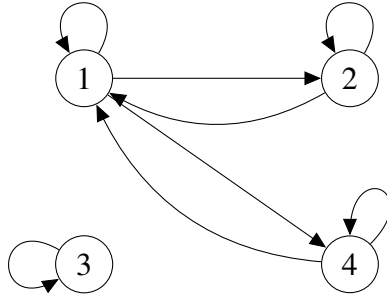
la gráfica dirigida queda como se muestra en la figura 6.4(a) de la siguiente página. Dado que todos los elementos tienen lazo, sabemos que esta relación es reflexiva.

Para el caso de la matriz booleana, la relación es reflexiva si tiene unos en la diagonal. La matriz correspondiente a la relación anterior se encuentra en la figura 6.4(b), también en la siguiente página.

Como esta matriz tiene, en efecto, unos en la diagonal, la relación es reflexiva, como ya vimos con la gráfica en la misma figura.

Figura 6.4. Representación de R_3 en el ejemplo 6.25

(a) Gráfica dirigida

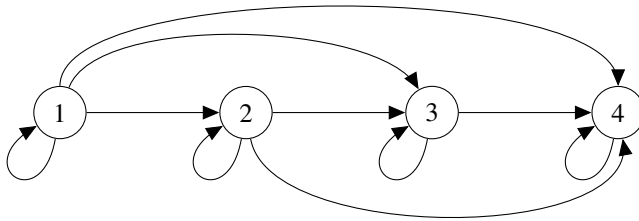


(b) Matriz booleana

$$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Para que una relación sea transitiva, siempre que haya un arco de r_i a r_j y de r_j a r_k , debe haber un arco de r_i a r_k . En cuanto a la matriz de adyacencia, siempre que haya un 1 en el elemento a_{ij} y en el elemento a_{jk} , debe haber un 1 en el elemento a_{ik} .

Ejemplo 6.26. Sea $R = \{(1,1), (1,2), (1,3), (1,4), (2,2), (2,3), (2,4), (3,3), (3,4), (4,4)\}$. La gráfica y la matriz de adyacencias correspondientes a esta relación se muestran enseguida.



$$\begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix}$$

Como ya se mencionó, existe una relación entre la propiedad de transitividad de una relación y la operación de composición, la cual se hace explícita en lo que sigue.

Proposición 6.1 Una relación R es transitiva si y sólo si contiene a todas sus potencias. Es decir, R es transitiva si y sólo si $R^n \subseteq R$ para toda $n \in \mathbb{N}$.

Demostración.

Siempre sucede que $R^0 \subseteq R$ y que $R^1 \subseteq R$, así que en adelante suponemos $n \geq 2$.

\Rightarrow : Supongamos que R es transitiva. Queremos demostrar que $R^n \subseteq R$ para $n \geq 2$.

Esto se hará por inducción sobre n .

Base $P(2)$:

Sea $(a, c) \in R^2$. Esto significa que existe $b \in A$ tal que $(a, b) \in R$ y $(b, c) \in R$, pero por la transitividad se tiene que $(a, c) \in R$. Por lo tanto $R^2 \subseteq R$.

Hipótesis de inducción: Supongamos que $R^n \subseteq R$.

Paso Inductivo: Sea $(a, c) \in R^{n+1}$. Entonces, por la definición recursiva de R^{n+1} , existe $b \in A$ tal que $(a, b) \in R^n$ y $(b, c) \in R$. Por la hipótesis de inducción $(a, b) \in R$; como además $(b, c) \in R$ entonces por la transitividad en R , $(a, c) \in R$. Luego entonces, $R^{n+1} \subseteq R$.

\therefore Si R es transitiva, entonces $\forall n \in \mathbb{N} (R^n \subseteq R)$

\Leftarrow : Supongamos ahora que $\forall n \geq 2, R^n \subseteq R$. Por demostrar que R es transitiva.

Sean $(a, b) \in R$ y $(b, c) \in R$; entonces $(a, c) \in R^2$ y como $R^2 \subseteq R$ entonces $(a, c) \in R$, lo cual implica que R es transitiva. \square

6.3.4. Operaciones de cerradura

Muchas veces nos interesa estudiar relaciones que cumplan una propiedad específica, como la simetría o la transitividad, pero que incluyan a una relación particular R , la cual puede no cumplir dicha propiedad. En esta sección estudiamos cómo es que en ciertos casos una relación arbitraria R puede extenderse a una relación que cumpla cierta propiedad.

Definición 6.6 (cerradura) Sean R una relación binaria sobre un conjunto A y \mathcal{P} una propiedad de relaciones, tal que R puede satisfacer \mathcal{P} agregando parejas². Definimos la cerradura de R con respecto a \mathcal{P} como la relación binaria más pequeña que contiene a R y además cumple \mathcal{P} . Es decir, la relación S es la cerradura de la relación R con respecto a \mathcal{P} si y sólo si se cumplen las siguientes tres condiciones:

1. $R \subseteq S$.
2. S cumple la propiedad \mathcal{P} .
3. Si Q es una relación que cumple 1 y 2 entonces $S \subseteq Q$.

La tercera propiedad nos dice que la cerradura de R debe ser la mínima relación, por lo que si cualquier otra cumple con las dos primeras condiciones, no puede estar contenida propiamente en S .

Veamos unos ejemplos sencillos tomando como base al conjunto $A = \{a, b, c\}$ y como relación a $R = \{(a, a), (a, b), (b, a), (b, c)\}$. Observemos que R no es reflexiva, simétrica, ni transitiva. Sin embargo, si agregamos pares a R podemos lograr que se cumplan estas propiedades, como vemos a continuación.

- Si agregamos los pares (b, b) y (c, c) a la relación R obtenemos una relación

$$S = R \cup \{(b, b), (c, c)\},$$

que es reflexiva.

²Por ejemplo, \mathcal{P} puede ser la propiedad de ser reflexiva, simétrica o transitiva, agregando las parejas necesarias para cumplir con estas propiedades, pero \mathcal{P} no podría ser la propiedad de ser antisimétrica o antirreflexiva, ya que para que una relación que no lo es obtenga la propiedad de antisimetría se deben eliminar parejas.

- Para obtener una relación simétrica S a partir de R basta agregar el par (c, b) .
- Si agregamos a R los pares (a, c) y (b, b) obtenemos una relación transitiva S .

En todos los casos obtuvimos la nueva relación S agregando a R los pares necesarios para lograr que se cumpla la propiedad deseada, por lo que claramente se cumplen las propiedades 1 y 2 de la definición 6.6. Más aún, el punto 3 se cumple pues se agregó el mínimo necesario de pares para lograr que la propiedad sea válida. Por ejemplo si suponemos que Q es una relación que contiene a R y es transitiva entonces necesariamente tiene a (a, c) y (b, b) como elementos y por lo tanto también se cumple que $S \subseteq Q$. Es decir, S es en realidad la cerradura de R con respecto a la propiedad de ser transitiva, también llamada de manera más simple *cerradura transitiva*. En los otros dos casos también se obtuvo una cerradura, a saber la *cerradura reflexiva* y la *cerradura simétrica*.

Enseguida mostramos el procedimiento general para construir las cerraduras reflexiva y simétrica.

Proposición 6.2 Sean $R \subseteq A \times A$ una relación binaria. Entonces:

- La cerradura reflexiva de R , denotada $refl(R)$ se construye como

$$refl(R) = R \cup I_A$$

donde I_A es la relación identidad sobre A , es decir, $I_A = \{(x, x) \mid x \in A\}$.

- La cerradura simétrica de R , denotada $sim(R)$ se construye como

$$sim(R) = R \cup R^{-1}$$

donde recordemos que R^{-1} es la relación inversa de R , es decir,

$$R^{-1} = \{(x, y) \mid (y, x) \in R\}.$$

Demostración.

En cada caso debemos demostrar la igualdad de conjuntos, lo cual se muestra por doble contención ($A \subseteq B \wedge B \subseteq Q \rightarrow A = B$).

- **$refl(R) \subseteq R \cup I_A$:**

Para esto, si utilizamos el punto 3 de la definición 6.6 con $Q =_{def} R \cup I_A$ y $S = refl(R)$, basta mostrar entonces los puntos 1 y 2 respecto a Q y S , como dice el punto 3 de esta definición.

Claramente se cumple que $R \subseteq R \cup I_A$ por lo que el punto 1 es válido. Además, como $I_A \subseteq Q$, entonces para toda $x \in A$ (se cumple la propiedad \mathcal{P} en Q) y la condición 2 se cumple. Finalmente, por el punto 3 podemos concluir que $refl(R) \subseteq Q$.

- **$R \cup I_A \subseteq refl(R)$:**

Ahora mostramos que si $Q =_{def} R \cup I_A$, entonces $Q \subseteq refl(R)$. Sea $(x, y) \in Q$. Tenemos dos casos: Si $(x, y) \in R$ entonces $(x, y) \in refl(R)$ puesto que $R \subseteq refl(R)$. Por otro lado, en caso de que $(x, y) \in I_A$ entonces $x = y$ y como $refl(R)$ es reflexiva se cumple que $(x, y) \in refl(R)$.

- **$sim(R) = R \cup R^{-1}$:**

Se demuestra de manera similar al caso anterior y se deja como ejercicio. □

Construir la cerradura transitiva de una relación es un proceso más complicado que involucra la operación de composición. Específicamente, necesitamos de las potencias de una relación.

Proposición 6.3 Sea $R \subseteq A \times A$ una relación binaria. La cerradura transitiva de R , denotada $\text{trans}(R)$ o bien R^+ , se construye como sigue:

$$R^+ = \bigcup_{n=1}^{\infty} R^n$$

Es decir, R^+ es la unión de todas las potencias de R .

Demostración.

Sea $Q = \bigcup_{n=1}^{\infty} R^n$. Mostremos primero que $R^+ \subseteq Q$. Si utilizamos nuevamente el punto 3 de la definición 6.6, basta mostrar entonces los puntos 1 y 2 para Q .

1. $R \subseteq Q$. Esto es claro pues $R^1 \subseteq Q$ y $R = R^1$.
2. Q es transitiva. Sean $(a, b), (b, c) \in Q$ y queremos mostrar que $(a, c) \in Q$. Como $(a, b) \in Q$ entonces $(a, b) \in R^n$ para alguna $n \in \mathbb{N}$; similarmente hay una $m \in \mathbb{N}$ tal que $(b, c) \in R^m$; pero entonces $(a, c) \in R^n \circ R^m$. Ahora bien, es fácil mostrar que $R^n \circ R^m = R^{n+m}$, por lo que $(a, c) \in R^{n+m}$, de donde $(a, c) \in Q$.
3. Para mostrar que $Q \subseteq R^+$, basta mostrar que $\forall n \in \mathbb{N}, R^n \subseteq R^+$, lo cual es un ejercicio de inducción que se deja al lector. □

Un gran problema en el proceso de construcción de la cerradura transitiva, desde el punto de vista computacional, es que involucra un proceso infinito, a saber el cálculo de todas las potencias de una relación. Por esta razón el proceso de construcción es inviable (*intratable*) aun cuando es fácilmente implementable. Sin embargo, en muchos casos de interés, la relación R es finita y entonces el proceso de construcción no requiere del cálculo de todas las potencias, ya que la composición de las mismas alcanza un *punto fijo*, como se asevera a continuación.

Proposición 6.4 Si A es finito, digamos $A = \{a_1, \dots, a_k\}$ y $R \subseteq A \times A$, entonces

$$R^+ = \bigcup_{i=1}^k R^i.$$

Demostración.

Damos un esbozo que el lector debe formalizar a detalle. Si existen k elementos en A , entonces en una cadena de pares de R de la forma

$$(a_1, a_2), (a_2, a_3), \dots, (a_{n-2}, a_{n-1}), (a_{n-1}, a_n)$$

donde $n > k$, necesariamente hay elementos repetidos, pues únicamente puede haber $k - 1$ pares distintos de esta forma. Pero obsérvese que tal sucesión de pares implica que $(a_1, a_n) \in R^{n-1}$. De aquí se sigue que si $n \geq k$ entonces $R^n = R^k$ y por lo tanto, usando la proposición 6.3 tenemos

$$R^+ = \bigcup_{i=1}^{\infty} R^i = \bigcup_{i=1}^k R^i$$

□

Enseguida discutimos un algoritmo útil para construir la cerradura transitiva.

Algoritmo de Warshall

Un algoritmo simple, no muy eficiente pero tratable, que calcula R^+ , cuando R es una relación finita, se debe a Stephen Warshall y fue publicado en 1962. Es conocido como el *algoritmo de Warshall para cerradura* y lo describimos en lo que sigue.

Dada una relación binaria R sobre un conjunto finito A , cuya cardinalidad es n , el algoritmo de Warshall calcula sucesivamente la potencia R^ℓ a partir de la relación R y la potencia $R^{\ell-1}$. Empezamos representando a R con una matriz, a la que identificaremos con M , y a cada elemento de la matriz lo identificaremos con m_{ij} , $1 \leq i \leq n$, $1 \leq j \leq n$. Este elemento tendrá un valor de verdadero (representado con 1) si el par $(a_i, a_j) \in R$ y falso (0) en el otro caso. Por lo tanto, M es la matriz que representa a R , la relación original. Llamemos a la matriz de adyacencias que representa a R^ℓ de esta relación C^ℓ y a cada elemento de la misma c_{ij}^ℓ . Si estuviésemos pensando en una matriz de enteros, la fórmula para calcular c_{ij}^ℓ está dada por la siguiente expresión aritmética:

$$c_{ij}^\ell = \sum_{k=1}^n c_{ik}^{\ell-1} \times m_{kj}$$

Como en este caso los elementos de las matrices C^k y M son valores lógicos (falso=0, verdadero=1) y, además, tenemos que tomar en cuenta si en la iteración anterior ya se encontró a la pareja que da la transitividad, en lugar de las operaciones de sumas tenemos operaciones de disyunción (\vee) y en lugar de multiplicaciones tenemos operaciones de conjunción (\wedge). Por lo tanto, la fórmula para el cálculo de la matriz que registra la transitividad en la relación queda como sigue:

$$c_{ij}^\ell = \begin{cases} c_{ij}^{\ell-1} \vee \left(\bigvee_{k=1}^n (c_{ik}^{\ell-1} \wedge m_{kj}) \right) & \text{si } \ell > 1 \\ m_{ij} & \text{si } \ell = 1 \end{cases}$$

La interpretación de $\bigvee_{k=1}^n$ corresponde al resultado de hacer una disyunción lógica, interpretando a 0 como falso y a 1 como verdadero, para todas las parejas con las que se lleva a cabo la conjunción $\neg a_{ik} \wedge a_{kj}$. El significado de la fórmula es que el par (a_i, a_j) va a estar en R^ℓ si la pareja ya estaba en $R^{\ell-1}$, o bien si la pareja $(a_i, a_k) \in R^{\ell-1}$ y la pareja $(a_k, a_j) \in R$, para alguna k , $1 \leq k \leq n$.

Como ya vimos que para una relación sobre un conjunto finito con n elementos, existe un ℓ tal que nos da un punto fijo, sabemos que $R^\ell = R^+$ para alguna $\ell < n$ y que $R^\ell = R^{\ell-1} \circ R$. Por lo tanto, esta fórmula se puede expresar fácilmente en un pseudocódigo similar a un lenguaje de programación, lo que se presenta en el listado 6.1, que se encuentra en la siguiente página.

Listado 6.1. Algoritmo de Warshall para calcular R^+

```

1  $A = \{a_1, a_2, \dots, a_n\}$  y sea  $R \subseteq A \times A$ .
2 boolean [][]  $M = \{a[i][j]\}$ ; // la matriz de adyacencias de  $R$ .
3 boolean [][]  $C = M$ ; // Copia  $M$  a  $C$ , una nueva matriz
4 boolean [][]  $CP = C$ ; // Se copia  $C$  a  $CP$ , una nueva matriz
5 int numIt = 1; // La primera potencia es 2
6 boolean bCambio = true; // Para registrar si cambió la relación
7 while (numIt <=  $n$  && bCambio) {
8     bCambio = false;
9     for (int i=1; i <=  $n$ ; i++) {
10         for (int j=1; j <=  $n$ ; j++) {
11              $CP[i][j] = CP[i][j] \mid (C[i][k] \ \&\& \ M[k][j])$ ;
12             bCambio = bCambio  $\mid (C[i][j] \neq CP[i][j])$ ;
13         } // for j
14     } // for i
15      $C = CP$ ;
16     numIt++;
17 }
```

En lugar de usar n matrices para las distintas R^ℓ , $1 \leq \ell \leq n$, únicamente usaremos tres matrices de $n \times n$. La primera, M , representará a R . La segunda, C , representará a $R^{\ell-1}$ y la tercera, —que en la primera iteración es igual a M —, CP (por C prima), representará a R^ℓ calculada como se acaba de describir y que es la que se está calculando en esa iteración. Antes de empezar cada iteración sustituimos a C por la última CP calculada. Como sabemos que $\ell \leq n$, cuidamos que, para seguir iterando, haya cambios en CP . Esto lo controlamos, para cada iteración, con una variable booleana **bCambio** que se hace verdadera si detecta algún cambio.

La única diferencia del pseudocódigo con la especificación matemática del problema es que, como CP empieza valiendo C , la acumulación se hace sobre la nueva versión de C , que es CP , pues de otra manera únicamente retendríamos el último valor de $C_{ij} \vee (C_{ik} \wedge M_{kj})$. De esta manera realmente se elige al valor de verdadero si y sólo si hay algún valor de verdadero en la conjunción al final de la fórmula.

Ejemplo 6.27. Veamos la matriz de adyacencias de la relación R dada por M y trabajemos con el algoritmo de Warshall para saber en cuál iteración se encuentra el punto fijo para R^+ .

$$R = \{ (a_1, a_2), (a_1, a_3), (a_2, a_1), (a_2, a_3), (a_2, a_4), (a_3, a_1), (a_3, a_2), (a_3, a_4), \\ (a_3, a_5), (a_4, a_2), (a_4, a_3), (a_4, a_5), (a_4, a_6), (a_5, a_3), (a_5, a_4), (a_5, a_7), \\ (a_6, a_4), (a_6, a_7), (a_6, a_8), (a_7, a_5), (a_7, a_6), (a_7, a_8), (a_8, a_6), (a_8, a_7) \}$$

Para esta relación la matriz de adyacencias queda como se ve en la figura 6.5 en la página que sigue.

Figura 6.5. Aplicación del Algoritmo de Warshall para cerradura

$$\begin{aligned}
 M = C^1 = & \begin{matrix} & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 \\ \begin{matrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{matrix} & \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \end{matrix} & C^2 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \\
C^3 = & \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} & C^4 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \\
C^5 = & \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}
\end{aligned}$$

De los productos obtenidos del algoritmo de Warshall podemos observar lo siguiente:

- i. En C^3 la entrada para $c_{1,8} = c_{8,1}$ es 0, lo que quiere decir que en R^3 no aparecen las parejas (a_1, a_8) ni (a_8, a_1) , pero como en C^3 aparece (a_1, a_6) y en M (a_6, a_8) , en la siguiente iteración se incorpora el par (a_1, a_8) . Para incorporar (a_8, a_1) basta observar que en C^3 tenemos el par (a_8, a_2) y en M tenemos (a_2, a_1) , por lo que también se incorporará el par (a_8, a_1) en la siguiente iteración.
- ii. En C^4 podemos observar que todas las entradas en la matriz son 1, lo que quiere decir que $R^4 = A \times A$.

- iii. A partir de C^4 el contenido de la matriz no cambia, pues ya se encontró que todos los pares en $A \times A$ están en C^4 . Sin embargo, con el algoritmo dado todavía se calcularía C_5 , pues es en esta matriz donde ya no se va a registrar ningún cambio en las entradas de la misma. No mostramos C^6 y C^7 para ahorrar espacio, aunque es claro que va a ser igual a las dos matrices anteriores.

La operación de cerradura transitiva es de gran relevancia en las ciencias de la computación. Diversos problemas se pueden modelar mediante relaciones y en particular la solución a muchos problemas involucra el cálculo de una cerradura transitiva. Como muestra mencionamos dos casos.

Ejemplo 6.28. Supongamos que se requiere construir una base de datos B con conexiones entre ciudades mediante vuelos, por ejemplo (*México, Guadalajara*), (*Toluca, Monterrey*), (*Cancún, Toluca*), etcétera. En tal caso no es necesario almacenar las conexiones indirectas, por ejemplo (*Cancún, Monterrey*). Bastará con mantener en la base de datos las conexiones directas y al consultar si existe un vuelo entre las ciudades a y b verificar si $(a, b) \in B^+$, es decir, el problema de buscar una conexión deseada entre a y b se reduce a verificar si (a, b) pertenece a la cerradura transitiva de B . Adicionalmente, si nos solicitan vuelos con 1 escala, bastaría con calcular B^2 , con dos escalas B^3 y así sucesivamente. B^+ nos da información poco precisa pues no nos dice el mínimo número de escalas para conectar dos ciudades, sino simplemente si hay o no manera de conectarlas.

Ejemplo 6.29. En una hoja de cálculo, cada vez que se actualiza una celda, en automático deben hacerlo todas las celdas que dependen de ella. Sea R una relación sobre las celdas tal que $(c_i, c_j) \in R$ si y sólo si en la fórmula o expresión declarada en la celda c_j figura una referencia a la celda c_i . Por ejemplo, si la celda $C8$ tiene declarada a la expresión “ $= 2.5 * B3$ ”, entonces $(B3, C8) \in R$. Obsérvese entonces que la relación R no puede ser reflexiva. Al momento que se actualiza una celda arbitraria x , la implementación de la hoja de cálculo debe actualizar no sólo aquellas celdas y tales que $(x, y) \in R$, sino que cualquier celda que dependa de y también debe cambiar su valor al actualizar x . Las celdas z que deben actualizarse son exactamente aquellas donde $(x, z) \in R^+$.

Ejercicios

6.3.1.- Demuestre que $\text{sim}(R) = R \cup R^{-1}$.

6.3.2.- Dadas las siguientes relaciones:

- $R_1 = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c), (b, d), (d, d)\}$,
con $A = \{a, b, c, d\}$ y $R \subseteq A \times A$.
- $R_2 = \{(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 3), (3, 4)\}$,
con $A = \{1, 2, 3, 4\}$ y $R \subseteq A \times A$.

- c) $R_3 = \{(2, 4), (3, 9), (4, 16), (4, 2), (9, 3), (16, 4)\}$,
 con $A = \{1, 2, 3, 4, 9, 16\}$ y $R \subseteq A \times A$.
- d) $R_4 = \{(a, b), (b, d), (c, f), (d, h), (a, a), (c, c), (b, a)\}$,
 con $A = \{a, b, c, d, e, f, g, h\}$ y $R \subseteq A \times A$.

Da el resultado de las siguientes operaciones entre ellas:

$$a) R_1 \cap R_4 \quad b) R_2 \setminus R_3 \quad c) R_1 \oplus R_4 \quad d) R_1 \circ R_4 \quad e) R_2^3$$

6.3.3.- Sea $A = \{1, 2, 3, 4, 5\}$ y R definida como

$$R = \{(1, 1), (1, 2), (1, 3), (2, 3), (2, 4), (3, 1), (3, 4), (3, 5), \\ (4, 2), (4, 5), (5, 1), (5, 2), (5, 4)\}$$

Encuentre: $a) R^2$. $b) R^3$. $c) R^4$. $d) R^5$.

6.3.4.- Mostrar que la composición de relaciones es asociativa, es decir, si R, S, T son relaciones binarias entonces

$$R \circ (S \circ T) = (R \circ S) \circ T$$

6.3.5.- ¿Es la composición de relaciones una operación conmutativa? Justifique.

6.3.6.- Sean $R \subseteq A \times B$. Muestre que $R \circ I_B = I_A \circ R = R$. Recuerde que I_X denota a la relación identidad sobre el conjunto X .

6.3.7.- Sean $A = \{0, 1, 2, 3\}$, $R = \{(a, b) \mid (b = a + 1) \text{ o bien } (b = a \div 2)\}$, —recordemos que \div es el cociente entero— y $S = \{(a, b) \mid a = b + 2\}$. Calcular las siguientes composiciones:

$$a) R \circ S \quad b) S \circ R \quad c) R \circ S \circ R \quad d) R^3$$

6.3.8.- Sean R, S, T relaciones binarias tales que $R \subseteq S$. Mostrar que

$$a) R \circ T \subseteq S \circ T \quad b) T \circ R \subseteq T \circ S \quad c) R^{-1} \subseteq S^{-1}$$

6.3.9.- Sea A el conjunto de personas. Definimos la relaciones R y S sobre A como aRb si y sólo si a es un padre de b (padre o madre), aSb si y sólo si a es hermano o hermana de b (en este y otros ejercicios usamos notación infija, es decir, aQb significa $(a, b) \in Q$). Describa cuáles relaciones familiares corresponden a las relaciones $S \circ R$ y $R \circ S$.

6.3.10.- Tenemos las siguientes relaciones sobre los números reales:

$$\begin{array}{ll} a) R_1 = \{(a, b) \mid a > b\}. & d) R_4 = \{(a, b) \mid a \leq b\}. \\ b) R_2 = \{(a, b) \mid a \geq b\}. & e) R_5 = \{(a, b) \mid a = b\}. \\ c) R_3 = \{(a, b) \mid a < b\}. & f) R_6 = \{(a, b) \mid a \neq b\}. \end{array}$$

Encuentre:

$$\begin{array}{llll} a) R_1 \cup R_3 & b) R_1 \cup R_5 & c) R_3 \cup R_4 & d) R_3 \cup R_6 \\ e) R_2 \cap R_4 & f) R_3 \cap R_5 & g) R_3 \cap R_6 & h) R_4 \cap R_6 \end{array}$$

6.3.11.- Con las relaciones del ejercicio 6.3.10, encuentre:

$$\begin{array}{llll} a) R_1 \setminus R_2 & b) R_2 \setminus R_1 & c) R_3 \setminus R_6 & d) R_6 \setminus R_3 \\ e) R_1 \oplus R_3 & f) R_2 \oplus R_4 & g) R_2 \oplus R_6 & h) R_3 \oplus R_5 \end{array}$$

6.3.12.- Con las relaciones del ejercicio 6.3.10, encuentre:

- a) $R_1 \circ R_1$ b) $R_1 \circ R_3$ c) $R_1 \circ R_5$ d) $R_2 \circ R_3$
 e) $R_2 \circ R_1$ f) $R_3 \circ R_5$ g) $R_5 \circ R_3$ h) $R_4 \circ R_6$

6.3.13.- Sean $A = \mathbb{N}$, R_1 la relación de divisibilidad y R_2 la relación *ser múltiplo de*. Encuentre:

- a) $R_1 \cup R_2$ b) $R_1 \setminus R_2$ c) $R_1 \oplus R_2$ d) $R_1 \cap R_2$ e) $R_2 \setminus R_1$

6.3.14.- Calcule la relación inversa R^{-1} para cada una de las relaciones del ejercicio 6.3.10.

6.3.15.- Demuestre las siguientes propiedades de la operación de relación inversa.

- a) Idempotencia: $(R^{-1})^{-1} = R$.
 b) Composición: $(R \circ S)^{-1} = S^{-1} \circ R^{-1}$.

6.3.16.- Sea $R \subseteq A \times B$ una relación binaria. Definimos a la relación complemento de R , denotada \bar{R} , como $\bar{R} = \{(a, b) \in A \times B \mid (a, b) \notin R\}$. Calcule \bar{R} para las siguientes relaciones R .

- a) Cada relación R_i del ejercicio 6.3.10.
 b) La relación de divisibilidad $R \subseteq \mathbb{Z} \times \mathbb{Z}$.
 c) R la relación entre estados de México tal que $(a, b) \in R$ si y sólo si a y b tienen frontera común.

6.3.17.- Para cada una de las relaciones que siguen, $R \subseteq A \times A$, dibuje

- i. el diagrama de coordenadas,
 ii. la gráfica dirigida y
 iii. la matriz booleana.

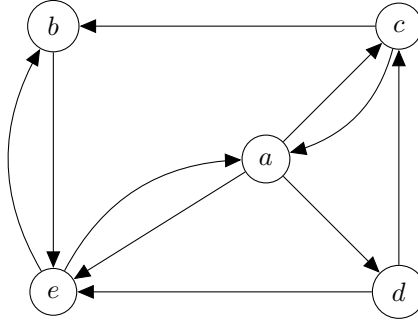
- a) $A = \{1, 2, 3, 4, 5, 6, 7, 8\}$. $(a, b) \in R$ si y sólo si $a < b$.
 b) $A = \{1, 2, 3, 4, 5, 6, 7, 8\}$. $(a, b) \in R$ si y sólo si $a = b$.
 c) $A = \{a, b, c, d, e, f\}$. $(x, y) \in R$ si y sólo si x e y son ambas vocales o ambas consonantes.
 d) $R \subseteq \mathcal{P}(A) \times \mathcal{P}(A)$, donde $A = \{1, 2, 3\}$ y $\mathcal{P}(A)$ es el conjunto de todos los subconjuntos de A , dada por $(a, b) \in R$ si y sólo si $a \subseteq b$.
 e) $R = \{(a, b), (a, c), (a, e), (b, a), (b, d), (c, b), (d, a), (d, d)\}$.

6.3.18.- Tenemos las siguientes matrices M_R y M_S que denotan a dos relaciones binarias R y S respectivamente:

$$M_R = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix} \quad M_S = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- a) Enumere los elementos de R y S . b) Dibuje las gráficas dirigidas de R y S .

6.3.19.- Una relación R sobre el conjunto $A = \{a, b, c, d, e\}$ tiene la gráfica dirigida que se muestra enseguida:



- a) Liste los elementos de R . b) Escriba la matriz booleana de R .

6.3.20.- Sean

$$A = \{1, 2, 3\}$$

$$B = \mathcal{P}(\{a, b, c\}) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}.$$

Sea R una relación sobre $A \times B$, denotada por la siguiente matriz booleana:

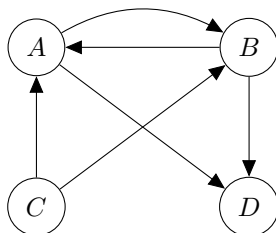
$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Liste los elementos de R , suponiendo que los renglones y las columnas están etiquetadas en el orden en que aparecen los elementos de A y B respectivamente.

6.3.21.- Tenemos cuatro equipos de fútbol, A , B , C , y D , que juegan cada uno contra los otros tres equipos dos partidos, uno en casa y otro en la casa del oponente. Una relación R en el conjunto $J = \{A, B, C, D\}$ está definida de la siguiente manera:

$(X, Y) \in R$ si y sólo si X le gana a Y cuando X juega en casa.

El siguiente diagrama es la gráfica dirigida de R .



Liste los elementos de R y escriba su matriz booleana.

6.3.22.- Encuentre condiciones sobre la digráfica o la matriz de una relación R que indiquen cuándo R es simétrica, antirreflexiva, antisimétrica o asimétrica.

6.3.23.- Sea $A = \{a, b, c, d\}$.

- a) ¿Cuántas relaciones diferentes hay sobre A ?
- b) ¿Cuántas de esas relaciones contienen a (a, a) ?
- c) ¿Cuáles de ellas son reflexivas?
- d) ¿Cuáles de ellas son simétricas?
- e) ¿Cuáles de ellas son antisimétricas?
- f) ¿Cuáles de ellas son transitivas?

6.3.24.- Sea A el conjunto de todas las páginas web en existencia. Definimos las siguientes relaciones sobre A :

- a) aR_1b si todo el que ha visitado la página web a también ha visitado la página web b .
- b) aR_2b si existe una página web que contiene ligas tanto a la página web a como a la página web b .
- c) aR_3b si no existe ninguna liga en común en las páginas web a y b .
- d) aR_4b si hay al menos una liga en común en la página web a y la página web b .

Determine, para cada una de estas relaciones, si cumplen o no con la propiedad de ser reflexiva, simétrica, antisimétrica y/o transitiva.

6.3.25.- Sea A el conjunto de todas las personas. Definimos las siguientes relaciones sobre A :

- a) aRb si a es más alto que b .
- b) aRb si a y b nacieron el mismo día.
- c) aRb si a y b se llaman igual.
- d) aRb si a y b tienen un abuelo en común.

Determine, para cada una de estas relaciones, si cumplen o no con la propiedad de ser reflexiva, simétrica, antisimétrica y/o transitiva.

6.3.26.- Para las siguientes relaciones, argumenta si cumplen o no con las propiedades de reflexividad, simetría, antisimetría y transitividad.

- a) Sea R una relación definida en los números reales tal que xRy si y sólo si $x \leq y$.
- b) Sean $A = \{a, b, c, d\}$ y $R = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c), (b, d), (d, d)\}$.
- c) Sea $A = \mathbb{Z}^+ \times \mathbb{Z}^+$ y R la relación definida por
 $(a, b) R (c, d)$ si y sólo si $a + d = b + c$.
- d) Sea $A = \{\text{falso}, \text{verdadero}\}$ y aRb si y sólo si $(a \rightarrow b) \wedge (a \rightarrow \neg b)$ es verdadero.

6.3.27.- ¿Puede una relación en un conjunto no ser ni reflexiva ni antirreflexiva? Justifique.

6.3.28.- ¿Puede una relación en un conjunto ser simétrica y antisimétrica? Justifique.

6.3.29.- Sea R una relación reflexiva sobre un conjunto A . Muestre que R^n es reflexiva para toda $n \in \mathbb{N}$.

- 6.3.30.- Supongamos que la relación R sobre un conjunto A no es reflexiva. ¿Qué se puede decir de R^2 , respecto a si es reflexiva o no? Justifique.
- 6.3.31.- Muestre que una relación R sobre A es reflexiva si y sólo si la relación $I_A \subseteq R$.
- 6.3.32.- Muestre que una relación R sobre A es antirreflexiva si y sólo si $R \cap I_A = \emptyset$.
- 6.3.33.- Muestre que una relación R sobre A es simétrica si y sólo si $R = R^{-1}$.
- 6.3.34.- Muestre que una relación R sobre A es antisimétrica si y sólo si $R \cap R^{-1} \subseteq I_A$.
- 6.3.35.- Muestre que una relación R sobre A es transitiva si y sólo si $R \circ R \subseteq R$.
- 6.3.36.- Muestre que si R es reflexiva y transitiva entonces para toda $n > 0$ se cumple $R^n = R$.
- 6.3.37.- Muestre que si R es simétrica, entonces para toda $n > 0$, R^n es simétrica.
- 6.3.38.- Demuestre las siguientes propiedades de las relaciones asimétricas.
- Si R es asimétrica entonces R es antirreflexiva.
 - Si R es asimétrica entonces R es antisimétrica.
 - R es asimétrica si y sólo si R es antirreflexiva y antisimétrica.
- 6.3.39.- Hallar las cerraduras reflexiva, simétrica y transitiva para las siguientes relaciones sobre A .
- $A = \{1, 2, 3, 4\}$, $R = \{(1, 2), (4, 3), (2, 2), (2, 1), (3, 1)\}$
 - $A = \{a, b, c, d\}$, $R = \{(a, b), (b, d), (c, b), (d, a)\}$
 - $A = \{a, b, c, d, e\}$,
 $R = \{(a, e), (b, a), (b, d), (c, d), (d, a), (d, c), (e, a), (e, b), (e, c), (e, e)\}$
- 6.3.40.- Dada una relación binaria $R \subseteq A \times A$, denotamos con R^* a la cerradura reflexiva y transitiva de R . Muestre que
- $R^* = I_A \cup R^+$.
 - $R^+ = R \circ R^*$.
- 6.3.41.- Sea $R \subseteq A \times A$. Muestre que la cerradura transitiva R^+ es la más pequeña, con respecto a \subseteq , entre las relaciones X tales que
- $$R \subseteq X \text{ y } R \circ X \subseteq X.$$
- 6.3.42.- Muestre que $R \cup R^{-1}$ es la relación más pequeña, con respecto a \subseteq , entre las relaciones X que cumplen
- $$X \circ R \subseteq X \text{ y } X^{-1} \subseteq X.$$
- 6.3.43.- Sea R una relación binaria sobre A . Demuestre lo siguiente acerca de las operaciones de cerradura.
- $\text{refl}(\text{trans}(R)) = \text{trans}(\text{refl}(R))$.
 - $\text{refl}(\text{sim}(R)) = \text{sim}(\text{refl}(R))$.
 - $\text{sim}(\text{trans}(R)) \subseteq \text{trans}(\text{sim}(R))$. De un ejemplo que muestre que la igualdad no es válida.
- 6.3.44.- Describa a la cerradura reflexiva de la relación vacía $\emptyset \subseteq A \times A$.
- 6.3.45.- Calcule la cerradura transitiva de la relación $R = \{(x, y) \in \mathbb{Z} \times \mathbb{Z} \mid x+y \text{ es impar}\}$.

- 6.3.46.- Sea $R \subseteq A \times A$ una relación ¿Qué significa que $I_A \cap R^+ = \emptyset$?
- 6.3.47.- Como mencionamos al discutir el Algoritmo de Warshall en el ejemplo 6.28, cuando obtenemos R^+ sabemos cuáles son todas las parejas en la relación, pero no sabemos cuál es la mínima ℓ para la cuál una pareja dada aparece en la relación. ¿Hay manera de determinar para cuál ℓ una pareja ingresa a una relación? Si esto es posible, ¿cuáles son los cambios que le haría al pseudocódigo dado para el Algoritmo de Warshall para detectar esta situación?
- 6.3.48.- Tenemos el conjunto $A = \{a, b, c, d, e\}$. Construya la cerradura transitiva R^+ de las siguientes relaciones sobre A , usando el algoritmo de Warshall.
- a) $R = \{(a, c), (b, d), (c, a), (d, b), (e, d)\}$
 - b) $R = \{(b, c), (b, e), (c, e), (d, a), (e, b), (e, c)\}$
 - c) $R = \{(a, b), (a, c), (a, e), (b, a), (b, c), (c, a), (c, b), (d, a), (e, d)\}$
 - d) $R = \{(a, e), (b, a), (b, d), (c, d), (d, a), (d, c), (e, a), (e, b), (e, c), (e, e)\}$

6.4. Relaciones de equivalencia

Existen diversas situaciones en la práctica donde no nos interesa la naturaleza particular de un individuo u objeto sino sólo algunas de sus propiedades. Por ejemplo, ser hombre o mujer en vez de ser Lorenzo o Marina; ser un número par o impar en vez de 38 o 55; o bien ser un programa que al recibir como entrada un número devuelve otro número, en vez de ser el programa que calcula el factorial o la mitad de un número. En estos casos nos interesa agrupar en una misma clase a todos los objetos que cumplan la propiedad en cuestión y tratar como iguales a cualesquiera individuos de una clase particular. Para formalizar esta idea usamos una clase especial de relación que respete las propiedades esenciales de la igualdad. A estas relaciones las llamamos *relaciones de equivalencia* y las estudiamos a continuación.

Definición 6.7 (relación de equivalencia) Una *relación de equivalencia* es una relación binaria R en la que se cumplen las propiedades de reflexividad, simetría y transitividad. Si R es una relación de equivalencia y $(a, b) \in R$, entonces decimos que a y b son equivalentes respecto a R .

Obsérvese que estas propiedades las cumple la relación de igualdad en cualquier conjunto, la cual es entonces una relación de equivalencia. De hecho cualquier relación cuya definición se base en la relación de igualdad será una relación de equivalencia.

Ejemplo 6.30. La relación \equiv entre fórmulas de la lógica proposicional es una relación de equivalencia ya que, como discutimos en su momento, cumple las tres propiedades requeridas, las cuales se demuestran fácilmente. Esto es porque $A \equiv B$ sucede si y sólo si para cualquier interpretación \mathcal{I} se cumple que $\mathcal{I}(A) = \mathcal{I}(B)$; por lo tanto, la relación \equiv hereda las propiedades requeridas de la igualdad.

Revisemos nuevamente algunos de los ejemplos que hemos dado para determinar cuáles de ellos conforman una relación de equivalencia. Empezaremos por las relaciones mencionadas en la sección 6.3.3, para las que ya determinamos cuáles son las propiedades que cumplen –recurrir a esta sección para ver a cuáles relaciones nos estamos refiriendo–.

Ejemplo 6.31. En las relaciones R_1 a R_6 listadas en la página 266 tenemos lo siguiente respecto a la relación de equivalencia.

- R_1 y R_2 no son relaciones de equivalencia, pues la primera no cumple ninguna de las propiedades y la segunda es reflexiva y transitiva pero no simétrica.
- R_3 es una relación de equivalencia pues cumple con ser reflexiva, simétrica y transitiva.
- R_4 y R_6 no son reflexivas (tampoco simétricas o transitivas) por lo que no son una relación de equivalencia.
- R_5 es reflexiva y transitiva pero no es simétrica, por lo que tampoco es una relación de equivalencia.

Estos ejemplos nos dan explícitamente la relación, por lo que simplemente hay que revisar cuáles son las parejas que se encuentran en ella. Veamos ahora otros ejemplos donde la relación se da en términos de una propiedad.

Ejemplo 6.32. Determinar si la relación $R_1 = \{(a, b) \mid a, b \in \mathbb{Z}, a < b\}$ es o no una relación de equivalencia.

Solución:

- **Reflexividad:** No la cumple pues ningún número es *estrictamente* menor que sí mismo.
- **Simetría:** Tampoco la cumple pues si $a < b$ entonces b no puede ser *estrictamente* menor que a .
- **Transitividad:** Esta propiedad sí la tiene pues si $a < b$ y $b < c$ sabemos que $a < c$.

En realidad deberíamos haber concluido que la relación no es de equivalencia con la primera propiedad inválida, ya que con una propiedad que no se cumpla es suficiente, pero seguimos adelante para dar una exposición más completa.

Ejemplo 6.33. Determinar si la relación $R_2 = \{(a, b) \mid a, b \in \mathbb{Z}, a \leq b\}$ es o no una relación de equivalencia.

Solución:

- **Reflexividad:** Todo número es menor o igual a sí mismo (en particular igual).
- **Simetría:** No la cumple. Por ejemplo $7 \leq 9$ pero $9 \not\leq 7$.
- **Transitividad:** Esta propiedad sí la tiene pues si $a \leq b$ y $b \leq c$ sabemos que $a \leq c$.

Como la relación no es simétrica, tampoco es una relación de equivalencia.

Ejemplo 6.34. Determinar si la relación $R_3 = \{(a, b) \mid a, b \in \mathbb{Z}, a = b\}$ es o no una relación de equivalencia.

Solución: R_3 no es más que la relación de igualdad en los enteros por lo que claramente resulta una relación de equivalencia. De cualquier forma verificamos las propiedades correspondientes:

- **Reflexividad:** Es claro que todo número es igual a sí mismo, por lo que $(a, a) \in R_3$.
- **Simetría:** También es una relación simétrica, pues si $a = b$ sabemos que $b = a$ y por lo tanto ambas parejas $(a, b), (b, a) \in R_3$.
- **Transitividad:** Esta propiedad también la tiene R_3 , pues si $a = b$ y $b = c$ sabemos que $a = c$.

El siguiente ejemplo es de relevancia primordial en la rama de las matemáticas conocida como teoría de los números, área de estudio que tiene importantes aplicaciones en teoría de códigos y criptografía y, por lo tanto, en lo relacionado a seguridad computacional.

Ejemplo 6.35. Sea $n \in \mathbb{Z}$, $n > 0$. Demostrar que la relación

$$R_4 = \{(a, b) \mid a, b \in \mathbb{Z}, \text{ con } (a \bmod n) = (b \bmod n)\}$$

es una relación de equivalencia³.

Solución: Podemos notar que la definición depende de una constante positiva n independiente de la pareja, la misma para todas las parejas. Cada constante n distinta nos definiría una relación de equivalencia distinta, por lo que realmente estamos definiendo una familia de relaciones.

Es claro que estamos ante una relación de equivalencia, puesto que su definición se basa en la igualdad. De cualquier forma verificamos las propiedades.

- **Reflexividad:** Esto es claro pues siempre sucede que $a \bmod c = a \bmod c$.
- **Simetría:** También es una relación simétrica, pues si $a \bmod c = b \bmod c$, como la igualdad es simétrica, tendremos también $b \bmod c = a \bmod c$.
- **Transitividad:** Es análogo al caso anterior, pues la igualdad es transitiva.

Esta relación se conoce como la *relación de congruencia*. Si $(a, b) \in R_4$ entonces decimos que a es congruente con b módulo n . Es fácil ver que a es congruente con b módulo n si y sólo si $n \mid (b - a)$.

Ejemplo 6.36. Determinar si la relación $R_6 = \{(a, b) \mid a, b \in \mathbb{Z}, a \text{ divide a } b (a \mid b)\}$ es o no una relación de equivalencia.

Solución: Recordemos que $a \mid b$ si y sólo si $b = k \cdot a$ para alguna $k > 0$, $k \in \mathbb{Z}$.

- **Reflexividad:** Por supuesto que $a \mid a$ pues $a = 1 \cdot a$.
- **Simetría:** La relación no es simétrica. Por ejemplo $4 \mid 24$ y no existe un entero k tal que $4 = k \cdot 24$, por lo que $24 \nmid 4$.

$\therefore R_6$ no es una relación de equivalencia.

Ejemplo 6.37. Determinar si la relación $R_7 = \{(a, b) \mid a, b \in \mathbb{Z}, a \text{ no es múltiplo de } b\}$ es o no una relación de equivalencia.

³Tiene mayor precedencia el operador \bmod que la relación de igualdad ($=$) por lo que, en adelante, no usaremos paréntesis innecesarios.

Solución: La relación no es reflexiva puesto que $a = 1 \cdot a$ y entonces a siempre es múltiplo de a . Por lo tanto $(a, a) \notin R_7$. La relación tampoco es simétrica; por ejemplo, $(4, 8) \in R_7$ pero $(8, 4) \notin R_7$, ya que 8 **sí** es múltiplo de 4. Tampoco es transitiva, pues $(8, 10) \in R_7$ y $(10, 4) \in R_7$, pero $(8, 4) \notin R_7$, pues 8 sí es múltiplo de 4.

Ejemplo 6.38. Sea A el conjunto de clases escritas en un lenguaje de programación como Java. Decimos que dos clases están relacionadas según R si la línea de herencia es la misma para ambas. ¿Es ésta una relación de equivalencia?

Solución: Veamos si esta relación cumple con las propiedades necesarias.

- **Reflexividad:** Obviamente toda clase está relacionada consigo misma, ya que corresponden exactamente a la misma definición.
- **Simetría:** También es claro que si ambas tienen la misma línea de herencia, no importa a quién mencionemos primero.
- **Transitividad:** Si dos clases C_1 y C_2 tienen la misma línea de herencia y las clases C_2 y C_3 tienen la misma línea de herencia, es claro que C_1 y C_3 tienen la misma línea de herencia.

$\therefore R$ es una relación de equivalencia.

Observen que el último ejemplo da nuevamente una relación definida a partir de la igualdad, en este caso dada por la propiedad de tener *la misma* línea de herencia.

6.4.1. Clases de equivalencia

Una vez que se tiene definida una relación de equivalencia, podemos usarla en lugar de la igualdad, identificando o considerando a todos aquellos individuos que sean equivalentes como uno mismo. Por ejemplo, si se desea hacer una implementación de conjuntos de números naturales usando listas, vamos a considerar equivalentes a aquellas listas que tengan los mismos elementos, sin importar su orden ni el número de veces que figuran en la lista, como son $[1, 3, 8]$, $[8, 3, 1]$, $[1, 3, 8, 1]$, $[8, 3, 1, 1, 3]$ y $[3, 8, 1, 8]$, entre otras. Esta relación permite hacer una clasificación del conjunto de todas las listas en subconjuntos ajenos cuya unión es todo el conjunto. Los conceptos formales que nos permiten trabajar considerando a todos los individuos equivalentes como uno solo son el de clase de equivalencia y el de partición de un conjunto en clases de equivalencia, discutidos a continuación.

Definición 6.8 (partición) Una *partición* de un conjunto S es una colección de subconjuntos de S , digamos S_1, S_2, \dots, S_k con $S_i \subseteq S, i = 1, \dots, k$, que cumplen con:

1. $S_i \neq \emptyset$, para toda $i = 1, \dots, k$
2. $S = \bigcup_{i=1}^n S_i$
3. $S_i \cap S_j = \emptyset$, para toda $i, j = 1, \dots, k$, con $i \neq j$.

La primera condición nos dice que ningún subconjunto puede ser vacío. La segunda requiere que el conjunto S sea el resultado de unir a todos los subconjuntos; o sea que cada elemento de S esté en algún subconjunto. La última condición nos dice que ningún elemento puede estar en más de un subconjunto.

Ejemplo 6.39. Sea $S = \{1, 2, \dots, 10\}$. Las siguientes son algunas particiones de S :

1. $S_1 = S$ (llamada la *partición trivial*).
2. $S_i = \{i\}, i = 1, \dots, 10$ (donde cada subconjunto contiene sólo a uno de los elementos del conjunto y que se conoce como la *partición total*).
3. $S_1 = \{1, 3, 5, 7, 9\}, S_2 = \{2, 4, 6, 8, 10\}$.
4. $S_1 = \{1, 10\}, S_2 = \{2, 9\}, S_3 = \{3, 8\}, S_4 = \{4, 7\}, S_5 = \{5, 6\}$.
5. $S_1 = \{1, 3, 8\}, S_2 = \{7\}, S_3 = \{9, 2, 4, 6\}, S_4 = \{5, 10\}$.

Se observa en el ejemplo 6.39, que algunas de estas particiones se pueden representar con ciertas relaciones de equivalencia, en el sentido que los subconjuntos se pueden definir como conjuntos de elementos relacionados de cierta manera.

- Por ejemplo, en el caso 1 podemos definir $S_1 = \{a \in S \mid (a, b) \in U_S\}$, donde recordamos que U_S es la relación universal en S .
- En el caso 2 tenemos $S_i = \{a \in S \mid (a, b) \in I_S\}$, donde I_S es la relación identidad en S (es decir, la igualdad).
- Para el caso 3, si definimos $R = \{(a, b) \in S \times S \mid a, b \text{ son ambos par o impar}\}$, entonces R es una relación de equivalencia con $S_1 = \{b \in S \mid (1, b) \in R\}, S_2 = \{b \in S \mid (2, b) \in R\}$ y $R = \{(a, b) \mid a \bmod 2 = b \bmod 2, a, b \in S\}$, o bien $R = \{(a, b) \mid a \equiv_2 b\}$.

Este fenómeno es cierto en general: cuando tenemos una relación de equivalencia en un conjunto, ésta *induce* (ocasiona, impone) una partición en el conjunto, donde quedan en un mismo subconjunto aquellos elementos que son equivalentes. A cada uno de estos subconjuntos es a lo que llamamos *clase de equivalencia*, definida formalmente como sigue.

Definición 6.9 (clase de equivalencia) Sean $R \subseteq A \times A$ una relación de equivalencia y $a \in A$. La clase de equivalencia de a bajo R , denotada $[a]_R$, o simplemente $[a]$, es el conjunto de aquellas $b \in A$ tales que $(a, b) \in R$. Es decir, $[a] = \{b \in A \mid (a, b) \in R\}$.

Si $b \in [a]$ entonces decimos que b es un *representante* de la clase $[a]$. El conjunto de clases de una relación de equivalencias R es de gran importancia en matemáticas y se conoce como el *conjunto cociente de A bajo R* , denotado A/R : $\mathbf{A}/\mathbf{R} = \{[a] \mid a \in A\}$.

Volviendo al ejemplo 6.39, tenemos que:

- En el caso 1 $S_1 = [1]$;
- en el caso 2 $S_i = [i]$;
- en el caso 1 se verifica que $S_1 = [1], S_2 = [2]$.
- Obsérvese que en el primer caso también sucede que $S_1 = [2] = [3] = \dots = [10]$ y
- en el tercer caso tenemos $S_1 = [3] = [5] = [7] = [9]$ y $S_2 = [4] = [6] = [8] = [10]$.

Esta situación deja ver que muchos elementos del conjunto generan la misma clase de equivalencia y que una relación de equivalencia induce una partición dada por el conjunto A/R , es decir, cada subconjunto de la partición es una clase de equivalencia. Esto sucede de manera general como se asevera en la proposición 6.5.

Proposición 6.5 Sean $R \subseteq A \times A$ una relación de equivalencia y $a, b \in A$. Demostraremos que las clases inducidas por la relación de equivalencia corresponden a una partición del conjunto, de acuerdo a la definición 6.8 que dimos de lo que es una partición.

1. $a \in [a]$ y por lo tanto $[a] \neq \emptyset$.
2. $[a] = [b]$ si y sólo si $(a, b) \in R$.
3. Si $(a, b) \notin R$ entonces $[a] \cap [b] = \emptyset$.
4. $A = \bigcup_{a \in A} [a]$.

Demostración.

1. Como R es reflexiva, entonces $(a, a) \in R$; es decir $a \in [a]$, lo que demuestra que ningún subconjunto es el conjunto vacío.
2. Si $[a] = [b]$ entonces, usando el inciso 1, $b \in [a]$, es decir $(a, b) \in R$. Ahora supongamos que $(a, b) \in R$. Si $c \in [a]$ entonces $(a, c) \in R$ y por simetría $(c, a) \in R$; así, por transitividad, $(c, b) \in R$ y por simetría $(b, c) \in R$, es decir $c \in [b]$. Por lo tanto hemos probado que $[a] \subseteq [b]$. Similarmente se prueba que $[b] \subseteq [a]$. Con ambos resultados podemos decir que $[a] = [b]$.
3. Por contrapositiva. Supongamos que $[a] \cap [b] \neq \emptyset$ y sea $c \in [a] \cap [b]$. Entonces $(a, c) \in R$ y $(b, c) \in R$. De aquí usando simetría y transitividad se concluye que $(a, b) \in R$, lo que contradice que la intersección sea vacía..
4. Se deja como ejercicio. □

Corolario 6.6 El conjunto cociente A/R es una partición de A .

Este corolario indica que toda relación de equivalencia R sobre A genera una partición de A , dada por las clases de equivalencia. Resulta natural preguntarse si el recíproco es cierto, es decir, si toda partición de A es generada por una relación de equivalencia. De manera más formal, nos preguntamos si dada una partición S de A existirá una relación de equivalencia R sobre A tal que $S = A/R$. La respuesta es positiva y la demostración se deja como ejercicio.

Determinamos ahora los conjuntos cociente de algunas de las relaciones de equivalencia discutidas anteriormente.

Ejemplo 6.40. Determina las clases de equivalencia para $R_3 = \{(a, b) \mid a, b \in \mathbb{Z}, a = b\}$, que vimos en el ejemplo 6.34.

Solución: En el caso de los enteros, cada elemento está en su propia clase ya que no hay ningún elemento que sea igual a otro. Por lo tanto, el número de clases es infinito y cada clase contiene a un único elemento, es decir, la partición generada es la partición total:

$$\mathbb{Z}/R_3 = \{ \dots, [-2], [-1], [0], [1], [2] \dots \}.$$

Si tomásemos $a, b \in \mathbb{Q}$, los racionales, entonces tendríamos nuevamente un número infinito de clases de equivalencia y además un número infinito de elementos en cada una, pues un racional puede ser igual a otro teniendo distintos numerador y denominador. Por ejemplo, $[1/2] = \{1/2, 2/4, 3/6, 4/8, \dots\}$.

Ejemplo 6.41. Describe las clases de equivalencia para la relación de congruencia

$$R_n = \{(a, b) \mid a, b \in \mathbb{Z}, a \bmod n = b \bmod n\} \text{ con } n > 0 \text{ (véase el ejemplo 6.35).}$$

Solución: Dada $n > 0$, hay n distintos valores para $a \bmod n$, a saber $0, 1, \dots, n-1$, por lo que las clases de equivalencia son:

$$[0] = \{a \in \mathbb{Z} \mid a = k \cdot n, k \in \mathbb{Z}\}$$

$$[1] = \{a \in \mathbb{Z} \mid a = k \cdot n + 1, k \in \mathbb{Z}\}$$

$$\vdots$$

$$[c-1] = \{a \in \mathbb{Z} \mid a = k \cdot n + (n-1), k \in \mathbb{Z}\}$$

Por lo tanto, $\mathbb{Z}/R_n = \{[0], [1], \dots, [c-1]\}$. Este conjunto cociente es de gran importancia en varias áreas de las Matemáticas y se conoce como la *estructura de enteros módulo n* , usualmente denotada con \mathbb{Z}_n .

Ejemplo 6.42. Consideremos la relación de equivalencia

$$R = \{(a, b) \mid a, b, c \in \mathbb{Z}, c > 1, a \div c = b \div c\}.$$

Describe las clases de equivalencia que induce esta relación.

Solución: Sean $c > 1$ y $a, b \in \mathbb{Z}$ tales que $(a, b) \in R$, es decir, $a \div c = b \div c$. Supongamos, sin pérdida de generalidad, que $a \leq b$ (se puede demostrar de la misma manera en el otro sentido). Entonces se debe cumplir que $b \leq a + c - 1$, pues en otro caso $b > a + c - 1$, lo cual implicaría que $b \div c = (a \div c) + 1$, lo que haría $a \div c \neq b \div c$. Por lo tanto, se tiene $a \leq b \leq a + c - 1$, de donde los posibles valores para b son $a, a+1, a+2, \dots, a+c-1$. Así se tiene que la clase de equivalencia de a es $[a] = \{a, a+1, \dots, a+c-1\}$. Por ejemplo, si $c = 3$ entonces

$$[-6] = \{-6, -5, -4\}$$

$$[-3] = \{-3, -2, -1\}$$

$$[0] = \{0, 1, 2\}$$

$$[3] = \{3, 4, 5\}$$

$$[6] = \{6, 7, 8\}$$

$$\vdots$$

$$[3k] = \{3k, 3k+1, 3k+2\}.$$

Por lo anterior, hay tantas clases de equivalencia como enteros, una por cada múltiplo de c .

Ejemplo 6.43. Consideremos nuevamente $A \neq \emptyset$, un conjunto arbitrario de personas, y la relación R definida de la siguiente manera:

$$R = \{(a, b) \mid a, b \in A, a \text{ y } b \text{ nacieron en el mismo país}\}.$$

Describe las clases de equivalencia y cuántas de ellas puede haber.

Solución: Según la definición de R , estarán en la misma clase de equivalencia aquellas personas del conjunto A que hayan nacido en el mismo país. Por ejemplo, si Clara pertenece

a A y $Clara$ es venezolana entonces $[Clara]$ consta de todos los venezolanos que figuren en A . En general, habrá a lo más tantas clases de equivalencia como personas en A o países del mundo, la cardinalidad menor. Si A tiene más personas que el número total n de países del mundo, entonces habrá a lo más n clases, siendo n cuando en A haya personas de cada nacionalidad posible. En otro caso el número de clases será a lo más la cardinalidad de A . Si el número de clases es igual a la cardinalidad de A es porque todas las personas en A son de distinta nacionalidad, lo cual corresponde a la partición total. Por otra parte, si todas las personas en A tienen la misma nacionalidad entonces habrá una sola clase de equivalencia, lo cual corresponde a la partición trivial.

Ejemplo 6.44. Sea $R = \{(x, y) \in \mathbb{R} \times \mathbb{R} \mid x, y \neq 0 \wedge xy > 0\} \cup \{(0, 0)\}$. Se deja como ejercicio demostrar que R es una relación de equivalencia. ¿Cuáles son las clases de equivalencia definidas por esta relación?

Solución: Se observa que un producto xy es positivo si y sólo si ambos x, y tienen el mismo signo, es decir, sucede que $x < 0$ e $y < 0$ o bien $x > 0$ e $y > 0$. Así, todos los positivos están relacionados entre sí y lo mismo sucede con todos los negativos, pero el cero sólo se relaciona consigo mismo. Por lo tanto existen tres clases de equivalencia, que escogemos definir mediante los representantes $-1, 1$ y necesariamente 0 .

$$[-1] = \{x \in \mathbb{R} \mid (-1, x) \in R\} = \{x \in \mathbb{R} \mid x < 0\}$$

$$[0] = \{0\}$$

$$[1] = \{x \in \mathbb{R} \mid (1, x) \in R\} = \{x \in \mathbb{R} \mid x > 0\}$$

Para terminar nuestra discusión de las relaciones de equivalencia, mostramos un ejemplo extendido, que ilustra una aplicación de este concepto en la implementación de una estructura de datos.

6.4.2. Estudio de un caso: conjuntos mediante listas

En esta sección estudiamos y desarrollamos una solución al problema de implementar una estructura de datos para conjuntos finitos, cuyos elementos se tomen de un tipo dado A . En este ejemplo nos bastará con implementar las operaciones de pertenencia, unión e intersección. De la misma manera como hemos definido y construido listas o árboles con elementos de un tipo A , nos gustaría ahora construir un tipo para conjuntos finitos, denotado $Conj_A$. Por ejemplo, si $A = \mathbb{N}$ entonces algunos elementos de $Conj_{\mathbb{N}}$ son:

$$\{0, 1\}, \{1, 3, 5\}, \{2, 7, 8, 10\}.$$

Lo más adecuado para cumplir nuestro propósito es utilizar estructuras conocidas o ya implementadas, pues esto nos facilitará el trabajo. Nos parece que usar listas es una buena idea, puesto que son muy similares a los conjuntos. Para los ejemplos de arriba, las listas⁴ correspondientes $\langle 0, 1 \rangle$, $\langle 1, 3, 5 \rangle$, $\langle 2, 7, 8, 10 \rangle$ son una buena representación. Sin

⁴Para evitar confusión entre una lista y una clase, usaremos en el estudio de este caso la notación $\langle a, b, c \rangle$ para la lista que contiene a los elementos a, b y c , en ese orden, y usaremos la notación $[a]$ para la clase de

embargo, no podemos simplemente definir $Conj_A = List_A$ pues las listas tienen dos características indeseables para los conjuntos: en las listas el orden y la repetición de elementos son no sólo permitidos sino relevantes. Es decir, las listas $\langle 1, 3, 2 \rangle$ y $\langle 2, 3, 1 \rangle$ son diferentes y las listas $\langle 3, 2, 3 \rangle$ y $\langle 2, 3 \rangle$ también, mientras que en conjuntos se cumple que $\{1, 3, 2\} = \{2, 3, 1\}$ y $\{3, 2, 3\} = \{2, 3\}$. Más aún, se cumple que $\{1, 3, 5, 1, 3\} = \{1, 3, 5\}$ mientras que $\langle 1, 3, 5, 1, 3 \rangle \neq \langle 1, 3, 5 \rangle$. El problema se resuelve si convenimos en considerar como iguales a dos listas que sólo difieren en el orden o en tener elementos repetidos, tal como hacemos con conjuntos. Pero para que esta convención sea válida necesitamos definirla formalmente mediante una relación de equivalencia entre listas. En adelante nos limitamos a hablar de listas y conjuntos de números naturales, aunque las ideas se pueden generalizar fácilmente a cualquier tipo A .

Decimos que dos listas de números naturales l_1 y l_2 son equivalentes, denotado $l_1 \sim l_2$, si y sólo si tienen los mismos elementos. Por ejemplo, $\langle 1, 3, 5 \rangle \sim \langle 5, 3, 1 \rangle$, pero también $\langle 7, 7 \rangle \sim \langle 7, 7, 7 \rangle$ y $\langle 5, 3, 1, 4, 1 \rangle \sim \langle 1, 4, 3, 5 \rangle$. Por otro lado se tiene que $\langle 1, 3 \rangle \not\sim \langle 2, 1 \rangle$ y $\langle 4, 7, 9 \rangle \not\sim \langle 9, 7, 1, 4 \rangle$.

Para poder definir formalmente esta relación tenemos que definir primero una función *elem* que verifique si un número es elemento o no de una lista dada. Esta función se especifica como sigue⁵:

$elem\ x\ \ell = true$ si y sólo si x es un elemento de ℓ

y se define recursivamente como sigue:

$elem\ x\ \langle \rangle = false$

$elem\ x\ (a : \ell) = if\ x = a\ then\ true\ else\ elem\ x\ \ell$

Ahora ya podemos definir formalmente la relación de equivalencia deseada.

Definición 6.10 Sea $\mathbb{L}_{\mathbb{N}}$ el conjunto de todas las listas formadas con elementos de \mathbb{N} y sean l_1, l_2 en $\mathbb{L}_{\mathbb{N}}$. Decimos que l_1 es equivalente a l_2 , denotado $l_1 \sim l_2$, si y sólo si

$$\forall x \in \mathbb{N} (elem\ x\ l_1 \leftrightarrow elem\ x\ l_2).$$

Veamos que la relación \sim realmente es una relación de equivalencia.

Proposición 6.7 La relación \sim es una relación de equivalencia.

Demostración.

- **Reflexividad:** $\ell \sim \ell$ si y sólo si $\forall x \in \mathbb{N} (elem\ x\ \ell \leftrightarrow elem\ x\ \ell)$ lo cual es claramente cierto.
- **Simetría:** Supongamos que $\ell_1 \sim \ell_2$, es decir, $\forall x \in \mathbb{N} (elem\ x\ \ell_1 \leftrightarrow elem\ x\ \ell_2)$, lo cual equivale a $\forall x \in \mathbb{N} (elem\ x\ \ell_2 \leftrightarrow elem\ x\ \ell_1)$, puesto que el conectivo lógico \leftrightarrow es simétrico. Pero esto último implica que $\ell_2 \sim \ell_1$.
- **Transitividad:** Análogo al punto anterior utilizando que el conectivo lógico \leftrightarrow es transitivo. □

equivalencia que contiene a . La lista vacía la representaremos con $\langle \rangle$ en este contexto.

⁵Recuérdese la definición de lista dada en el capítulo anterior.

Extendamos las propiedades que cumple esta relación en la siguiente proposición.

Proposición 6.8 La relación \sim cumple las siguientes propiedades:

- i. Si $\ell_1 \sim \ell_2$ y $\text{elem } y \ell_1$ entonces $\text{elem } y \ell_2$.
- ii. Si $\ell_1 \sim \ell_2$ entonces para toda $y \in \mathbb{N}$, $(y : \ell_1) \sim (y : \ell_2)$
- iii. Si $\langle \rangle \sim \ell_2$ entonces $\ell_2 = \langle \rangle$.

Demostración.

- i. Es consecuencia directa de la definición de \sim mediante un razonamiento puramente lógico.
- ii. Supongamos que $\ell_1 \sim \ell_2$, es decir, $\forall x \in \mathbb{N} (\text{elem } x \ell_1 \leftrightarrow \text{elem } x \ell_2)$. Queremos demostrar que $\forall x \in \mathbb{N} (\text{elem } x (y : \ell_1) \leftrightarrow \text{elem } x (y : \ell_2))$. Obsérvese que para cualquier lista ℓ se cumple que $\text{elem } x (y : \ell)$ si y sólo si $x = y$ o $\text{elem } x \ell$. Por lo tanto basta probar que

$$\forall x \in \mathbb{N} (x = y \vee \text{elem } x \ell_1 \leftrightarrow x = y \vee \text{elem } x \ell_2).$$

Pero esto es clara consecuencia de nuestra hipótesis $\ell_1 \sim \ell_2$.

- iii. Supongamos que $\langle \rangle \sim \ell_2$. Por simetría de \sim se tiene que $\ell_2 \sim \langle \rangle$ y por la primera parte de esta proposición, si suponemos que existe $x \in \mathbb{N}$ tal que $\text{elem } x \ell_2$, se tendría que $\text{elem } x \langle \rangle$, lo cual es absurdo. Por lo tanto $\ell_2 = \langle \rangle$. □

Las proposiciones anteriores nos permiten justificar el uso de listas cualesquiera como conjuntos sin importar cuál lista escoger para la representación. Por ejemplo, alguien puede representar al conjunto $\{3, 8, 9\}$ con $\langle 9, 3, 8 \rangle$ y alguien más con $\langle 9, 3, 8, 8 \rangle$, pero claramente ambas listas son equivalentes. Por lo tanto definimos nuestra estructura para conjuntos como el conjunto cociente generado por las clases de equivalencia de la relación \sim . Es decir, $\text{Conj}_{\mathbb{N}} = \mathbb{L}_{\mathbb{N}} / \sim = \{[\ell] \mid \ell \in \mathbb{L}_{\mathbb{N}}\}$.

Obsérvese que un conjunto c es entonces una clase de equivalencia $c = [\ell]$. Por lo tanto, dos conjuntos son iguales si y sólo si tienen los mismos elementos, puesto que si $c_1 = [\ell_1]$ y $c_2 = [\ell_2]$ entonces $c_1 = c_2$ si y sólo si $[\ell_1] = [\ell_2]$, lo cual sucede si y sólo si $\ell_1 \sim \ell_2$. Pero, por definición de la relación \sim , esto indica que c_1 y c_2 tienen los mismos elementos.

Una vez definida nuestra estructura de datos para conjuntos nos queda el problema de implementar las operaciones de pertenencia, unión e intersección. Comencemos definiendo dichas operaciones para listas cualesquiera. Ya lo hemos hecho con la operación de pertenencia, pues ésta queda implementada con nuestra función *elem*.

Definición 6.11 Definimos las operaciones de unión e intersección de listas como sigue:

- **Unión:** $\text{union } \ell_1 \ell_2 = \ell_1 \sqcup \ell_2$.
- **Intersección:** $\text{inter } \langle \rangle \ell_2 = \langle \rangle$, $\text{inter } \ell_1 \langle \rangle = \langle \rangle$.
 $\text{inter } (a : \ell_1) \ell_2 = \text{if } \text{elem } a \ell_2 \text{ then } (a : \text{inter } \ell_1 \ell_2)$
else $\text{inter } \ell_1 \ell_2$.

Observen que estas son operaciones sobre listas y nosotros deseamos operaciones sobre conjuntos, las cuales definimos a continuación, junto con la operación de pertenencia:

Definición 6.12 Definimos las operaciones de pertenencia, unión e intersección de conjuntos como sigue:

- **Pertenencia:** $elemC\ x\ [\ell] = elem\ x\ \ell$
- **Unión:** $unionC\ [\ell_1]\ [\ell_2] = [union\ \ell_1\ \ell_2]$
- **Intersección:** $interC\ [\ell_1]\ [\ell_2] = [inter\ \ell_1\ \ell_2]$

Es decir, si se desea calcular la pertenencia a un conjunto, simplemente se calcula la pertenencia a la lista que lo representa; si se desea calcular la unión o intersección de dos conjuntos, entonces, como un conjunto es una clase de equivalencia de listas, se hace la operación correspondiente con los representantes de estas clases, que son listas, y se devuelve la clase de equivalencia del resultado. Por ejemplo:

$$\begin{aligned} unionC\ \{1, 3, 5\}\ \{2, 8, 9\} &= [union\ \langle 1, 3, 5 \rangle\ \langle 2, 8, 9 \rangle] \\ &= [\langle 1, 3, 5, 2, 8, 9 \rangle] \\ &= \{1, 3, 5, 2, 8, 9\}. \end{aligned}$$

Pero para poder considerar a éstas como operaciones entre conjuntos, debemos ver que las definiciones de $unionC$ e $interC$ son independientes del representante de cada clase de equivalencia. Es decir, si se eligen otros representantes el resultado volverá a ser equivalente. Por ejemplo,

$$\begin{aligned} unionC\ \{1, 3, 5\}\ \{2, 8, 9\} &= [union\ \langle 3, 1, 5, 1 \rangle\ \langle 2, 2, 9, 8, 9 \rangle] \\ &= [\langle 3, 1, 5, 1, 2, 2, 9, 8, 9 \rangle] \\ &= \{1, 3, 5, 2, 8, 9\}. \end{aligned}$$

En este caso tenemos que $[\langle 1, 3, 5, 2, 8, 9 \rangle] = [\langle 3, 1, 5, 1, 2, 2, 9, 8, 9 \rangle]$, puesto que $\langle 1, 3, 5, 2, 8, 9 \rangle \sim \langle 3, 1, 5, 1, 2, 2, 9, 8, 9 \rangle$, por lo que efectivamente en ambos casos el resultado es $\{1, 3, 5, 2, 8, 9\}$. Sin embargo, esto debe ser mostrado formalmente en todos los casos, lo cual hacemos a continuación.

El siguiente lema muestra algunas propiedades de la función $elem$ y de la relación \sim con respecto a las operaciones de unión e intersección de listas.

Lema 6.9 Se cumplen las siguientes propiedades para cualesquiera listas ℓ, ℓ_1, ℓ_2 y $x \in \mathbb{N}$.

1. $elem\ x\ (union\ \ell_1\ \ell_2)$ si y sólo si $elem\ x\ \ell_1 \vee elem\ x\ \ell_2$.
2. $elem\ x\ (inter\ \ell_1\ \ell_2)$ si y sólo si $elem\ x\ \ell_1 \wedge elem\ x\ \ell_2$.
3. Si $\ell_1 \sim \ell_2$ entonces $union\ \ell\ \ell_1 \sim union\ \ell\ \ell_2$.
4. Si $\ell_1 \sim \ell_2$ entonces $inter\ \ell\ \ell_1 \sim inter\ \ell\ \ell_2$.
5. $union\ \ell_1\ \ell_2 \sim union\ \ell_2\ \ell_1$.
6. $inter\ \ell_1\ \ell_2 \sim inter\ \ell_2\ \ell_1$.
7. Si $\ell_1 \sim \ell_2$ entonces $union\ \ell_1\ \ell \sim union\ \ell_2\ \ell$.
8. Si $\ell_1 \sim \ell_2$ entonces $inter\ \ell_1\ \ell \sim inter\ \ell_2\ \ell$.

Demostración.

1. La prueba es por inducción sobre ℓ_1 y se deja como ejercicio.
2. Si ℓ_2 es vacía entonces ambos lados de la equivalencia son falsos y por lo tanto la equivalencia es verdadera. Para los casos restantes la prueba es por inducción sobre ℓ_1 . El caso base es análogo a cuando ℓ_2 es vacía. El paso inductivo se deja al lector.
3. Supóngase que $\ell_1 \sim \ell_2$. Queremos demostrar que

$$\forall x \in \mathbb{N} \left(\text{elem } x (\text{union } \ell \ell_1) \leftrightarrow \text{elem } x (\text{union } \ell \ell_2) \right),$$

lo cual, usando la parte 1 de este mismo lema, equivale a

$$\forall x \in \mathbb{N} \left(\text{elem } x \ell \vee \text{elem } x \ell_1 \leftrightarrow \text{elem } x \ell \vee \text{elem } x \ell_2 \right),$$

lo cual es consecuencia inmediata de $\ell_1 \sim \ell_2$.

4. Análogo al punto anterior utilizando la parte 2 de este mismo lema.
5. Es consecuencia directa de la parte 1 de este lema, utilizando la conmutatividad del operador \vee .
6. Es consecuencia directa de la parte 2 de este lema, utilizando la conmutatividad del operador \wedge .
7. Supóngase que $\ell_1 \sim \ell_2$. Entonces utilizando las partes 3 y 5 de este mismo lema:

$$\text{union } \ell_1 \ell \sim \text{union } \ell \ell_1 \sim \text{union } \ell \ell_2 \sim \text{union } \ell_2 \ell.$$

8. Supóngase que $\ell_1 \sim \ell_2$. Entonces, utilizando las partes 4 y 6 de este mismo lema:

$$\text{inter } \ell_1 \ell \sim \text{inter } \ell \ell_1 \sim \text{inter } \ell \ell_2 \sim \text{inter } \ell_2 \ell. \quad \square$$

La siguiente proposición garantiza que nuestra definición de unión e intersección para conjuntos es correcta.

Proposición 6.10 Sean $\ell_1, \ell'_1, \ell_2, \ell'_2$ listas cualesquiera, tales que $\ell_1 \sim \ell'_1$ y $\ell_2 \sim \ell'_2$. Entonces

- i. $\text{union } \ell_1 \ell_2 \sim \text{union } \ell'_1 \ell'_2$.
- ii. $\text{inter } \ell_1 \ell_2 \sim \text{inter } \ell'_1 \ell'_2$.

Demostración.

- i. Por las partes 3 y 7 del lema 6.9 se tiene que

$$\text{union } \ell_1 \ell_2 \sim \text{union } \ell_1 \ell'_2 \sim \text{union } \ell'_1 \ell'_2.$$

- ii. Por las partes 4 y 8 del lema 6.9 se tiene que

$$\text{inter } \ell_1 \ell_2 \sim \text{inter } \ell_1 \ell'_2 \sim \text{inter } \ell'_1 \ell'_2. \quad \square$$

Finalmente podemos demostrar que nuestras operaciones son independientes del representante y, por lo tanto, están bien definidas.

Corolario 6.11 Las operaciones $\text{elem}C, \text{union}C, \text{inter}C$ están bien definidas. Es decir para cualesquiera conjuntos c_1, c'_1, c_2, c'_2 se cumple lo siguiente:

1. Si $c_1 = c'_1$ entonces $\text{elem}C \ x \ c_1 = \text{elem}C \ x \ c'_1$
2. Si $c_1 = c'_1$ y $c_2 = c'_2$ entonces $\text{union}C \ c_1 \ c_2 = \text{union}C \ c'_1 \ c'_2$
3. Si $c_1 = c'_1$ y $c_2 = c'_2$ entonces $\text{inter}C \ c_1 \ c_2 = \text{inter}C \ c'_1 \ c'_2$

Demostración.

1. Sean $c_1 = [\ell_1]$, $c'_1 = [\ell'_1]$. Como $c_1 = c'_1$ entonces $\ell_1 \sim \ell'_1$, de donde, por definición de \sim , se cumple que $\text{elem } x \ell_1 \leftrightarrow \text{elem } x \ell'_1$. Como ambos lados de la doble implicación tienen siempre el mismo valor, tenemos $\text{elem } C x c_1 = \text{elem } C x c'_1$.
2. Sean $c_1 = [\ell_1]$, $c'_1 = [\ell'_1]$, $c_2 = [\ell_2]$, $c'_2 = [\ell'_2]$. Como $c_1 = c'_1$ y $c_2 = c'_2$, entonces $\ell_1 \sim \ell'_1$ y $\ell_2 \sim \ell'_2$.
De donde, por la proposición anterior, se sigue que $\text{union } \ell_1 \ell_2 \sim \text{union } \ell'_1 \ell'_2$ y por lo tanto $[\text{union } \ell_1 \ell_2] = [\text{union } \ell'_1 \ell'_2]$. Es decir, $\text{union } C c_1 c_2 = \text{union } C c'_1 c'_2$.
3. Es análogo al caso anterior. □

Con esto queda demostrado formalmente que nuestra implementación para las operaciones de conjuntos es correcta. Como última observación queremos mencionar que la estructura base elegida, las listas, resulta ser inadecuada muchas veces en la práctica, puesto que la operación de pertenencia *elem* es muy ineficiente. Sin embargo, podemos elegir otra estructura base, como son los árboles binarios de búsqueda o árboles rojinegros, los cuales cuentan con una función de pertenencia eficiente, e implementar conjuntos utilizando exactamente la misma relación de equivalencia \sim . Todo esto indica que nuestra definición es modular, característica muy deseable en la programación de aplicaciones.

Ejercicios

- 6.4.1.- Sea $A = \mathbb{R}$, el conjunto de números reales. Definimos $R = \{(x, y) \mid x^2 = y^2\}$. Demuestre que R es una relación de equivalencia y encuentre las clases de equivalencia.
- 6.4.2.- Sea $R = \{(x, y) \mid x, y \in \mathbb{R}, [2x] = [2y]\}$ una relación definida en \mathbb{R} , donde $[x]$ se define como el mayor entero $i \in \mathbb{Z}$, tal que $i \leq x$. Por ejemplo $[2.3] = 2$, $[-3.5] = -4$.
 - a) Verifique que R sea una relación de equivalencia.
 - b) Determine las clases de equivalencia de $1/4$ y $1/2$.
 - c) Describa la partición de \mathbb{R} en clases de equivalencia.
- 6.4.3.- Sea S una relación definida en \mathbb{R} , $S = \{(x, y) \mid x, y \in \mathbb{R}, [3x] = [3y]\}$. Determine la partición de \mathbb{R} generada por la relación S .
- 6.4.4.- Demuestre que I_A y U_A (como se definieron en 6.6.4 y 6.6.5) son relaciones de equivalencia ¿Cuáles son las clases de equivalencia de I_A ? ¿Y las de U_A ?
- 6.4.5.- Cada una de las siguientes relaciones R sobre A no es una relación de equivalencia. En cada caso muestre mediante un ejemplo las propiedades que no se satisfacen.
 - a) $A = \mathbb{Z}$, xRy si y sólo si $x + y$ es impar.
 - b) $A = \mathbb{Q}$, xRy si y sólo si x/y es entero.
 - c) $A = \mathbb{Z}$, xRy si y sólo si $|x - y| \leq 5$.

d) $A = \mathbb{N}$, xRy si y sólo si $x \bmod 4 = y \bmod 4$ o $x \bmod 6 = y \bmod 6$.

e) $A = \mathbb{R}$, xRy si y sólo si $k < x/10 < k+1$ y $x \leq y/10 < k+1$, para algún $k \in \mathbb{Z}$.

6.4.6.- Sea $R \subseteq \mathbb{Q} \times \mathbb{Q}$ una relación sobre los números racionales definida como

aRb si y sólo si $|a - b| < 1$.

¿Es R una relación de equivalencia?

6.4.7.- La estructura \mathbb{Z}_n de números enteros módulo n , definida en el ejemplo 6.41, puede dotarse de operaciones aritméticas de suma y multiplicación mediante la operación correspondiente en los representantes de cada clase, es decir:

$$[a + b] = [a] + [b] \quad [a \cdot b] = [a] \cdot [b]$$

a) Muestre que estas operaciones están bien definidas; esto es, muestre que:

si $[a] = [a']$ y $[b] = [b']$

entonces $[a + b] = [a' + b']$ y $[a \cdot b] = [a' \cdot b']$

b) Elabora las tablas de suma y multiplicación para \mathbb{Z}_n para los casos

$n = 3, 4, 5, 6$ y 7 .

6.4.8.- Supongamos que tenemos un lenguaje de programación donde sólo se consideran los primeros k caracteres para distinguir identificadores. De esta manera, si $k = 4$ por ejemplo, el identificador `casaChica` es el mismo que `casaGrande`. Sean s y t cadenas, donde se distinguen mayúsculas y minúsculas, de tamaño arbitrario y denotemos con $|x|$ a la longitud o tamaño de una cadena x . Dado $k > 0$, sea R_k la relación sobre cadenas definida como sRt si y sólo si

- $s = t$ o bien
- $|s| > k$, $|t| > k$ y los primeros k caracteres coinciden.

Demuestre que R_k es una relación de equivalencia.

6.4.9.- Defina tres relaciones de equivalencia entre los estudiantes de su curso de Estructuras Discretas.

6.4.10.- Sea R la relación en el conjunto de todas las *URL* (o direcciones en la web) tal que xRy si y sólo si la página web en x es la misma que la página web en y . Muestre que R es una relación de equivalencia.

6.4.11.- ¿Cuáles de las siguientes colecciones de subconjuntos son particiones del conjunto de cadenas de bits de tamaño 8?

- a) El conjunto de cadenas de bits que empiezan con 1, el de las que empiezan con 00 y el de las que empiezan con 01.
- b) El conjunto de cadenas de bits que tienen como subcadena a 00, el conjunto de cadenas de bits que tienen como subcadena a 01, el conjunto de cadenas de bits que tienen como subcadena a 10 y el conjunto de cadenas de bits que tienen como subcadena a 11.

- c) El conjunto de cadenas de bits que terminan con 00, el conjunto de cadenas de bits que terminan con 01, el conjunto de cadenas de bits que terminan con 10, el conjunto de cadenas de bits que terminan con 11.
- d) El conjunto de cadenas de bits que terminan con 111, el conjunto de cadenas de bits que terminan con 011 y el conjunto de cadenas de bits que terminan con 00.
- e) El conjunto de cadenas de bits que tienen $3k$ unos, donde $k \geq 0$, el conjunto de cadenas de bits que tienen $3k + 1$ unos y el conjunto de cadenas de bits que tienen $3k + 2$ unos.

6.4.12.- Una partición P_1 es un *refinamiento* de la partición P_2 si cada conjunto en P_1 es subconjunto de algún conjunto en P_2 . Muestre que la partición formada por las clases de congruencia módulo 6 es un refinamiento de la partición formada por las clases de congruencia módulo 3.

6.4.13.- Sea R_8 la relación de equivalencia entre identificadores de un lenguaje de programación como la definimos en el ejercicio 6.4.8, con $k=8$. Supongamos que con el paso del tiempo las clases de equivalencia se definen con $k=31$ (generando la relación R_{31}). Demuestre que R_8 es un refinamiento de R_{31} .

6.4.14.- Considere la siguiente relación \sim sobre parejas de números naturales, es decir $R \subseteq \mathbb{N}^2 \times \mathbb{N}^2$: $(a, b) \sim (c, d)$ si y sólo si $a + d = c + b$

- a) Muestre que \sim es una relación de equivalencia.
- b) Calcule el conjunto cociente \mathbb{N}^2 / \sim .
- c) Muestre que si $a \geq 1$ entonces cada clase de equivalencia contiene exactamente uno de los pares $(0, a)$ o $(a, 0)$.
- d) Concluya que cada clase de equivalencia es de la forma $[(0, 0)]$, $[(0, a)]$ o bien $[(a, 0)]$, con $a \geq 1$.
- e) Definimos $\bar{0} = [(0, 0)]$, $\bar{a} = [(a, 0)]$, $-\bar{a} = [(0, a)]$. Muestre que con estas definiciones es posible definir las operaciones usuales de suma, producto y resta, por lo que podemos implementar a los números enteros mediante esta construcción. Es decir, podemos definir $\mathbb{Z} = \{\dots, -\bar{2}, -\bar{1}, \bar{0}, \bar{1}, \bar{2}, \dots\}$

6.4.15.- Sea R una relación binaria sobre A . Muestre que $I_A \cup \text{trans}(R \cup R^{-1})$ es una relación de equivalencia –recuerden que $\text{trans}(A)$ es la cerradura transitiva de A –.

6.4.16.- Sea R una relación binaria sobre A . Muestre que si R es reflexiva y simétrica entonces R^+ es una relación de equivalencia.

6.4.17.- Agregue las siguientes operaciones a la implementación de conjuntos mediante listas discutida en la sección 6.4.2. En cada caso debe mostrar que la operación está bien definida de manera análoga al corolario 6.11.

- a) *diffC* que calcula la diferencia de conjuntos.
- b) *incC* que verifica si un conjunto está contenido en otro.

- 6.4.18.- ¿Es posible definir la operación $compC$ que calcula el complemento de un conjunto? Justifique su respuesta.
- 6.4.19.- ¿Es posible definir las operaciones $potC$ que calcula la potencia de un conjunto y $prodC$ que calcula el producto cartesiano de dos conjuntos? Discuta qué estructuras y operaciones auxiliares son necesarias.
- 6.4.20.- ¿Qué modificaciones habría que hacer a la implementación de conjuntos de la sección 6.4.2 para implementar multiconjuntos? Recuerde que los multiconjuntos son una generalización de los conjuntos, donde no importa el orden de los elementos pero sí el número de veces que figuran en la estructura. Por ejemplo los multiconjuntos $\{ \{ 2, 3, 1, 2 \} \}$ y $\{ \{ 2, 3, 1, 3, 2 \} \}$ se consideran distintos y los multiconjuntos $\{ \{ 2, 1, 1, 4 \} \}$ y $\{ \{ 4, 1, 2, 1 \} \}$ son iguales.

6.5. Relaciones de orden

Las relaciones de orden están presentes todo el tiempo tanto en matemáticas como en la vida real. Escenarios donde se requiere contrastar pequeño contra grande o mejor contra peor; nociones de progresión, precedencia o preferencia; son todos ejemplos de orden. De manera más concreta, las siguientes afirmaciones involucran orden:

- Dos primos hermanos tienen un abuelo en común.
- La sucesión de planetas de acuerdo a su distancia desde el sol es: Mercurio, Venus, Tierra, ..., Neptuno.
- Ninguno de los conjuntos $\{1, 2, 4\}$, $\{2, 3, 5\}$ es subconjunto del otro, pero el conjunto $\{1, 2, 3, 4, 5\}$ contiene a los dos.
- $0 < 1$, $1 < 10^2$
- Un teniente es superior a un cabo.
- En el código ASCII se cumple que “A” es menor que “a” (va antes).
- “Sánchez” es mayor que “Olguín” de acuerdo al directorio telefónico.
- Si a, b son números reales distintos, entonces sucede que $a < b$ o bien $b < a$.

El orden no es una propiedad intrínseca de un objeto, sino que concierne a la comparación de dos objetos y, por lo tanto, es adecuado representarlo como una relación binaria sobre un conjunto dado, como los números naturales, enteros, reales, entre otros, los caracteres, las personas, los planetas, los apellidos, los rangos militares, etcétera.

¿Qué propiedades debe cumplir una relación binaria para ser considerada un orden? Un razonamiento intuitivo nos permite concluir, por ejemplo, que si Marte está más cerca de la tierra que Júpiter y Júpiter más cerca que Neptuno, entonces también Marte está más cerca que Neptuno. Es decir, un orden debe cumplir la llamada propiedad de transitividad. En otro contexto, si un teniente es superior a un cabo entonces no sucede que un cabo sea superior a un teniente. Esto significa que el único caso cuando suceden ambas aRb y bRa ,

en este caso en la jerarquía militar, es cuando son el mismo nombramiento, es decir $a = b$. Un orden debe cumplir entonces con la propiedad de antisimetría.

Adicionalmente, las relaciones de orden pueden ser de dos clases, *estrictas* o *no estrictas*. Se llaman estrictas cuando en el orden no hay un objeto relacionado consigo mismo. Por ejemplo, una persona no es más alta que ella misma, ni un número es menor que él mismo. Sin embargo en Matemáticas sí consideramos órdenes no estrictos como \leq entre números. Las propiedades que modelan estas condiciones son la antirreflexividad y la reflexividad respectivamente.

Muchos conjuntos presentan un orden natural entre sus elementos, como es el caso de las letras del alfabeto; el orden impuesto por los códigos de los caracteres en un sistema o lenguaje de programación (los códigos ASCII o UTF8 por ejemplo); los números naturales (\mathbb{Z}), enteros (\mathbb{N}) y reales (\mathbb{R}); los miembros de una familia; las palabras en un diccionario o los nombres en un directorio telefónico (estas últimas dependen del orden dado a los caracteres).

Con estas ideas en mente enunciamos la definición formal de relación de orden.

Definición 6.13 Sean A un conjunto y $R \subseteq A \times A$ una relación binaria sobre A . Decimos que R es una relación **de orden parcial** si y sólo si R cumple con ser reflexiva, antisimétrica y transitiva. En tal caso decimos que A es un conjunto parcialmente ordenado⁶ por R o que R ordena parcialmente al conjunto A .

Los conjuntos parcialmente ordenados se representan, en general, como una pareja con el conjunto al que se aplica la relación y la relación particular; por ejemplo (\mathbb{N}, \leq) , (\mathbb{Z}, \leq) , (\mathbb{R}, \geq) , $(\{a, b, \dots, z\}, \leq)$ son algunos de estos casos. En adelante denotamos a una relación de orden sobre un conjunto A con (A, \leq) , donde \leq es la relación particular R de orden parcial entre los elementos. Además si $x \leq y$ entonces diremos que x es menor o igual que y .

En lo que sigue damos algunos ejemplos de órdenes parciales.

Ejemplo 6.45. Todos los órdenes numéricos usuales en matemáticas son órdenes parciales, por ejemplo (\mathbb{N}, \leq) , (\mathbb{Z}, \leq) , (\mathbb{R}, \leq) pero también (\mathbb{N}, \geq) , (\mathbb{Z}, \geq) , (\mathbb{R}, \geq) .

Ejemplo 6.46. Si A es un conjunto cualquiera entonces el orden discreto para A se define como (A, \leq) donde \leq es la relación de igualdad. Es decir, definimos $x \leq y$ si y sólo si $x = y$. Observe que se trata de un orden puesto que la igualdad es reflexiva y transitiva. Además resulta ser antisimétrica pues si $a = b$ y $b = a$ entonces $a = a$.

Ejemplo 6.47. Sea $S = \{S_i \mid i \in I\}$ una colección de conjuntos arbitrarios y definimos $S_i \leq S_j$ si $S_i \subseteq S_j$. Esta relación es un orden parcial. Como $A \subseteq A$ entonces \leq es reflexiva. Si $A \leq B$ y $B \leq A$ quiere decir que $A \subseteq B$ y $B \subseteq A$, por lo que se cumple que $A = B$ y la propiedad de antisimetría es válida. Por último, si $A \subseteq B$ y $B \subseteq C$ tenemos $A \subseteq C$ por lo que también cumple la propiedad de transitividad.

⁶En inglés *poset* de Partially Ordered Set

Ejemplo 6.48. Sea $A = \mathbb{Z}^+$ el conjunto de números enteros positivos y definimos la relación $R = \{(a, b) \mid a, b \in \mathbb{Z}^+, a \text{ divide a } b\}$ (denotada usualmente por $a \mid b$). Recordamos que $a \mid b$ si y sólo si $b = k \cdot a$ para alguna $k \in \mathbb{Z}^+$. Verificamos las tres propiedades de orden parcial.

1. **Reflexividad:** $a = 1 \cdot a$ por lo que $\forall a \in \mathbb{Z}^+, a \mid a$.
2. **Antisimetría:** Supongamos $a \mid b$ y $b \mid a$. Entonces $b = k \cdot a$ para alguna k , puesto que $a \mid b$. Además, como $b \mid a$ entonces hay un m tal que $a = m \cdot b$. Luego entonces, $b = k \cdot a = k \cdot (m \cdot b)$; por lo que $b = (k \cdot m) \cdot b$ lo cual implica que $k \cdot m = 1$ y por lo tanto $k = m = 1$ dado que se trata de enteros positivos. Finalmente se concluye que $a = m \cdot b = 1 \cdot b = b$.
3. **Transitividad:** Supongamos $a \mid b$ y $b \mid c$. La primera pareja nos dice que $b = k \cdot a$. La segunda pareja implica que $c = m \cdot b$. De esto tenemos $c = m \cdot (k \cdot a) = (m \cdot k) \cdot a$. Pero esta es la definición de que $a \mid c$.

Sigue un ejemplo más elaborado de un orden parcial.

Ejemplo 6.49. Sea $A = \{s \mid s \text{ es una cadena de ceros y unos de tamaño } 5\}$, es decir, A es el conjunto de cadenas que van del 00000 al 11111. Por ejemplo, 01010, 11001, 00010 $\in A$, así como 10001, pero 0, 10, 110, 0101 $\notin A$. Notemos que A es un conjunto finito que tiene 2^5 elementos. Dada una cadena $a \in A$, denotamos con $a[i]$ al dígito binario en la posición i , contando de izquierda a derecha empezando en 1. Por ejemplo, si $a = 10010$ entonces $a[3] = 0$ y $a[1] = 1$. Denotamos con $a[i \dots j]$, $1 \leq i \leq j \leq 5$, a la subcadena que va de la posición i a la posición j inclusive; si $j < i$ entonces la cadena no incluye a ningún símbolo. Por ejemplo, si $a = 11010$ entonces $a[1 \dots 3] = 110$, $a[2 \dots 4] = 101$ y $a[3 \dots 1]$ es la cadena vacía. Definimos la relación (A, \leq) de la siguiente manera: $s \leq t$ si y sólo si:

- (i) $s = t$ o bien
- (ii) existe $1 \leq j < 5$ tal que $s[1 \dots j] = t[1 \dots j]$ y $s[j+1] < t[j+1]$.

donde para $a, b \in \{0, 1\}$, definimos $a < b$ si y sólo si $a = 0$ y $b = 1$.

Veamos que (A, \leq) es, en efecto, un orden parcial.

Reflexividad: Se cumple $s \leq s$ por el inciso (i).

Antisimetría: Supongamos $s \leq t$ y $t \leq s$. La primera opción es que se cumpla el inciso (i), en cuyo caso $s = t$ y se cumple la antisimetría.

Supongamos ahora que $s \neq t$. Entonces, para alguna $j < 5$, $s[1 \dots j] = t[1 \dots j]$ y $s[j+1] < t[j+1]$.

Pero como también $t \leq s$, entonces para alguna $k < 5$, $s[1 \dots k] = t[1 \dots k]$ y $t[k+1] < s[k+1]$. Por las propiedades de j y k es fácil ver que necesariamente $j = k$. Pero entonces sucedería al mismo tiempo $s[j+1] < t[j+1]$ y $t[j+1] < s[j+1]$, lo cual es una contradicción por la definición de $<$. De manera que debe suceder $s = t$.

Transitividad: Supongamos $s \leq t$ y $t \leq u$. Si $s = t$ entonces $s \leq u$ y se cumple la transitividad. Lo mismo sucede si $t = u$.

Supongamos entonces que $s \neq t$ y $s \neq u$. Como $s \neq t$ y $s \leq t$, para alguna $j < 5$, sucede que $s[1 \dots j] = t[1 \dots j]$ y $s[j+1] < t[j+1]$. Análogamente, como $t \leq u$,

para alguna $k < 5$ se cumple que $t[1 \dots k] = u[1 \dots k]$ y $t[k + 1] < u[k + 1]$. Si $j = k$ entonces $s[1 \dots j] = u[1 \dots j]$ y $s[j + 1] < u[j + 1]$ por lo que podemos concluir que $s \leq u$. Si $j \neq k$ entonces supongamos, sin pérdida de generalidad, que $k < j$. Así se sigue que $s[1 \dots k] = t[1 \dots k]$ y por lo tanto $s[1 \dots k] = u[1 \dots k]$. Además como $k < j$ entonces $k + 1 \leq j$ lo cual implica que $s[k + 1] = t[k + 1]$, de donde $s[k + 1] < u[k + 1]$ y por lo tanto $s \leq u$.

Dado un orden parcial cualquiera (A, \leq) existen otros órdenes parciales asociados a él, que definimos a continuación.

Definición 6.14 Sea (A, \leq) un orden parcial. La relación de orden \leq genera las siguientes tres relaciones binarias:

- $x < y$ si y sólo si $x \leq y$ y $x \neq y$.
- $x \geq y$ si y sólo si $y \leq x$.
- $x > y$ si y sólo si $x \geq y$ y $x \neq y$.

Si $x < y$ decimos que x es menor que y o que x está por debajo de y ; si $x \geq y$ decimos que x es mayor o igual que y y si $x > y$ decimos que x es mayor o está por arriba de y . El lector debe verificar que las relaciones recién definidas a partir de \leq también son órdenes parciales en A . En particular, un orden parcial (A, \leq) se conoce también como orden parcial no estricto o suave, dado que es reflexivo. Es decir, siempre sucede que $x \leq x$. Por otro lado, los órdenes de la forma $(A, <)$ son antirreflexivos, es decir nunca sucede que $x < x$ y se conocen como órdenes parciales estrictos. También puede observarse que el orden \geq ($>$) es simplemente la relación inversa del orden \leq ($<$).

El lector se preguntará el porqué del adjetivo *parcial* en nuestra definición de orden. Esto se debe a que dados dos elementos en un conjunto parcialmente ordenado, no necesariamente pueden compararse, en el sentido de la definición que sigue.

Definición 6.15 Sea (A, \leq) un conjunto parcialmente ordenado. Decimos que $a, b \in A$ son *comparables* si $a \leq b$ o bien $b \leq a$. Si no es así, decimos que a y b son *incomparables*.

Muchos órdenes que surgen de manera natural en distintos ámbitos son parciales pues tienen elementos incomparables. Por ejemplo, si el orden \leq modela a la relación de parentesco en personas, claramente hay elementos incomparables, puesto que no todos somos de la misma familia.

Obsérvese que debido a la existencia de elementos incomparables, si sucede $x \not\leq y$, entonces no podemos concluir, en general, $y \leq x$, puesto que x e y podrían ser incomparables.

El siguiente ejemplo muestra una aplicación importante de los órdenes parciales.

Ejemplo 6.50. Pensemos en el proceso de construcción de una casa dada mediante una serie de actividades a realizarse de acuerdo a ciertos requerimientos previos, definidas en la tabla 6.3 en la siguiente página.

Intuitivamente se observa que existe un orden entre las actividades dictado por los requerimientos previos de la última columna de la tabla. Si definimos al conjunto de activi-

dades como $\mathcal{A} = \{A, \dots, N\}$ entonces podemos definir formalmente este orden mediante la relación $\leq \subseteq \mathcal{A} \times \mathcal{A}$, definida de la siguiente manera:

$$\leq = \{(a, b) \mid \begin{array}{l} \text{la actividad } a \text{ es la misma que la actividad } b \text{ o bien} \\ \text{es un requerimiento de la actividad } b \text{ según la tabla} \end{array}\}.$$

Podemos ver que \leq es una relación de orden parcial pues cumple:

Reflexividad: $a \leq a$ ya que todas las actividades son iguales a sí mismas.

Antisimetría: Si $a \leq b$ y $b \leq a$, entonces necesariamente a y b son la misma actividad puesto que no puede ser que a sea requerimiento para b y al mismo tiempo que b sea requisito para a , lo cual se ve por inspección de la tabla y porque intuitivamente si tal cosa sucediera la construcción nunca se podría llevar a cabo.

Transitividad: Si la tarea a tiene que realizarse antes que la b , y la b , a su vez, tiene que realizarse antes que la c , es claro que la tarea a tiene que realizarse antes que la c .

Tabla 6.3. Ejemplo 6.50: Relación de orden parcial

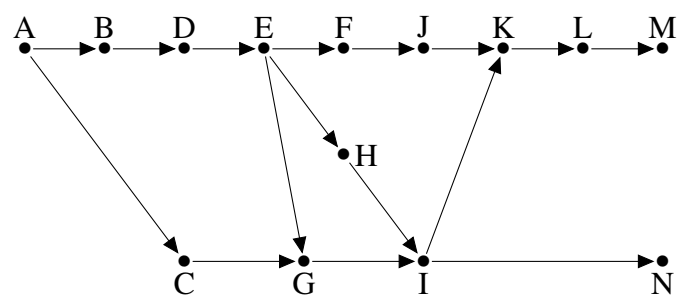
Tarea	Tiempo en días	Pasos previos
A Preparación del terreno	4	—
B Cimientos	6	A
C Tuberías y servicios	3	A
D Columnas	10	B
E Techo	5	D
F Ventanas	2	E
G Plomería	4	C, E
H Electricidad	3	E
I Impermeabilización	2	G, H
J Muros	6	F
K Enyesado	5	I, J
L Limpieza y pintura	3	K
M Pisos y detalles	4	L
N Inspección	10	I

Independientemente del tiempo que le lleve a cada tarea, podemos escribir la relación en la tabla 6.4, donde el renglón indica la tarea necesaria y la columna indica quién la necesita. Esta tabla, que representa a esta relación, se encuentra en la página que sigue.

En este caso podemos ver que la tabla que describe a la relación es poco densa (tiene pocas entradas) por lo que esta representación resulta inconveniente. Veamos la descripción

de la relación con la gráfica correspondiente en la figura 6.6.

Figura 6.6. Dependencias en la ejecución de las tareas



Se observa que la representación gráfica es más fácil de manejar por lo que resulta más adecuada que la tabla.

Tabla 6.4. Dependencia en la ejecución de las tareas

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
A		✓	✓											
B				✓										
C							✓							
D					✓									
E						✓	✓	✓						
F										✓				
G									✓					
H									✓					
I											✓			✓
J											✓			
K												✓		
L													✓	
M														
N														

Una vez definido formalmente el orden de actividades y contando con una representación gráfica del mismo, nos gustaría resolver el siguiente problema. Supongamos que se debe realizar la construcción de la casa pero por razones de presupuesto sólo se puede contratar a una compañía modesta que cuenta únicamente con un pequeño equipo de trabajadores. Asimismo, esta cuadrilla puede realizar una sola tarea a la vez. La pregunta es: ¿de qué manera deben organizarse las tareas de manera que se respeten los prerrequisitos proporcionados? Obsérvese que la solución debe ser una lista de actividades en un orden

secuencial (una actividad tras otra) que respete el orden formal recién definido. Además en este problema no nos interesa el asunto de minimizar el tiempo de construcción, por lo que la segunda columna de la tabla no es relevante.

Resolveremos más adelante esta cuestión, pero antes presentamos una forma más simple de representar gráficamente órdenes parciales.

6.5.1. Diagramas de Hasse

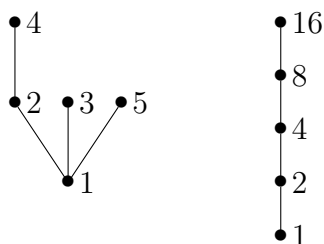
Los diagramas de Hasse nos sirven para representar gráficamente las relaciones de orden parcial; son similares a la gráfica que mostramos en el ejemplo 6.50.

Supongamos que tenemos una relación de orden parcial R sobre un conjunto S . Para construir el diagrama de Hasse representamos a cada elemento de S como un punto en el diagrama. Dibujamos una arista (o línea) de x a y siempre que $x \leq y$ y no exista $s \in S$ tal que $x \leq s$ y $s \leq y$ (sólo dibujamos las relaciones de orden “directas”). Adicionalmente, organizamos el diagrama de tal manera que cuando una línea va de x a y , x está en una región abajo de aquella en la que está y . Por último, como el orden de la relación está dado visualmente, con unir puntos con líneas es suficiente (no requerimos flechas).

El orden de la relación es muy claro pues siempre va del nodo que está más bajo al nodo que está más alto.

Veamos algunos ejemplos de diagramas de Hasse para órdenes parciales.

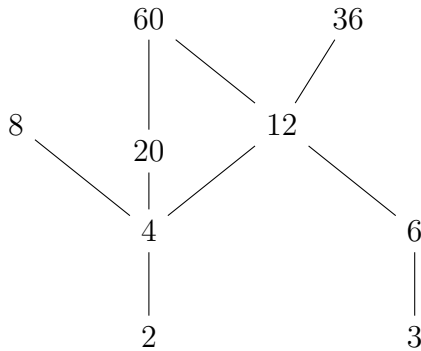
Ejemplo 6.51. Consideremos los conjuntos de enteros $S_1 = \{1, 2, 3, 4, 5\}$ y $S_2 = \{1, 2, 4, 8, 16\}$ con la relación de divisibilidad. Los diagramas de Hasse correspondientes se muestran a continuación:



Un ejemplo más elaborado con la relación de divisibilidad se encuentra enseguida.

Ejemplo 6.52. Sea $A = \{2, 3, 4, 6, 8, 12, 16, 20, 60\}$. El diagrama de Hasse para el orden parcial de divisibilidad $(A, |)$ es el que se ve en la figura 6.7 que se encuentra en la siguiente página.

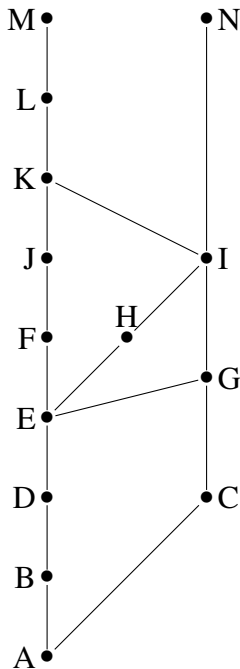
Figura 6.7. Orden parcial bajo la relación de divisibilidad



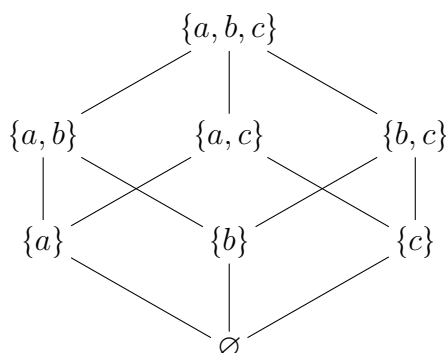
Nótese por ejemplo que, aún cuando $6 \mid 60$ y $6 \mid 36$, esto no se muestra en el Diagrama de Hasse correspondiente ya que existe otro entero, el 12, que permite concluir aquellas relaciones mediante transitividad.

Ejemplo 6.53. El diagrama de Hasse del ejemplo 6.50, que se encuentra en la figura 6.8, en la siguiente página, resulta ser una rotación de 90° de la gráfica de dicho ejemplo.

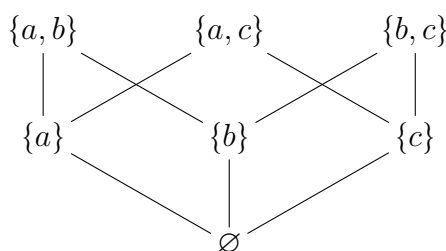
Figura 6.8. Diagrama de Hasse para el orden parcial del ejercicio 6.50



Ejemplo 6.54. Un ejemplo clásico de orden parcial es la relación que existe entre un conjunto y sus subconjuntos, dada por la relación \subseteq . Sea $S = \{a, b, c\}$. El Diagrama de Hasse para la relación de contención \subseteq se encuentra en la figura 6.9.

Figura 6.9. Diagrama de Hasse para subconjuntos del conjunto $\{a, b, c\}$ 

Ejemplo 6.55. Podemos hacer un ejercicio similar al del ejemplo anterior, pero la relación va a ser la de subconjuntos propios. El Diagrama de Hasse correspondiente se encuentra en la figura 6.10 de la página siguiente.

Figura 6.10. Diagrama de Hasse para subconjuntos propios de $\{a, b, c\}$ 

Se observa que los puntos extremos en un diagrama de Hasse, por ejemplo A hasta abajo y M, N hasta arriba en el ejemplo anterior, cumplen ciertas propiedades particulares en el orden, por ejemplo no hay ninguna actividad $X \neq A$ tal que $X \leq A$. En la siguiente sección estudiamos a esta clase de elementos y sus propiedades de manera general.

Ejercicios

6.5.1.- Sea (A, \leq) un orden parcial. Definimos la relación ternaria E sobre A como sigue:

$$(a, x, b) \in E \text{ si y sólo si } a \leq x \leq b \text{ o } b \leq x \leq a$$

Demuestre las siguientes propiedades para cualesquiera $a, b, c, d, x \in A$:

- Si $(a, x, b) \in E$ entonces $(b, x, a) \in E$.
- Si $(a, x, b) \in E$ y $(a, b, x) \in E$ entonces $x = b$.
- Si $(a, x, b) \in E$ y $(a, y, x) \in E$ entonces $(a, y, b) \in E$.
- Si $(a, x, b) \in E$ y $(x, b, y) \in E$ y $x \neq b$ entonces $(a, b, y) \in E$.
- Si $(a, b, c) \in E$ y $(a, c, d) \in E$ entonces $(b, c, d) \in E$.

- 6.5.2.- Sean (A, \leq) un orden parcial y x_1, x_2, \dots, x_n , $x_i \in A$, $i = 1, \dots, n$. Demuestre que si $x_1 \leq x_2 \leq x_3 \leq \dots \leq x_{n-1} \leq x_n \leq x_1$, entonces $x_1 = x_2 = \dots = x_n$.
- 6.5.3.- Verifique que las relaciones dadas en la definición 6.14 son órdenes parciales.
- 6.5.4.- Dada una relación R en $A \times B$, demuestre que si R es una relación de orden parcial entonces también lo es R^{-1} .
- 6.5.5.- Demuestre que (\mathbb{N}, \leq) es un orden parcial donde $a \leq b$ si y sólo si a es múltiplo de b .
- 6.5.6.- Demuestre que (\mathbb{Z}, \leq) es un orden parcial donde $x \leq y$ si y sólo si sucede una de las siguientes dos condiciones:
- $x - y$ es impar y x es par
 - $x - y$ es par y $x \leq y$ (aquí \leq es el orden usual en \mathbb{Z}).
- 6.5.7.- Sea X un conjunto finito y para $A, B \subseteq X$ definimos la relación \leq como $A \leq B$ si y sólo si $|A| \leq |B|$. Es decir $A \leq B$ si y sólo si A tiene menor o igual cardinalidad (número de elementos) que B . Discuta si \leq define un orden parcial sobre el conjunto potencia $\mathcal{P}(X)$.
- 6.5.8.- Sea A un conjunto arbitrario de personas y sea E una relación sobre A , tal que pEq si y sólo si la edad de p es menor o igual que la edad de q . ¿Es E un orden parcial? Justifique.
- 6.5.9.- Sea $A \subset \mathbb{N}$, definido como $A = \{n \in \mathbb{N} \mid n \leq 29, n = 2k \vee n = 3k \text{ para alguna } k\}$.
- Lista los elementos de A .
 - Haz el diagrama de Hasse correspondiente a $(A, |)$, es decir, el orden dado por la relación de divisibilidad.
- 6.5.10.- Tomemos el conjunto A , que consiste de las cadenas de tres dígitos binarios, y definimos $s \leq t$ de la misma manera que en el ejemplo 6.49. Dibuja el diagrama de Hasse para esta relación de orden.
- 6.5.11.- Sean $A = \{1, 2, 3, 4\}$ y $B = \{X \subseteq A \mid 2 \in X \text{ y } 3 \notin X\}$. Dibuja el diagrama de Hasse del conjunto B con respecto a los órdenes parciales \subseteq y \supseteq .
- 6.5.12.- Sean $(A, \leq_A), (B, \leq_B)$ dos órdenes parciales. El producto cartesiano $A \times B$ puede ordenarse mediante el orden \leq definido como sigue: si $a_1, a_2 \in A$ y $b_1, b_2 \in B$, entonces
- $$(a_1, b_1) \leq (a_2, b_2) \text{ si y sólo si } a_1 \leq_A a_2 \text{ o bien } a_1 = a_2 \text{ y } b_1 \leq_B b_2$$
- Demuestre que esta definición, en efecto, genera un orden parcial sobre $A \times B$. Este orden se conoce como *orden lexicográfico*.
- 6.5.13.- Sean $(A, \leq_A), (B, \leq_B)$ dos órdenes parciales tales que $A \cap B = \emptyset$. La unión $A \cup B$ puede ordenarse mediante el orden \leq definido como sigue:
- Si $x, y \in A$ entonces $x \leq y$, si $x \leq_A y$.
 - Si $x, y \in B$ entonces $x \leq y$, si $x \leq_B y$.

Demuestre que esta definición, en efecto, genera un orden parcial sobre $A \cup B$. Este orden se conoce como *suma* de A y B .

6.5.14.- Sean $(A, \leq_A), (B, \leq_B)$ dos órdenes parciales tales que $A \cap B = \emptyset$. La unión $A \cup B$ puede ordenarse mediante el orden \leq definido como sigue:

- Si $x, y \in A$ entonces $x \leq y$, si $x \leq_A y$.
- Si $x, y \in B$ entonces $x \leq y$, si $x \leq_B y$.
- Si $x \in A$ y $y \in B$ entonces $x \leq y$.

Demuestre que esta definición, en efecto, genera un orden parcial sobre $A \cup B$. Este orden se conoce como *suma lineal* de A y B .

6.5.15.- Sean $(A, \leq_A), (B, \leq_B)$ dos órdenes parciales. Muestre que la siguiente relación es un orden parcial sobre $A \times B$, definida como sigue: Si $a_1, a_2 \in A$ y $b_1, b_2 \in B$, entonces

$$(a_1, b_1) \leq (a_2, b_2) \text{ si y sólo si } b_1 <_B b_2 \text{ o bien } b_1 = b_2 \text{ y } a_1 \leq_A a_2$$

Esta relación se conoce como el *orden antilexicográfico*.

6.5.16.- De acuerdo al orden lexicográfico sobre $\mathbb{N} \times \mathbb{N}$ generado por el orden usual de los naturales, diga cuáles son la parejas (a, b) que cumplen con $(a, b) \leq (4, 2)$. Conteste lo mismo para el orden antilexicográfico.

6.5.17.- Sean $(A, \leq_A), (B, \leq_B)$ dos órdenes parciales. El producto cartesiano $A \times B$ puede ordenarse mediante el orden \leq definido como sigue: Si $a_1, a_2 \in A$ y $b_1, b_2 \in B$, entonces

$$(a_1, b_1) \leq (a_2, b_2) \text{ si y sólo si } a_1 \leq_A a_2 \text{ y } b_1 \leq_B b_2$$

Demuestre que esta definición, en efecto, genera un orden parcial sobre $A \times B$. Este orden se conoce como *orden de coordenadas*.

6.5.18.- De acuerdo al orden de coordenadas sobre $\mathbb{N} \times \mathbb{N}$ generado por el orden usual de los naturales, diga cuáles son la parejas (a, b) que cumplen con $(a, b) \leq (4, 2)$.

6.5.19.- Sean A el conjunto de divisores del 28 y B el conjunto de divisores del 42. Construya los órdenes lexicográfico, antilexicográfico y de coordenadas para $A \times B$ y para $B \times A$.

6.5.20.- Dibuja los diagramas de Hasse para los órdenes lexicográfico, antilexicográfico y de coordenadas del producto $S_1 \times S_2$, donde S_1 y S_2 son los órdenes del ejemplo 6.51.

6.5.21.- Los órdenes lexicográfico, antilexicográfico y de coordenadas pueden generalizarse para un producto cartesiano arbitrario $A_1 \times A_2 \times \dots \times A_n$. Redacte las definiciones formales.

6.5.22.- Considere el orden parcial usual (\mathbb{N}, \leq) . Verifique si los siguientes son órdenes parciales sobre $\mathbb{N} \times \mathbb{N}$.

- a) $(a, b) \leq (c, d)$ si y sólo si $a \leq c$
- b) $(a, b) \leq (c, d)$ si y sólo si $a \leq c$ y $b \geq d$

6.6. Elementos extremos

En algunos órdenes parciales existen ciertos elementos que cumplen propiedades específicas en la relación de orden. Por ejemplo el 0 en el orden (\mathbb{N}, \leq) o X en el orden $(\mathcal{P}(X), \subseteq)$. Estos son algunos ejemplos de los llamados elementos extremos que definiremos enseguida.

Definición 6.16 Sean $\langle A, \leq \rangle$ un orden parcial, $P \subseteq A$ y $m \in A$ (al ser $P \subseteq A$, P hereda el orden dado por \leq en A , por lo que (P, \leq) es también un orden parcial: para cualesquiera $p, q \in P$, mantienen entre ellos la misma relación que tienen en A ; en adelante daremos por hecho que P hereda el orden definido en A).

- m es un elemento mínimo de P si y sólo si $m \in P$ y para toda $p \in P$, $m \leq p$.
- m es un elemento máximo de P si y sólo si $m \in P$ y para toda $p \in P$, $p \leq m$.
- m es un elemento minimal de P si y sólo si $m \in P$ y no existe $p \in P$ tal que $p < m$.
- m es un elemento maximal de P si y sólo si $m \in P$ y no existe $p \in P$ tal que $m < p$.

Decimos que m es un *elemento extremo* de P si cumple alguna de las condiciones anteriores.

Obsérvese que estas definiciones dependen del orden parcial particular. Si este orden cambia, también cambiarán los elementos extremos. Más aún, un conjunto P puede o no tener alguna de las clases de elementos extremos bajo un determinado orden.

Es importante darse cuenta de la diferencia entre las nociones de máximo/maximal y mínimo/minimal. El elemento $m \in P$ es máximo de P si todos los elementos de P son menores o iguales que m ($\forall x \in P (x \leq m)$). Por otra parte $m \in P$ es maximal de P si P no contiene elementos estrictamente mayores o arriba de m ($\neg \exists x \in P (m < x)$). La misma diferencia está presente entre mínimo y minimal.

Veamos algunos ejemplos.

Ejemplo 6.56. Si $A = \{2, 4, 5, 10, 12, 20, 25\}$ y el orden es $\langle A, | \rangle$ (la divisibilidad), entonces podemos observar que A tiene dos elementos minimales que son 2 y 5; además 12, 20 y 25 son elementos maximales. Este ejemplo muestra que los elementos minimales o maximales no tienen que ser únicos.

Ejemplo 6.57. En el orden parcial $\langle \mathcal{P}(X), \subseteq \rangle$, donde X es un conjunto cualquiera, el mínimo es el conjunto vacío (\emptyset) y el máximo es el conjunto mismo (X). Si consideramos al conjunto $P = \mathcal{P}(X) \setminus \{\emptyset\}$, donde X tiene al menos dos elementos, entonces P no tiene un mínimo y los minimales son los conjuntos unitarios $\{p\}$ con $p \in P$. Si X es finito con $n \geq 2$ elementos y $P = \mathcal{P}(X) \setminus \{X\}$, entonces P no tiene máximo y los maximales son todos los subconjuntos de P con $n - 1$ elementos.

Ejemplo 6.58. En el orden parcial $\langle \mathbb{N}^+, | \rangle$ el conjunto no tiene máximo y el mínimo es 1. Si $P = \mathbb{N} \setminus \{1\}$ entonces P no tiene mínimo y los minimales son todos los números primos. Si $P = \mathbb{N}$ entonces P no tiene mínimo con respecto al orden de divisibilidad $|$, pero su

máximo es 0 puesto que para todo $n \in \mathbb{N}$, $0 \mid n$.

Ejemplo 6.59. Sea A un conjunto cualquiera. En el orden discreto $\langle A, = \rangle$ todos los elementos $x \in A$ son minimales y maximales.

Ejemplo 6.60. En el orden parcial de los enteros $\langle \mathbb{Z}, \leq \rangle$ no hay elementos extremos de ninguna clase.

Sea P un orden parcial definido para los elementos de A . Si se cuenta con el diagrama de Hasse \mathcal{H} bajo P , es sencillo encontrar los elementos extremos observando lo siguiente:

- Un elemento m es minimal si no hay nadie por debajo de él en \mathcal{H} .
- Un elemento m es maximal si no hay nadie por arriba de él en \mathcal{H} .
- Un elemento m es mínimo si esta por debajo de todos los elementos en \mathcal{H} .
- Un elemento m es máximo si esta por arriba de todos los elementos en \mathcal{H} .

Se invita al lector a obtener los elementos extremos de los órdenes de la sección 6.5.1, a partir de los diagramas de Hasse, siguiendo la observación anterior.

Terminamos esta sección con algunas propiedades importantes de los elementos extremos en un orden parcial $\langle A, \leq \rangle$ cualquiera.

Proposición 6.12 Si A bajo \leq tiene un elemento mínimo o máximo, entonces éste es único.

Demostración.

Supongamos que hay dos elementos mínimos, $m_1, m_2 \in A$. Como m_1 es mínimo y $m_2 \in A$, por definición tenemos $m_1 \leq m_2$. Ahora bien, como m_2 también es mínimo y $m_1 \in A$, de la misma manera $m_2 \leq m_1$. Sin embargo, la antisimetría de \leq implica que $m_1 = m_2$, de donde no hay dos mínimos en A bajo P . El caso para elementos máximos es análogo. □

A la luz de la proposición anterior podemos denotar al mínimo de un conjunto X con $\min X$ y al máximo de X con $\max X$.

Proposición 6.13 Si $m_1, m_2 \in A$ bajo P son dos elementos minimales (maximales) distintos, entonces son incomparables.

Demostración.

Sean m_1, m_2 maximales tales que $m_1 \neq m_2$. Es fácil mostrar que si m es un maximal cualquiera y $x \in A$ es tal que $m \leq x$, entonces $m = x$. De aquí se sigue que $m_1 \leq m_2$ no puede suceder puesto que $m_1 \neq m_2$ y m_1 es maximal. Similarmente, es imposible que $m_2 \leq m_1$ puesto que m_2 es maximal. Por lo tanto m_1 y m_2 son incomparables. El caso para elementos minimales es análogo. □

Los elementos extremos, en particular los minimales, nos servirán para resolver el problema de la construcción de la casa en el ejemplo 6.50.

6.6.1. Órdenes totales y ordenamiento topológico

Los órdenes totales son una clase especial de órdenes parciales, que resultan de utilidad para lograr que una serie de elementos se ordenen secuencialmente.

Definición 6.17 Decimos que el orden parcial (A, \leq) es un *orden total* o *lineal* si y sólo si para cualesquiera $x, y \in A$ se cumple $x \leq y$ o bien $y \leq x$.

Obsérvese que, por definición, todo orden total es un orden parcial mas no al revés. El adjetivo *total* se refiere a que la propiedad de comparar es total en el sentido de que cualesquiera dos elementos del orden son comparables. Los órdenes totales también se conocen como órdenes lineales o *cadena*s debido a que su diagrama de Hasse es una línea recta o cadena de puntos.

Algunos ejemplos de órdenes totales son los órdenes numéricos usuales como (\mathbb{N}, \leq) , (\mathbb{R}, \leq) , etcétera. Por otra parte, los órdenes de divisibilidad $(X, |)$ y los órdenes de conjuntos (X, \subseteq) casi nunca son totales.

Es claro que en un orden total no hay elementos incomparables. Más aún, utilizando el orden estricto es posible dar más información acerca de la relación de orden entre dos elementos, de acuerdo al siguiente resultado.

Proposición 6.14 (Tricotomía) Sean (A, \leq) un orden total y $x, y \in A$. Entonces sucede una y sólo una de las siguientes condiciones:

1. $x = y$.
2. $x < y$.
3. $y < x$.

Demostración.

Si $x = y$ entonces no puede suceder que $x < y$, ni tampoco que $y < x$, puesto que $<$ es antirreflexiva. Supongamos ahora que $x \neq y$. Como el orden es total, sucede que $x \leq y$ o bien $y \leq x$. Supongamos, sin pérdida de generalidad, que $x \leq y$. En tal caso, como $x \neq y$, si se cumple $x < y$, no puede ser que también suceda $y < x$, puesto que si así fuera, entonces también $y \leq x$ y por antisimetría $x = y$, cuando supusimos $x \neq y$. \square

La importancia de los órdenes totales radica precisamente en este hecho: dado que no existen elementos incomparables, siempre es posible saber o dar una jerarquía de orden entre todos los elementos. En este sentido, nos gustaría verificar si dado un orden parcial (A, \leq) , por ejemplo, el orden de tareas en la construcción de la casa del ejemplo 6.50, podrá existir una manera de ordenar todos los elementos de A , en este caso las tareas, de manera que se respeten las relaciones del orden parcial, pero que además cualesquiera dos tareas sean comparables. Es decir, queremos encontrar un orden total que “cubra” al orden parcial original⁷.

⁷Por ahora nos interesa sólo la existencia del orden total, pero en un caso como el de la construcción de una casa, más que sólo darle un orden total cualquiera a las actividades, lo que buscaríamos es determinar el tiempo mínimo que se llevaría en hacer todo el trabajo, aprovechando el paralelismo en aquellas actividades que no están relacionadas, usando, por ejemplo, el método *PERT*.

La pregunta general es: Dado un orden parcial (A, \leq) , ¿existirá un orden total (A, \leq_T) tal que si $x \leq y$ entonces $x \leq_T y$? La respuesta es positiva en el caso en que A sea un conjunto finito. Al proceso para hallar dicho orden se le conoce como *ordenamiento topológico*. En la página que sigue describimos un algoritmo que resuelve el problema.

Veamos la aplicación de este algoritmo a la construcción de la casa del ejemplo 6.5. Podemos pensar que $S_p = \{A, B, C, D, E, F, G, H, I, J, K, L, M, N\}$. Debemos notar que el orden de las tareas listadas no tiene que ver, forzosamente, con el orden parcial que las relaciona, por lo que la representación interna puede ser distinta en el orden en que presenta a sus elementos. La ejecución, mostrando los valores que van tomando S_p , s y k , y considerando una representación interna arbitraria, pero que refleje la relación de orden parcial, se encuentra en el listado 6.3 en la siguiente página. Lo que garantiza el resultado del ordenamiento es que cuando se ejecute una tarea y todas las tareas que la preceden ya estarán realizadas, pero este orden no garantiza, como ya mencionamos, un tiempo mínimo.

Figura 6.11. Ordenamiento topológico

Objetivo: Producir un orden total (A, \leq_T) a partir de una relación de orden parcial (A, \leq) .

Entrada: Un orden parcial (A, \leq) con A un conjunto finito.

Salida: El orden total (A, \leq_T) .

Método: Se encuentra en el listado 6.2 que sigue.

Listado 6.2. Algoritmo de ordenamiento topológico

```

1 // Inicialización
2 k=1; Sp=A; s=new array(|A|);
3 // Elige el siguiente elemento
4 while (Sp != ∅)
5     s[k] = Cualquier elemento minimal de Sp;
6     Sp = Sp - s[k]
7     k++
8 // Define  $\leq_T$ 
9  $\leq_T = \{(s[i], s[j]) \mid i \leq j\}$ 

```

Listado 6.3. Ejecución del ordenamiento topológico

(1/2)

k=1	$s[3]=B$
$S_p=\{A, B, C, D, E, F, G, H, I, J, K, L, M, N\}$	$S_p=\{D, E, F, G, H, I, J, K, L, M, N\}$
3 $s=\emptyset$;	12 k=4
$s[1] = A$	$s[4]=D$
$S_p=\{B, C, D, E, F, G, H, I, J, K, L, M, N\}$	$S_p=\{E, F, G, H, I, J, K, L, M, N\}$
6 k=2	15 k=5
$s[2]=C$	$s[5]=E$
$S_p=\{B, D, E, F, G, H, I, J, K, L, M, N\}$	$S_p=\{F, G, H, I, J, K, L, M, N\}$
9 k=3	18 k=6

Listado 6.3. Ejecución del ordenamiento topológico (2/2)

	$s[6]=H$	$S_p=\{K, L, M, N\}$ $k=11$
	$S_p=\{F, G, I, J, K, L, M, N\}$	33 $k=11$
21	$k=7$	$s[11]=N$
	$s[7]=F$	$S_p=\{K, L, M\}$
	$S_p=\{G, I, J, K, L, M, N\}$	36 $k=12$
24	$k=8$	$s[12]=K$
	$s[8]=J$	$S_p=\{L, M\}$
	$S_p=\{G, I, K, L, M, N\}$	39 $k=13$
27	$k=9$	$s[13]=L$
	$s[9]=G$	$S_p=\{M\}$
	$S_p=\{I, K, L, M, N\}$	42 $k=14$
30	$k=10$	$s[14]=M$
	$s[10]=I$	$S_p=\emptyset$
		45 $k=15$
		$s=[A, C, B, D, E, H, F, J, G,$
		$I, N, K, L, M]$

Podemos ver en la figura 6.12, mediante los diagramas de Hasse que van resultando cada vez que se elige un elemento minimal a partir de la línea 5, por qué las elecciones que hizo el algoritmo son válidas, ya que siempre se elimina a un elemento minimal. El diagrama de Hasse completo ya no lo mostramos.

Figura 6.12. Progreso del ordenamiento topológico sobre el ejemplo 6.50

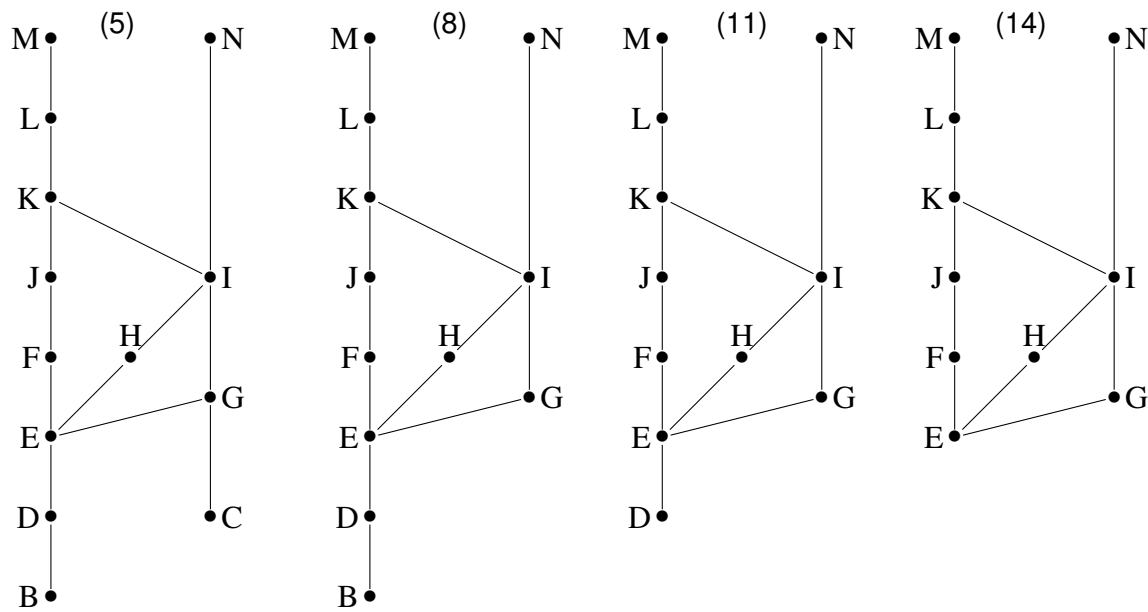
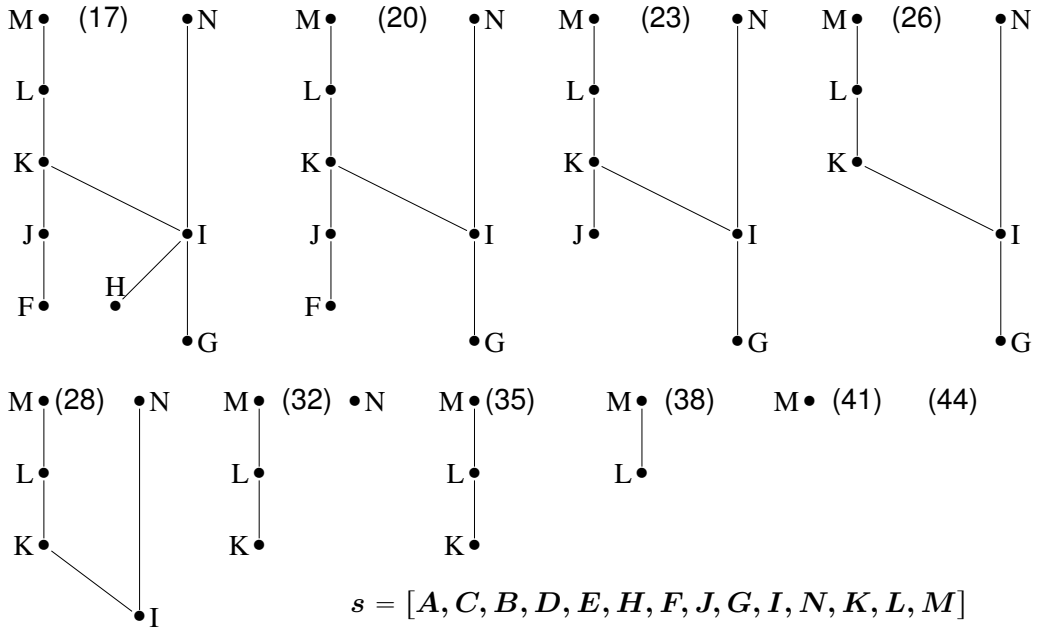


Figura 6.12. Progreso del ordenamiento topológico sobre el ejemplo 6.50

(2/2)



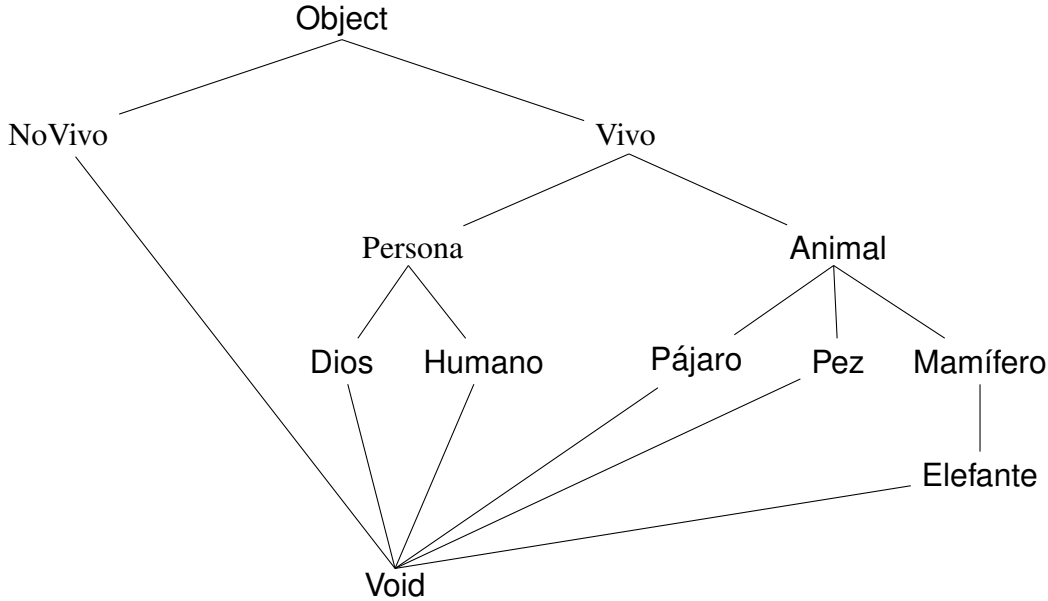
Enseguida estudiamos otra clase de elementos extremos más generales que los ya estudiados, que en cierto sentido son una generalización de los maximales y minimales.

6.6.2. Cotas

El concepto de cota es una generalización de los elementos maximales y minimales, y resulta de gran utilidad en Ciencias de la Computación. Por ejemplo, juega un papel de importancia en el análisis de algoritmos y en la teoría de la complejidad computacional y es de gran importancia en la programación orientada a objetos, al estar relacionado a los conceptos fundamentales de subtipado o herencia y conversión de clases (*casting*). Considérese el diagrama de Hasse en la figura 6.13, en la siguiente página, para la relación de herencia de clases en un lenguaje orientado a objetos.

Si se desea diseñar un método m que tenga sentido para los objetos de clase **Humano** y **Pez**, digamos *nadar*, resulta conveniente definirlo de manera que reciba objetos de una clase más general que englobe a aquéllas. Una posibilidad sería **Object**, pero esto resulta inconveniente dado que, de acuerdo a la relación de herencia, entonces sería legal enviarle a m un objeto de clase **NoVivo**. Lo obvio de acuerdo al diagrama es definir el método m en objetos de la clase **Vivo**, puesto que **Vivo** es la clase más pequeña que engloba tanto a **Humano** como a **Pez**. Es decir, las clases **Humano** y **Pez** están acotadas superiormente por las clases **Object** y **Vivo**. De estas cotas la más pequeña es **Vivo**. A continuación estudiamos de manera general estos conceptos.

Figura 6.13. Diagrama de Hasse para una relación de herencia en un lenguaje orientado a objetos



Definición 6.18 Sean (A, \leq) un orden parcial y $P \subseteq A$. Decimos que $x \in A$ es una cota inferior de P si y sólo si para toda $p \in P$ se cumple $x \leq p$. Análogamente, $y \in A$ es una cota superior de P si y sólo si para toda $p \in P$ se cumple $p \leq y$.

Es relevante observar que, a diferencia de un minimal, maximal, máximo o mínimo, una cota de un subconjunto P no tiene porqué ser elemento del subconjunto en cuestión. Por ejemplo, en el orden parcial (\mathbb{N}, \leq) , el subconjunto $P = \{3, 8, 25, 67, 126\}$ tiene a 2 como cota inferior y a 200 como cota superior, con ninguna de estas cotas elementos de P . Este mismo ejemplo muestra que las cotas de un subconjunto no son únicas necesariamente. En nuestro caso, P tiene también a 0 y 1 como cotas inferiores y además a 127, 514, 1234. En general, cualquier $n > 126$ es cota superior de P .

Una vez que se ha mostrado que un subconjunto dado P tiene al menos una cota superior o inferior, es natural preguntarse si entre todas sus cotas superiores (inferiores) habrá una mínima (máxima). En caso de que tales cotas existan son más útiles que una cota cualquiera, puesto que son las más cercanas a los elementos del subconjunto en cuestión, lo cual significa que son las más exactas.

Definición 6.19 Sean (A, \leq) un orden parcial y $P \subseteq A$. Si entre las cotas superiores de P existe una mínima, digamos z , entonces decimos que z es el supremo o mínima cota superior de P . Análogamente, si entre las cotas inferiores de P existe una máxima, digamos w , entonces decimos que w es el ínfimo o máxima cota inferior de P . El supremo de P se denota con $\sup P$ o $\sqcup P$, mientras que $\inf P$ o $\sqcap P$ denota al ínfimo de P .

Es claro que si $p \in P$ entonces $p \leq \sqcup P$ y $\sqcap P \leq p$. Obsérvese que si un conjunto tiene máximo (mínimo), éste también es una cota superior (inferior). La relación exacta entre el supremo (ínfimo) y el máximo (mínimo) de un conjunto se da a continuación.

Proposición 6.15 Sean (A, \leq) un orden parcial y $B \subseteq A$. Si B tiene un elemento máximo (mínimo) entonces B tiene supremo (ínfimo) y $\sqcup B = \max B$; también tenemos $\sqcap B = \min B$.

Demostración.

Si B tiene un elemento máximo $M = \max B$ entonces se cumple que para toda $x \in B$, $x \leq M$, por lo que M es cota superior de B y entonces $\sqcup B \leq M$. Por otro lado, como $M \in B$ entonces claramente $M \leq \sqcup B$. De donde, por la antisimetría se concluye que $\sqcup B = M$. El caso para el ínfimo es análogo. \square

La siguiente propiedad relaciona a los supremos e ínfimos con la contención y unión de conjuntos.

Proposición 6.16 Sean $S, T \subseteq A$ tales que $\sqcup S$, $\sqcup T$, $\sqcap S$, $\sqcap T$ existen. Entonces:

1. Si $S \subseteq T$ entonces $\sqcup S \leq \sqcup T$.
2. Si $S \subseteq T$ entonces $\sqcap T \leq \sqcap S$.
3. $\sqcup(S \cup T) = \sqcup\{\sqcup S, \sqcup T\}$.
4. $\sqcap(S \cup T) = \sqcap\{\sqcap S, \sqcap T\}$.

Demostración.

1. Como $S \subseteq T$, entonces para toda $x \in S$, $x \leq \sqcup T$. Por lo tanto, $\sqcup T$ es cota superior de S y, entonces por definición, $\sqcup S \leq \sqcup T$.
2. Análogo al caso anterior.
3. Análogo al siguiente caso.
4. Sean $p = \sqcap(S \cup T)$ y $q = \sqcap\{\sqcap S, \sqcap T\}$. Queremos mostrar que $p = q$. Como $S \subseteq S \cup T$, entonces, por la parte 2 de esta proposición, $p \leq \sqcap S$ y similarmente $p \leq \sqcap T$. Por lo tanto, p es cota inferior del conjunto $\{\sqcap S, \sqcap T\}$ y entonces $p \leq q$. Por otro lado, si $x \in S$ entonces $\sqcap S \leq x$ y como $q \leq \sqcap S$ entonces, por transitividad, $q \leq x$. Similarmente, si $x \in T$ entonces $q \leq x$. Así, q es cota inferior de $S \cup T$ y por lo tanto $q \leq p$. La antisimetría permite concluir ahora que $p = q$. \square

Un caso de particular importancia en el cálculo de supremos e ínfimos es cuando el conjunto P tiene únicamente dos elementos, digamos $P = \{x, y\}$. En este caso, las operaciones de supremo e ínfimo se vuelven operaciones binarias y escribimos de manera infija $x \sqcap y$ o $x \sqcup y$ en vez de $\sqcap\{x, y\}$ o $\sqcup\{x, y\}$.

Para un razonamiento más eficaz con las operaciones binarias supremo e ínfimo volvemos a enunciar su definición en una forma conveniente.

Definición 6.20 (supremo) Sean (A, \leq) un orden parcial y $x, y \in A$. Si $z \in A$ es tal que se cumplen las siguientes propiedades:

- (supC): $x \leq z$ y $y \leq z$
- (supM): para toda $w \in A$, si $x \leq w$ e $y \leq w$ entonces $z \leq w$.

Decimos entonces que z es el supremo de x e y , denotado $z = x \sqcup y$.

Análogamente definimos al ínfimo:

Definición 6.21 (ínfimo) Sean (A, \leq) un orden parcial y $x, y \in A$. Si $z \in A$ es tal que se cumplen las siguientes propiedades:

- (infC): $z \leq x$; y $z \leq y$
- (infM): para toda $w \in A$, si $w \leq x$ y $w \leq y$ entonces $w \leq z$.

Decimos en este caso que z es el ínfimo de x e y , denotado $z = x \sqcap y$.

Es claro que siempre sucede $x \sqcap y \leq x$, $x \sqcap y \leq y$ y también $x \leq x \sqcup y$, $y \leq x \sqcup y$. Veamos ahora otras propiedades de estas operaciones con respecto al orden parcial.

Proposición 6.17 Sea (A, \leq) un orden parcial. Se cumplen las siguientes propiedades para cualesquiera $x, y, z, v \in A$:

1. Si $x \leq y$ entonces $x \sqcap z \leq y \sqcap z$.
2. Si $x \leq y$ entonces $x \sqcup z \leq y \sqcup z$.
3. Si $x \leq y$ y $z \leq v$ entonces $x \sqcap z \leq y \sqcap v$.
4. Si $x \leq y$ y $z \leq v$ entonces $x \sqcup z \leq y \sqcup v$.

Demostración.

1. Para este caso basta ver que $x \sqcap z$ es cota inferior de $\{y, z\}$ y apelar a la propiedad (infM) de $y \sqcap z$. Se tiene $x \sqcap z \leq x$ y por hipótesis $x \leq y$, de donde por transitividad se sigue que $x \sqcap z \leq y$. Además es claro que $x \sqcap z \leq z$, por lo tanto hemos probado que $x \sqcap z$ es cota inferior de $\{y, z\}$ y entonces necesariamente $x \sqcap z \leq y \sqcap z$.
2. Análogo al anterior.
3. Análogo al siguiente caso.
4. Como $x \leq y$, por el inciso 2 de esta proposición se tiene que $x \sqcup z \leq y \sqcup z$. Similarmente, como $z \leq v$ entonces $z \sqcup y \leq v \sqcup y$ y como el operador \sqcup es conmutativo (véase la proposición 6.20) entonces $y \sqcup z \leq y \sqcup v$, de donde, por transitividad, se concluye que $x \sqcup z \leq y \sqcup v$. □

Ejercicios

- 6.6.1.- Demuestre que si (A, \leq) es un conjunto parcialmente ordenado que tiene un elemento máximo (mínimo) m , entonces m es maximal (minimal) y no existe ningún otro maximal (minimal).
- 6.6.2.- ¿Existe un elemento máximo en el conjunto parcialmente ordenado $(\mathbb{Z}^+, |)$? Justifique.

6.6.3.- Sea $A = \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$ y \leq la relación *ser múltiplo de* en A .

- a) ¿Tiene esta relación un elemento máximo? Si es así, ¿cuál?
- b) ¿Tiene esta relación un elemento mínimo? Si es así, ¿cuál?
- c) ¿Tiene esta relación elemento(s) maximal(es)? Si es así, ¿cuál(es)?
- d) ¿Tiene esta relación elemento(s) minimal(es)? Si es así, ¿cuál(es)?

6.6.4.- Sea \mathcal{F} una familia de conjuntos finitos tales que en \mathcal{F} no hay dos conjuntos con el mismo número de elementos. Además, para dos conjuntos A y B , $A \leq B$ si $|A| \leq |B|$. Es decir, el número de elementos en A –cardinalidad de A – es menor o igual que el número de elementos de B .

- a) Demuestre que R es una relación de orden total.
- b) ¿Tiene elemento máximo? Si es así, ¿cuál?
- c) ¿Tiene elemento mínimo? Si es así, ¿cuál?

6.6.5.- Sean (A, \leq) un orden parcial finito y M la matriz booleana para \leq .

- a) ¿Cómo podemos reconocer un elemento minimal de A a partir de M ?
- b) ¿Cómo podemos reconocer un elemento maximal de A a partir de M ?
- c) ¿Cómo podemos reconocer un elemento mínimo de A a partir de M ?
- d) ¿Cómo podemos reconocer un elemento máximo de A a partir de M ?

6.6.6.- Un orden parcial (A, \leq) es un *buen orden*, si y sólo si cualquier subconjunto no vacío de A tiene un elemento mínimo con respecto a \leq .

- a) Muestre que todo buen orden es un orden total.
- b) Muestre que todo orden total en un conjunto finito A es un buen orden.
- c) Muestre que la relación de orden usual \leq no constituye un buen orden en \mathbb{Z} o en \mathbb{R}^+ .
- d) Dé un ejemplo de un conjunto infinito con un buen orden.
- e) Dé un ejemplo de un orden total que no sea un buen orden.

6.6.7.- Sean $x, y \in \mathbb{R}$ con $x \leq y$. Considérese el intervalo abierto $I = (x, y)$.

- a) ¿Tiene I elementos máximo y mínimo? Justifique.
- b) ¿Tiene I elementos maximales y minimales? Justifique.
- c) Dados dos elementos arbitrarios $u, v \in I$, ¿existe el supremo $u \sqcup v$? ¿existe el ínfimo $u \sqcap v$? Justifique.
- d) Conteste lo anterior para el intervalo cerrado $I = [x, y]$.

6.6.8.- Sean (A, \leq) un orden parcial y $X \subseteq A$. Pruebe que si X tiene dos maximales (distintos) entonces X no tiene máximo.

6.6.9.- Sean (A, \leq) un orden parcial. Pruebe que si el orden es total y A tiene un maximal m , entonces m es máximo.

6.6.10.- Sea $A = \{2, 3, 4, \dots, 1999, 2000\}$. Considere el orden de divisibilidad $(A, |)$. ¿Cuántos elementos maximales tiene este orden?

6.6.11.- Sea $A = \{x, y\}$. Conteste lo siguiente:

- ¿Cuántos órdenes parciales hay para A ?
- ¿Cuáles de los órdenes parciales de A tienen a x como elemento minimal?
- ¿Cuáles de los órdenes parciales de A tienen a y como elemento minimal?
- ¿Cuáles de los órdenes parciales de A tienen a x como elemento maximal?
- ¿Cuáles de los órdenes parciales de A tienen a y como elemento maximal?

6.6.12.- Realice de nuevo el ejercicio anterior para $A = \{x, y, z\}$.

6.6.13.- Sean $B = \{0, 1\}$ y \leq el orden lexicográfico sobre $B \times B$.

- Determine todos los maximales y minimales.
- ¿Hay un mínimo o un máximo?
- ¿Es éste un orden total?

6.6.14.- Realice de nuevo el ejercicio 6.6.13 utilizando los órdenes antilexicográfico y de coordenadas.

6.6.15.- Repita el ejercicio 6.6.13 para el conjunto $X = \{0, 1, 2\}$ y los órdenes lexicográfico, antilexicográfico y de coordenadas.

6.6.16.- Considere el orden parcial (\mathbb{Q}, \leq) y $A = \{x \in \mathbb{Q} \mid x < 1\}$. Muestre que A no tiene un máximo pero sí tiene supremo.

6.6.17.- Encuentre un orden total a partir del siguiente orden parcial sobre el conjunto $A = \{1, 2, 3, 4, 5, 6, 7\}$, con $7 < 6$, $6 < 3$, $6 < 5$, $3 < 2$, $3 < 1$, $5 < 4$

6.6.18.- Un proyecto de desarrollo en una compañía requiere completar 7 procesos. Algunos de ellos pueden ser iniciados sólo después de que otros son finalizados. En la tabla de abajo se muestran las dependencias de los procesos.

Tarea	Dependencias
A	
B	A,C
C	
D	B
E	
F	B,E
G	D,F

- Haga el diagrama de Hasse correspondiente al orden parcial que se genera al definir $X < Y$ si Y no puede ser iniciado antes del término del proceso X .
- Encuentre un orden total para realizar los procesos.

6.6.19.- Un programador debe realizar pruebas de verificación de 10 programas que, debido a su interdependencia, están sujetos a las siguientes restricciones en el orden de ejecución:

$$\begin{array}{cccccc} 10 > 8 & 10 > 3 & 8 > 7 & 7 > 5 & 3 > 9 & 3 > 6 \\ 6 > 4 & 6 > 1 & 9 > 4 & 9 > 5 & 4 > 2 & 5 > 2 & 1 > 2 \end{array}$$

Determine un orden total para la ejecución de los programas.

6.6.20.- Considérese el orden parcial \subseteq sobre $\mathcal{P}(A)$ donde $A = \{1, 2, 3, 4, 5, 6, 7\}$. Sea $B = \{\{1\}, \{2\}, \{2, 3\}\}$. Determine lo siguiente:

- a) Las cotas superiores de B . b) Las cotas inferiores de B .
c) $\sqcup B$. d) $\sqcap B$.

6.6.21.- Realice de nuevo el ejercicio anterior esta vez para el orden parcial \supseteq .

6.6.22.- Sean (A, \leq) un orden parcial y $X, Y \subseteq A$ tales que $\sqcup X$ y $\sqcup Y$ existen. Pruebe que si $\sqcup X \in Y$ entonces $\sqcup Y$ es cota superior de X .

6.6.23.- Sean (A, \leq) un orden parcial y $X, Y \subseteq A$ tales que $\sqcup X$ existe y m es minimal de Y . Pruebe que si $\sqcup X \leq m$ entonces $X \cap Y \subseteq \{m\}$.

6.6.24.- Sean (A, \leq) un orden parcial y $X, Y \subseteq A$ tales que $\sqcup X$ y $\sqcap Y$ existen. Pruebe que si $\forall x \in X \forall y \in Y (x \leq y)$, entonces $\sqcup X \leq \sqcap Y$.

6.6.25.- Sea (A, \leq) un orden parcial tal que A tiene máximo, denotado 1, y mínimo, denotado 0. Pruebe que:

- a) $1 = \sqcup A$ y $1 = \sqcap \emptyset$ b) $0 = \sqcap \emptyset$ y $0 = \sqcap A$

6.7. Latices

Dado un orden parcial cualquiera (A, \leq) , puede suceder que para ciertos $x, y \in A$, los elementos $x \sqcup y$ y $x \sqcap y$ no existan. Una clase especial de órdenes parciales son aquellos donde se garantiza la existencia del supremo e ínfimo de cualesquiera dos elementos. Estos órdenes parciales se llaman retículas o latices y los estudiamos en lo que sigue.

Definición 6.22 Una *retícula* o *latiz*⁸ es un conjunto parcialmente ordenado A tal que para cualesquiera $x, y \in A$, tanto $x \sqcup y$ como $x \sqcap y$ existen, es decir, $x \sqcup y \in A$ y $x \sqcap y \in A$. Veamos algunos ejemplos de latices.

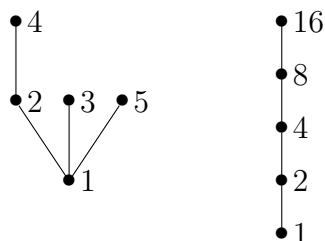
Ejemplo 6.61. Sea $\mathcal{B} = \{0, 1\}$ con $0 < 1$. Entonces \mathcal{B} es una latiz si definimos

$$x \sqcup y = \max\{x, y\}, \quad x \sqcap y = \min\{x, y\}.$$

Es decir, el supremo es el máximo mientras que el ínfimo es el mínimo de dos elementos.

⁸En inglés *lattice*

Ejemplo 6.62. Consideremos los conjuntos de enteros $S_1 = \{1, 2, 3, 4, 5\}$ y $S_2 = \{1, 2, 4, 8, 16\}$ con el orden de divisibilidad –los diagramas de Hasse correspondientes se muestran a continuación–.



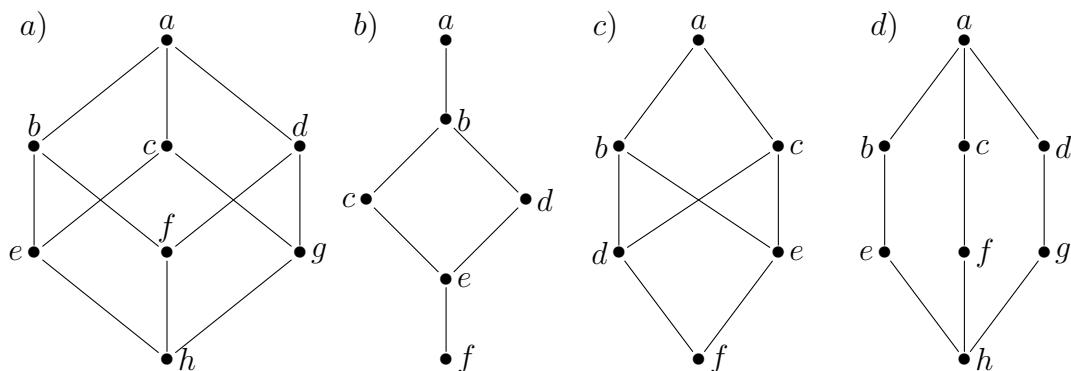
En el primer caso no tenemos una latiz puesto que $2 \sqcup 3$ no existe, dado que no hay cotas superiores para el conjunto $\{2, 3\}$. El segundo caso muestra un orden total, por lo que el supremo de cualesquiera dos elementos es el máximo de ellos y el ínfimo es el mínimo entre ellos.

Como se ve en el ejemplo anterior no cualquier conjunto de números conforma una latiz con el orden de divisibilidad, pero todo el conjunto de números naturales sí lo hace.

Ejemplo 6.63. El conjunto de números naturales \mathbb{N} con el orden de divisibilidad $|$ forma una latiz al definir $n \sqcap m = MCD(n, m)$ y $n \sqcup m = mcm(n, m)$. Es decir, el supremo de dos números es su mínimo común múltiplo, mientras que el ínfimo es su máximo común divisor.

Veamos otros ejemplos, esta vez razonando con los diagramas de Hasse.

Ejemplo 6.64. De los siguientes diagramas de Hasse, ¿cuáles corresponden a retículas?



- a) Este conjunto parcialmente ordenado corresponde a una retícula. Cualesquiera dos elementos tienen una mínima cota superior y una máxima cota inferior en el conjunto, lo cual se puede mostrar por inspección directa. Es fácil verificar que si a es un conjunto con tres elementos, digamos $a = \{1, 2, 3\}$ entonces este diagrama corresponde a la retícula $\mathcal{P}(a)$ del ejemplo 6.54, donde $h = \emptyset$; e, f, g son los subconjuntos de a con un elemento y b, c, d son los subconjuntos de a con dos elementos.

- b) Este diagrama también corresponde a una retícula. Para cualesquiera dos elementos, si están conectados, la mínima cota superior es el máximo de ellos (es decir, el que está más arriba), similarmente el ínfimo es el mínimo de ellos (el que está más abajo); esto sólo deja por verificar a la pareja (c, d) pero claramente se tiene $c \sqcup d = b$ y $c \sqcap d = e$.
- c) Este diagrama no corresponde a una latiz puesto que $d \sqcup e$ no existe. Se observa que las cotas superiores de $\{d, e\}$ son c, b, a pero no hay una mínima entre ellas, puesto que b y c son incomparables.
- d) Este diagrama de Hasse también corresponde a una retícula, pues como en el caso del diagrama b), todas las parejas directamente conectadas tienen su mínima cota superior en el mayor de ellos y su máxima cota inferior en el menor de ellos. Para las parejas restantes el lector debe verificar que el supremo es a y el ínfimo es h .

El siguiente ejemplo es de particular importancia en la teoría de conjuntos.

Ejemplo 6.65. Sea X un conjunto. El conjunto potencia de X , denotado $\mathcal{P}(X)$ –que corresponde a todos los subconjuntos de X –, con el orden \subseteq es una latiz. En particular, si $A, B \in \mathcal{P}(X)$, el supremo e ínfimo se definen como $A \sqcup B = A \cup B$ y $A \sqcap B = A \cap B$ respectivamente. El lector debe cerciorarse que con estas definiciones se cumplen las propiedades de las definiciones 6.20 y 6.21.

Veamos ahora diversas propiedades algebraicas de las latices.

En adelante se sugiere al lector pensar en el significado de cada propiedad en el caso específico de la latiz de conjuntos del ejemplo 6.65.

Lema 6.18 (Lema de conexión) *Sea A una latiz. Las siguientes condiciones son equivalentes:*

- 1) $x \leq y$.
- 2) $x \sqcup y = y$.
- 3) $x \sqcap y = x$.

Demostración.

- 1) \Rightarrow 2): Supongamos que $x \leq y$. Claramente $y \leq y$ y por hipótesis $x \leq y$, por lo que y es cota superior de $\{x, y\}$. Además, si $x \leq w$ e $y \leq w$, entonces obviamente $y \leq w$. Hemos probado $(\sup C)$ y $(\sup M)$ para y , por lo que se concluye que $x \sqcup y = y$.
- 2) \Rightarrow 3): Supongamos que $x \sqcup y = y$. Observemos que si w es cota inferior de $\{x, y\}$, entonces, en particular, $w \leq x$. Basta por lo tanto mostrar que x es cota inferior de $\{x, y\}$. Pero esto es claro, pues $x \leq x$ y como $x \leq x \sqcup y$, entonces, por la hipótesis, $x \leq y$.
- 3) \Rightarrow 1): Supongamos $x \sqcap y = x$. Como claramente $x \sqcap y \leq y$, entonces se concluye que $x \leq y$. □

El lema de conexión resulta ser una herramienta útil en muchas demostraciones, como vemos en la siguiente proposición que nos dice que todos los órdenes totales son latices.

Proposición 6.19 Si (A, \leq) es un orden total entonces es una latiz.

Demostración.

Sean (A, \leq) un orden total y $x, y \in A$. Como el orden es total, se tiene que $x \leq y$ o $y \leq x$. Supongamos, sin pérdida de generalidad, que $x \leq y$. Por el lema de conexión 6.18, se sigue que $x \sqcup y = y$ y $x \sqcap y = x$. Por lo tanto, el supremo e ínfimo de $\{x, y\}$ existen en A y A es una latiz. \square

Las siguientes propiedades nos son familiares de la teoría de conjuntos y, como veremos, son válidas también en todas las latices.

Proposición 6.20 Sean A una latiz y $x, y, z \in A$. Se cumple lo siguiente:

- *Conmutatividad:* $x \sqcap y = y \sqcap x$, $x \sqcup y = y \sqcup x$.
- *Asociatividad:* $x \sqcap (y \sqcap z) = (x \sqcap y) \sqcap z$, $x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z$.
- *Absorción:* $(x \sqcup y) \sqcap y = y$, $(x \sqcap y) \sqcup y = y$.

Demostración.

- *Conmutatividad:* es directo de las definiciones puesto que los conjuntos $\{x, y\}$ y $\{y, x\}$ son el mismo.
- *Asociatividad:* Veamos el caso para supremos. Para mostrar que

$$x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z,$$

de acuerdo a la definición 6.20, basta mostrar que

- (supC): $x \sqcup y \leq x \sqcup (y \sqcup z)$ y $z \leq x \sqcup (y \sqcup z)$ y
- (supM): Si $x \sqcup y \leq w$ y $z \leq w$ entonces $x \sqcup (y \sqcup z) \leq w$.

Como $y \leq y \sqcup z$, entonces $x \sqcup y \leq x \sqcup (y \sqcup z)$ por la conmutatividad de \sqcup y la parte 1 de la proposición 6.17. Además, $z \leq y \sqcup z \leq x \sqcup (y \sqcup z)$. Con esto queda verificada la propiedad (supC). Supongamos que $x \sqcup y \leq w$ y $z \leq w$. Para mostrar (supM) basta ver que w es cota superior de $\{x, y \sqcup z\}$. Se tiene $x \leq x \sqcup y \leq w$. Por otra parte $y \leq x \sqcup y \leq w$ y $z \leq w$, de donde w es cota superior de $\{y, z\}$ y por lo tanto $y \sqcup z \leq w$.

- *Absorción:* Por el lema de conexión 6.18, para probar $(x \sqcup y) \sqcap y = y$ basta probar $y \leq x \sqcup y$, pero esto último es claro. El otro caso es análogo. \square

Si pensamos en la latiz de conjuntos $\mathcal{P}(X)$ generada por el orden de la inclusión \subseteq (ejemplo 6.65), vemos que además de las operaciones de supremo \sqcup e ínfimo \sqcap , existe un elemento máximo X y un elemento mínimo \emptyset , además de una operación de complemento \overline{A} o $X - A = X \setminus A$. Enseguida generalizamos estas propiedades para cualquier latiz.

Definición 6.23 (latiz acotada) Sea (A, \leq) una latiz. Decimos que A es acotada si A tiene un elemento mínimo y un elemento máximo. En tal caso, estos elementos se denotan con 0 y 1 respectivamente.

Proposición 6.21 Sea (A, \leq) una latiz acotada. Para cualquier $x \in A$ se cumple:

$$x \sqcup 1 = 1 \quad x \sqcup 0 = x \quad x \sqcap 1 = x \quad x \sqcap 0 = 0$$

Demostración.

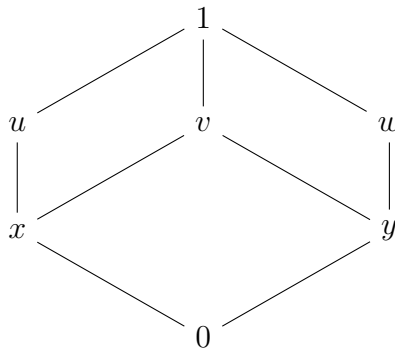
Son consecuencias directas del lema de conexión 6.18. □

Definición 6.24 Sean (A, \leq) una latiz acotada y $x! \in A$. Decimos que $z \in A$ es un complemento de x si sucede que $x \sqcup z = 1$ y $x \sqcap z = 0$.

Es importante observar que en una latiz cualquiera un elemento puede o no tener complemento y además, en caso de tenerlo, no tiene porqué ser único. Veamos un ejemplo.

Ejemplo 6.66. Sean $A = \{0, 1, u, v, w, x, y\}$ donde $0, 1$ son mínimo y máximo y además $x \leq u$, $x \leq v$, $y \leq v$ e $y \leq w$. El diagrama de Hasse correspondiente se encuentra en la figura 6.14 a continuación. Es claro que A es una latiz acotada. Más aún, w tiene como complementos a u y x , mientras que v no tiene complementos.

Figura 6.14. Diagrama de Hasse correspondiente al ejemplo 6.66



Una clase importante de latices son aquellas llamadas *complementadas*, cuya definición se encuentra a continuación.

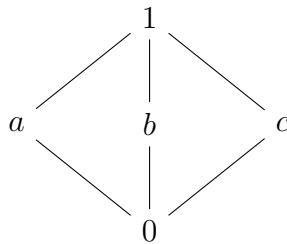
Definición 6.25 (latiz complementada) Sea (A, \leq) una latiz acotada. Decimos que A es una latiz complementada si todo $x \in A$ tiene un complemento.

En una latiz complementada no se pide que los complementos sea únicos, como en el siguiente ejemplo.

Ejemplo 6.67. Sea $A = \{0, 1, a, b, c\}$ donde el orden se define como

$$x \leq y \text{ si y sólo si } x = 0 \text{ o } y = 1 \text{ o } x = y,$$

cuyo diagrama de Hasse es:



Es claro que A es una latiz acotada y que $a \sqcup b = b \sqcup c = c \sqcup a = 1$ y $a \sqcap b = b \sqcap c = c \sqcap a = 0$. Por lo tanto, cada uno de los elementos a, b, c es complemento de los otros dos.

La condición necesaria para garantizar la unicidad de los complementos es otra propiedad familiar de las operaciones con conjuntos, la distributividad.

Definición 6.26 (latiz distributiva) Sea (A, \leq) una latiz. Decimos que A es distributiva si cumple las siguientes condiciones para cualesquiera $x, y, z \in A$:

$$x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z), \quad x \sqcup (y \sqcap z) = (x \sqcup y) \sqcap (x \sqcup z).$$

Proposición 6.22 Sea (A, \leq) una latiz complementada. Si A es distributiva entonces los complementos son únicos.

Demostración.

Sea $x \in A$ y supongamos que $y, z \in A$ son complementos de x . Vamos a mostrar que $y = z$. Como z es complemento de x se tiene $x \sqcap z = 0$, de donde se sigue:

$$y = y \sqcup 0 = y \sqcup (x \sqcap z).$$

Por la distributividad,

$$y \sqcup (x \sqcap z) = (y \sqcup x) \sqcap (y \sqcup z).$$

Como y también es complemento de x se tiene $y \sqcup x = 1$, de donde

$$(y \sqcup x) \sqcap (y \sqcup z) = 1 \sqcap (y \sqcup z) = y \sqcup z.$$

Por lo tanto $y = y \sqcup z$. Análogamente se prueba $z = y \sqcup z$ y por lo tanto $z = y$. □

De acuerdo a esta proposición, en adelante podemos denotar al complemento único de un elemento x en una latiz distributiva con x^* .

Proposición 6.23 Sea (A, \leq) una latiz complementada distributiva. Se cumplen las siguientes propiedades para cualesquiera $x, y \in A$:

1. *Idempotencia:* $(x^*)^* = x$.
2. *De Morgan:* $(x \sqcup y)^* = x^* \sqcap y^*$.
3. *De Morgan:* $(x \sqcap y)^* = x^* \sqcup y^*$.

Demostración.

1. Es claro que x es complemento de x^* puesto que por la definición de x^* y la conmutatividad del supremo e ínfimo, se tiene $x^* \sqcup x = 1$ y $x^* \sqcap x = 0$. De aquí se sigue $(x^*)^* = x$, puesto que los complementos son únicos.
2. Basta ver que $x^* \sqcap y^*$ es complemento de $x \sqcup y$ y aplicar la unicidad del complemento.

■ Comprobemos que

$$(x^* \sqcap y^*) \sqcup (x \sqcup y) = 1.$$

Como (A, \leq) es asociativa,

$$(x^* \sqcap y^*) \sqcup (x \sqcup y) = ((x^* \sqcap y^*) \sqcup x) \sqcup y$$

y por la distributividad

$$\begin{aligned} ((x^* \sqcap y^*) \sqcup x) \sqcup y &= ((x^* \sqcup x) \sqcap (y^* \sqcup x)) \sqcup y \\ &= (1 \sqcap (y^* \sqcup x)) \sqcup y \\ &= (y^* \sqcup x) \sqcup y \\ &= y^* \sqcup (x \sqcup y) \\ &= y^* \sqcup (y \sqcup x) \\ &= (y^* \sqcup y) \sqcup x \\ &= 1 \sqcup x = 1 \end{aligned}$$

■ Comprobemos ahora que

$$\begin{aligned} (x^* \sqcap y^*) \sqcap (x \sqcup y) &= 0. \\ (x^* \sqcap y^*) \sqcap (x \sqcup y) &= (x^* \sqcap y^* \sqcap x) \sqcup (x^* \sqcap y^* \sqcap y) \\ &= ((x^* \sqcap x) \sqcap y^*) \sqcup (x^* \sqcap (y^* \sqcap y)) \\ &= (0 \sqcap y^*) \sqcup (x^* \sqcap 0) \\ &= 0 \sqcup 0 = 0 \end{aligned}$$

3. Es análogo al caso anterior. □

Finalmente hemos logrado definir una clase especial de latices, las complementadas distributivas, que cumplen todas las propiedades del álgebra de conjuntos. Estas latices son tan importantes que tienen un nombre propio.

Definición 6.27 (álgebra booleana) Un álgebra booleana es una latiz complementada distributiva (A, \leq) .

El ejemplo clásico de álgebra booleana es el álgebra de un conjunto potencia del ejemplo 6.65. Esta clase de estructuras surgen en diversas ramas de la matemática como el Álgebra y la Topología. Otro ejemplo que ya hemos utilizado a profundidad es el que se encuentra enseguida.

Ejemplo 6.68. La latiz $B = \{0, 1\}$ donde $0 < 1$, definida en el ejemplo 6.7, es un álgebra booleana donde los complementos se definen como: $0^* =_{def} 1$, $1^* =_{def} 0$.

Este ejemplo corresponde esencialmente a lo que llamamos álgebra booleana en el ámbito de los circuitos digitales, y que estudiamos ampliamente en el capítulo 3.

Queremos terminar este capítulo con un ejemplo de álgebra booleana que a estas alturas de nuestro aprendizaje ya es muy familiar.

Ejemplo 6.69. Sea A el conjunto de fórmulas de la lógica proposicional bajo la equivalencia lógica. Es decir, $A = Prop / \equiv = \{[P] \mid P \in Prop\}$.

Definimos un orden parcial \leq sobre A como sigue: $[P] \leq [Q]$ si y sólo si $P \models Q$.

Entonces, (A, \leq) es un orden parcial y además es un álgebra booleana, al definir

Supremo: $[P] \sqcup [Q] = [P \vee Q]$.

Ínfimo: $[P] \sqcap [Q] = [P \wedge Q]$.

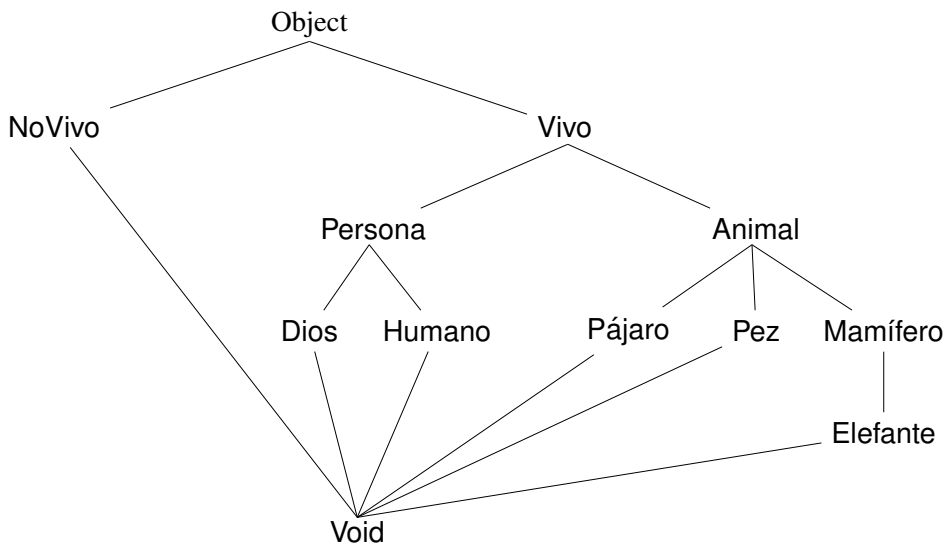
Complemento: $[P]^* = [\neg P]$.

Se deja al lector verificar todas las propiedades correspondientes.

Este ejemplo deja ver una situación que no es poco común en matemáticas. En el capítulo 2 estudiamos ampliamente a la estructura matemática conocida como Lógica Proposicional, la cual aparentemente no tiene una conexión directa con los conceptos de este capítulo. Sin embargo, como acabamos de mostrar, la lógica proposicional es un ejemplo de latiz. Esto proporciona evidencia de que ciertos objetos o conceptos matemáticos que son muy generales desde un punto de vista, pueden ser ejemplos particulares de otros objetos o conceptos desde la perspectiva de otra teoría matemática.

Ejercicios

6.7.1.- Considere el diagrama de clases discutido en la sección 6.6.2 (que se repite enseguida). Defina formalmente el conjunto de clases A , así como la relación de herencia \leq y muestre que (A, \leq) es una latiz.



6.7.2.- Sea (A, \leq) un orden parcial con máximo y mínimo. Muestre que A es una latiz si la siguiente condición se cumple:

- Para cualesquiera $a_1, a_2, b_1, b_2 \in A$ tales que $a_i \leq b_j$, para $i, j = 1, 2$, existe $c \in A$ tal que $a_i \leq c \leq b_j$ para $i, j = 1, 2$.

6.7.3.- Sean (A, \leq_A) , (B, \leq_B) latices. Muestre que $(A \times B, \leq)$ es una latiz, donde \leq es el orden de coordenadas y las operaciones de supremo e ínfimo se definen coordenada a coordenada, es decir:

$$(a_1, b_1) \sqcup (a_2, b_2) = (a_1 \sqcup a_2, b_1 \sqcup b_2), \quad (a_1, b_1) \sqcap (a_2, b_2) = (a_1 \sqcap a_2, b_1 \sqcap b_2).$$

6.7.4.- Sea (A, \leq) una latiz. Muestre que si $B \subseteq A$ es finito, entonces B tiene supremo e ínfimo.

6.7.5.- Sean (A, \leq) una latiz y $x, y, z \in A$. Muestre que si $x \leq z$ entonces $z \sqcup (x \sqcap y) = z$.

6.7.6.- Sean (A, \leq) una latiz y $x, y, z \in A$. Demuestre las llamadas *desigualdades distributivas*:

$$a) \quad x \sqcup (y \sqcap z) \leq (x \sqcup y) \sqcap (x \sqcup z).$$

$$b) \quad (x \sqcap y) \sqcup (x \sqcap z) \leq x \sqcap (y \sqcup z).$$

6.7.7.- Sean (A, \leq) una latiz y $x, y, z \in A$. Muestre que:

$$a) \quad x \sqcup y \leq z \text{ si y sólo si } x \leq z \text{ y } y \leq z.$$

$$b) \quad z \leq x \sqcap y \text{ si y sólo si } z \leq x \text{ y } z \leq y.$$

6.7.8.- Sean (A, \leq) una latiz y $x, y, z \in A$. Demuestre que si $y \leq x \leq y \sqcup z$ entonces $x \sqcup z = y \sqcup z$.

6.7.9.- Sean (A, \leq) una latiz y $x, y, z \in A$. Demuestre la siguiente desigualdad:

$$(x \sqcap y) \sqcup (y \sqcap z) \sqcup (z \sqcap x) \leq (x \sqcup y) \sqcap (y \sqcup z) \sqcap (z \sqcup x)$$

6.7.10.- Muestre que si (A, \leq) es una latiz acotada y A tiene más de un elemento, entonces $1 \neq 0$.

6.7.11.- Muestre que cualquier latiz (A, \leq) , donde A tiene a lo más cuatro elementos, es distributiva.

6.7.12.- Sea (A, \leq) una latiz distributiva. Pruebe que las dos leyes distributivas son equivalentes, es decir, demuestre que las siguientes condiciones son equivalentes.

$$a) \quad \forall x, y, z \in A \left(x \sqcup (y \sqcap z) = (x \sqcup y) \sqcap (x \sqcup z) \right)$$

$$b) \quad \forall x, y, z \in A \left(x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z) \right)$$

6.7.13.- Decimos que una latiz (A, \leq) es *completa* si y sólo si cualquier subconjunto $B \subseteq A$ tiene supremo e ínfimo. Demuestre lo siguiente:

a) Toda latiz finita es completa.

b) El orden parcial $(\mathbb{R}, <)$ no es una latiz completa.

c) Si $x, y \in \mathbb{R}$ con $x < y$, entonces el intervalo cerrado $[x, y]$ es una latiz completa.

6.7.14.- Sean (A, \leq) una latiz. Demuestre que las siguientes condiciones son equivalentes.

a) (A, \leq) es una latiz completa.

b) Cualquier subconjunto $B \subseteq A$ tiene ínfimo.

c) Cualquier subconjunto $B \subseteq A$ tiene supremo.

6.7.15.- ¿Es $(\mathbb{N}, |)$ una latiz acotada?

6.7.16.- Determine para qué casos de X es la latiz $(\mathcal{P}(X), \subseteq)$ una latiz acotada.

6.7.17.- Determine si el orden total (A, \leq) es una latiz distributiva.

6.7.18.- Sean (A, \leq) una latiz distributiva y $x, y, z \in A$. Demuestre la ley de cancelación:

$$\text{Si } x \sqcup z = y \sqcup z \text{ y } x \sqcap z = y \sqcap z \text{ entonces } x = y.$$

6.7.19.- Sean $n \in \mathbb{N}, n > 0$ y D_n el conjunto de divisores de n .

a) Muestre que el orden de divisibilidad $(D_n, |)$ es una latiz complementada.

b) ¿Para qué $n \in \mathbb{N}$ es $(D_n, |)$ un álgebra booleana?

6.7.20.- Sean (A, \leq) un álgebra booleana y $x, y \in A$. Demuestre lo siguiente:

a) $x \leq y$ si y sólo si $x \sqcap y^* = 0$.

b) $y \leq x$ si y sólo si $x \sqcup y^* = 1$.

6.7.21.- Sean (A, \leq) un álgebra booleana y $x, y, z \in A$. Demuestre lo siguiente:

a) Si $x \leq y$ entonces $y^* \leq x^*$

b) Si $y \sqcap z = 0$ entonces $y \leq z^*$.

c) Si $x \leq y$ y $y \sqcap z = 0$ entonces $z \leq x^*$.

6.8. Bases de datos relacionales

Ya que hemos estudiado ampliamente las diversas clases de relaciones binarias y algunas de sus aplicaciones, volvemos al caso de las relaciones n -arias. Recordemos su definición.

Definición 6.28 (relación n -aria) Sea $n > 0$. Una relación n -aria R sobre los conjuntos A_1, A_2, \dots, A_n es un subconjunto del producto cartesiano $A_1 \times A_2 \times \dots \times A_n$, es decir, $R \subseteq A_1 \times A_2 \times \dots \times A_n$. En tal caso los conjuntos A_1, A_2, \dots, A_n son los *dominios* de la relación y n es el *índice* (grado, aridad) de la relación. Si la n -ada (x_1, \dots, x_n) forma parte de una relación R , escribimos indistintamente $(x_1, \dots, x_n) \in R$ o bien $R(x_1, \dots, x_n)$.

En particular, vamos a discutir la utilidad de esta clase de relaciones en el manejo de información mediante bases de datos. Empecemos mencionando algunos ejemplos.

Ejemplo 6.70. En lo que sigue listamos varias relaciones n -arias con $n > 2$.

- $entre(a, b, c)$. Una tríada (a, b, c) está en la relación $entre$ si cumple $a, b, c \in \mathbb{R}$ y $a < b < c$. Por ejemplo, $(5.0, 7.0, 7.00001) \in entre$.
- $R = \{(x, y, z) \mid x, y, z \in \mathbb{R} \text{ e } y = (x - z)/2\}$. Ejemplos de tripletas (o tríadas) que están en esta relación son $(9, 2, 5)$, $(17, 6, 5)$ y $(2435, 0.5, 2434)$.
- $vuelo(a, o, d, ds, hs, ha)$. Una séxtupla (a, o, d, ds, hs, ha) está en la relación $vuelo$ si y sólo si a es un nombre de aerolínea; o es la ciudad origen; d es la ciudad destino; ds es un día de la semana; hs es la hora de salida y ha es la hora de arribo. Por ejemplo, el vuelo 499 de Lufthansa parte los días viernes de la ciudad de México a Francfort del Meno a las 21hrs, llegando a las 13:30hrs del día siguiente, lo cual significa que $vuelo(Lufthansa, Mex, Fra, Vi, 2100, 1330 + 1)$.

Estudiamos ahora una aplicación importante de las relaciones n -arias en el manejo de información mediante bases de datos. Para nuestros propósitos, una *base de datos* es un conjunto de información organizada de cierta manera, que permita acceso, inserción, eliminación, actualización y búsqueda de elementos particulares, de forma eficiente y correcta. Debido a la importancia de estas operaciones se requiere un modelo formal que las represente, en nuestro caso el llamado modelo relacional.

Definición 6.29 (base de datos relacional) Una *base de datos relacional* es una relación n -aria $D \subseteq A_1 \times A_2 \times \dots \times A_n$.

Se acostumbra mostrar la base de datos relacional⁹ como una tabla, donde cada columna tiene el nombre del dominio y cada renglón contiene a un elemento (tupla) de la relación, con los valores correspondientes para A_1, A_2, \dots, A_n , por lo que muchas veces se usa directamente este apelativo (*tabla*) para referirse a una base de datos relacional.

A cada renglón de la tabla de la relación se le conoce como *registro* y a cada columna como *atributo*, pensando en que cumple con una cierta limitación que se dio sobre cada conjunto A_i . Al producto cartesiano $A_1 \times A_2 \times \dots \times A_n$ se le llama la *intención* de la base de datos, mientras que a la tabla se le llama la *extensión* de la base de datos. Obsérvese que un registro es formalmente una tupla de la relación, mientras que un atributo es una de las entradas de dicha tupla.

Ejemplo 6.71. Considérese la siguiente base de datos llamada *Estudiantes*, donde

$$Estudiantes \subseteq \text{Nombre} \times \text{Núm.Cta.} \times \text{Carrera} \times \text{Promedio},$$

cuya definición se muestra en la tabla 6.5. Un elemento de D es, por ejemplo,

$$(Medina, 067493208, Actuaría, 8.00).$$

La *intención* de la tabla (el dominio de la relación) es el producto cartesiano

$$\text{Nombre} \times \text{Núm.Cta.} \times \text{Carrera} \times \text{Promedio},$$

mientras que la *extensión* es el contenido que se muestra en la tabla 6.5, en la siguiente página.

⁹En adelante *base de datos* únicamente.

Tabla 6.5. Base de datos de *Estudiantes*

Estudiantes			
Nombre	Núm. Cta.	Carrera	Promedio
Avella	924273894	Matemáticas	9.75
Aguilar	958395179	Actuaría	8.50
Medina	067493208	Actuaría	8.00
Morales	105730762	Física	8.40
Padilla	103726591	C. de la Computación	8.50
Vázquez	966429754	Actuaría	9.75
Zuazua	102582843	Matemáticas	9.50

Definición 6.30 (llave primaria) Dada una base de datos $D \subseteq A_1 \times A_2 \dots \times A_n$, decimos que el dominio A_i es una *llave primaria* para D si y sólo si no existen dos registros distintos $r_1 = (a_1, \dots, a_i, \dots, a_n)$ y $r_2 = (b_1, \dots, b_i, \dots, b_n) \in D$ que tengan el mismo valor en A_i , es decir, no existen r_1 y r_2 tales que $r_1 \neq r_2$ y $a_i = b_i$. En otras palabras, el valor del atributo i -ésimo determina unívocamente al registro.

En la base de datos *Estudiantes* de la tabla 6.5 en la página anterior, vemos que el nombre es único para todos sus elementos, por lo que este atributo es una llave primaria. Asimismo, el número de cuenta es único y constituye otra llave primaria.

A veces no es posible hallar una llave primaria en una base de datos por lo que requerimos del concepto de llave compuesta.

Definición 6.31 (llave compuesta) Una *llave compuesta* para la base de datos $D \subseteq A_1 \times A_2 \dots \times A_n$ es una colección de dos o más dominios $\{A_{i_1}, \dots, A_{i_k}\}$, $k \geq 2$, tales que juntos determinan unívocamente a cada registro.

Para la base de datos de la tabla 6.5 en la página anterior, una llave compuesta es la carrera junto con el promedio, que también identifican unívocamente a cada registro.

Las bases de datos se comportan como un álgebra de relaciones (o relacional) y, como tal, tienen asociadas las operaciones que se mencionan en los siguientes párrafos, que se aplican a cualquier relación n -aria.

Operaciones del álgebra relacional

Además de las operaciones de conjuntos, que ya vimos, el álgebra relacional presenta tres operaciones que no provienen de operaciones con conjuntos y que discutimos en lo que sigue.

La operación más simple del álgebra relacional es la de seleccionar algunos elementos de la base de datos original, de acuerdo a si satisfacen o no una condición dada.

select, $s_C(\mathbf{R})$: Dada una relación R y una condición C , elige a aquellos registros de D que cumplan con la condición C , formando así una nueva relación que resulta ser un subconjunto de la relación original.

Ejemplo 6.72. Considere la relación $\text{suma}(a_1, a_2, a_3)$, donde a_1, a_2 y a_3 están en suma si y sólo si $a_1 + a_2 = a_3$, de tal forma que

$$\text{suma} = \{(2, 4, 6), (5, -3, 2), (8, 2, 10), (3, 4, 7), (8, -2, 6)\}.$$

Entonces:

- Si C es la condición $a_2 \geq 0$ entonces $s_C(\text{suma}) = \{(2, 4, 6), (8, 2, 10), (3, 4, 7)\}$.
- Si C es la condición $a_3 = 4$ entonces $s_C(\text{suma}) = \emptyset$, ya que ninguna de las parejas de la relación tiene como tercer componente a 4.

Otra forma de representar una selección es colocando directamente la condición como argumento al operador select, como en $\text{select}(\text{suma}, a_3 = 4)$, que indica la operación seleccionar de la relación suma a aquellos registros donde el tercer atributo sea 4.

Ejemplo 6.73. De acuerdo a la base de datos *Estudiantes*, mostrada en la tabla 6.5 se tiene que:

$$\text{selec } t(\text{Estudiantes}, \text{Promedio} \geq 9.00) = \{(\text{Avella}, 924273894, \text{Matemáticas}, 9.75), \\ (\text{Vázquez}, 966429754, \text{Actuaría}, 9.75), (\text{Zuazua}, 102582843, \text{Matemáticas}, 9.50)\}.$$

La siguiente operación consiste en eliminar algunos atributos de cada registro en una relación dada.

Proyección, $P_{i_1, i_2, \dots, i_m}(\mathbf{R})$: Sean R una relación n -aria e $i_1 < i_2 < \dots < i_m, m \leq n$. La operación de proyección transforma cada n -ada $r = (a_1, a_2, \dots, a_n) \in R$ en un registro de aridad m , construido al tomar únicamente los atributos a_{i_k} de r . Esto es, a partir de todas las n -adas de la relación R se construye una nueva relación m -aria $Q \subseteq A_{i_1} \times A_{i_2} \times \dots \times A_{i_m}$ que contiene el mismo número de registros que la relación original (aunque en ciertas situaciones podrían ser menos), pero su extensión consta de registros de aridad m determinados por los atributos $A_{i_k}, 1 \leq k \leq m$.

Ejemplo 6.74. Sea $R = \{(8, 5, 7, 9), (3, 4, 3, 2), (7, 5, 2, 9), (3, 4, 5, 6)\}$ una relación de índice 4. Entonces, $P_{2,4}(R) = \{(5, 9), (4, 2), (5, 9), (4, 6)\}$, pero como estamos hablando de conjuntos y el primer y tercer elemento de la proyección son iguales, entonces en realidad se obtiene la relación binaria: $P_{2,4} = \{(5, 9), (4, 2), (4, 6)\}$.

Este ejemplo muestra que, bajo la operación de proyección, el número original de registros en una base de datos puede disminuir.

Ejemplo 6.75. Usemos nuevamente la tabla 6.5. En ella, $P_{2,4}(\text{Estudiantes})$ resulta en la (sub)tabla 6.6, en la siguiente página.

Tabla 6.6. Proyección de la base de datos de alumnos

<i>Estudiantes_{2,4}</i>	
Núm. Cta.	Promedio
924273894	9.75
958395179	8.50
067493208	8.00
105730762	8.40
103726591	8.50
966429754	9.75
102582843	9.50

La última operación del álgebra relacional que discutiremos consiste en la unión o combinación de dos relaciones, lo cual es posible cuando dichas relaciones comparten algunos atributos.

Join, $J_p(R, S)$: Sean R, S dos relaciones de índices m y n respectivamente.

Si $p \leq m$ y $p \leq n$, definimos la unión de R y S como sigue

$$Join_p(R, S) = \{ (a_1, \dots, a_{m-p}, c_1, \dots, c_p, b_1, \dots, b_{n-p}) \\ | (a_1, \dots, a_{m-p}, c_1, \dots, c_p) \in R, \\ (c_1, \dots, c_p, b_1, \dots, b_{n-p}) \in S \}.$$

Observaciones:

- El operador J_p produce una nueva relación de aridad $m+n-p$ combinando m -tuplas r de R con n -tuplas s de S , donde los últimos p componentes de la m -tupla r coinciden con los primeros p componentes de la n -tupla s .
- El número p corresponde al número de columnas con el mismo atributo en R y en S . La relación resultante de la operación $Join$ es de índice $n+m-p$ columnas, pues las p columnas en común no se repiten. Más aún, este relación tiene a lo más el número de renglones de la primera tabla.
- Si las relaciones R, S no tienen atributos en común, la operación $Join$ regresará un conjunto vacío.

Ejemplo 6.76. Sean $R \subseteq \mathbb{N} \times \mathbb{N}$ y $S \subseteq \mathbb{N} \times \mathbb{N}$ dos relaciones definidas como sigue:

$$R = \{ (1, 2, 3, 4, 5), (5, 8, 3, 4, 5), (8, 16, 24, 32, 40), (4, 6, 12, 16, 20), \\ (5, 10, 15, 20, 25) \}$$

$$S = \{ (3, 4, 5, 6, 8, 10), (3, 4, 5, 25, 30, 35), (24, 32, 40, 48, 56, 64), \\ (4, 6, 12, 16, 20, 24), (6, 12, 18, 24, 30, 36) \}$$

Entonces, el número de elementos en cada tupla del conjunto $Join_3(R, S)$ es

$5 + (6 - 3) = 8$ y el resultado es:

$$Join_3(R, S) = \{ (1, 2, 3, 4, 5, 6, 8, 10), (1, 2, 3, 4, 5, 25, 30, 35), \\ (5, 8, 3, 4, 5, 16, 6, 8, 10), (5, 8, 3, 4, 5, 25, 30, 35), \\ (8, 16, 24, 32, 40, 48, 56, 64) \}$$

A continuación mostramos el cálculo de cada tupla de esta relación.

tupla	viene de:	
	R	S
(1, 2, 3, 4, 5 , 6, 8, 10),	(1, 2, 3, 4, 5)	(3, 4, 5 , 6, 8, 10)
(1, 2, 3, 4, 5 , 25, 30, 35),	(1, 2, 3, 4, 5)	(3, 4, 5 , 25, 30, 35)
(5, 8, 3, 4, 5 , 6, 8, 10),	(5, 8, 3, 4, 5)	(3, 4, 5 , 6, 8, 10)
(5, 8, 3, 4, 5 , 25, 30, 35),	(5, 8, 3, 4, 5)	(3, 4, 5 , 25, 30, 35)
(8, 16, 24, 32, 40 , 48, 56, 64)	(8, 16, 24, 32, 40)	(24, 32, 40 , 48, 56, 64)

No hay ninguna otra tupla que pertenezca a $Join_3(R, S)$, ya que ninguna otra tupla en R termina con los mismos tres números con los que empieza alguna tupla en S .

Es importante observar que esta no es la única manera de definir la operación $Join$. De hecho, es fácil darse cuenta que no importa que los atributos finales de una relación R coincidan con los atributos iniciales de una relación S para que la operación de unión tenga sentido; basta con que existan atributos en común, no importando el orden.

Para terminar esta sección desarrollamos una serie de ejemplos acerca de la organización de salones, cursos y horarios en la Facultad de Ciencias de la UNAM.

Ejemplo 6.77. Empezamos definiendo una base de datos acerca de los salones y sus características.

$$Salones \subseteq edificio \times núm \times tipo \times capacidad \times equipo$$

$$\begin{aligned} \text{donde:} \quad edificio &= \{O, P, B1, B2, F, A, T, S\} \\ núm &= \mathbb{N} \\ tipo &= \{lab, taller, salón, aula, auditorio\} \\ capacidad &= \mathbb{N} \\ equipo &= \{falso, verdadero\} \end{aligned}$$

El atributo *edificio* indica el nombre particular de cada edificio de salones; *núm* indica el número de salón; *tipo* se refiere a la clase de instalaciones en un salón particular; *capacidad* denota al número posible de alumnos por salón y *equipo* indica si el salón cuenta o no con computadoras. La extensión de nuestra base de datos está dada por la tabla 6.7, que se encuentra en la página que sigue.

Tabla 6.7. Relación dada para los salones

<i>Salones</i>				
<i>edificio</i>	<i>núm</i>	<i>tipo</i>	<i>capacidad</i>	<i>equipo</i>
O	106	salón	64	F
O	130	salón	32	F
T	215	taller	30	T
T	8	salón	70	F
A	25	lab	25	F
B	10	lab	25	F

En lo que sigue definimos una base de datos. Supongamos *Cursos* sobre

$lice \times clave \times grupo \times creds \times horario \times días \times edificio \times núm \times prof$
donde:

$lice : \{act, bio, cc, ct, fis, mat\},$ $clave : \mathbb{N},$
 $grupo : \mathbb{N},$ $creds : \mathbb{N},$
 $horario : String,$ $días : String$
 $prof : String,$ $edificio : \{O, P, B1, B2, F, A, T, S\},$
 $num : \mathbb{N}$

Aquí, *lice* indica la licenciatura; *clave*, *grupo* y *creds* son números que indican la clave, el número de grupo y los créditos, respectivamente, de un curso; *horario*, *días* y *prof* son cadenas (elementos del tipo *String*) que indican el horario, días y profesor de un curso; finalmente, *edificio* y *num* son como en la base de datos *Salones*.

La extensión de esta base de datos se encuentra en la tabla 6.8.

Tabla 6.8.

<i>Cursos</i>								
<i>lice</i>	<i>clave</i>	<i>grupo</i>	<i>creds</i>	<i>horario</i>	<i>días</i>	<i>prof</i>	<i>edificio</i>	<i>núm</i>
mat	7	4010	10	10 a 11	lu ma mi ju vi	AMV	O	106
bio	7126	5050	12	12 a 15	lu ju	ABR	A	25
bio	7230	5010	12	7 a 10	ma vi	RRR	B	10
act	1167	6120	10	7 a 8	lu ma mi ju vi	JVA	O	106
act	1065	6041	10	8 a 9	lu ma mi ju vi	MCC	O	110
cc	287	7020	10	4 a 5:30	lu mi	FSC	T	218
ct	1040	1010	12	12 a 14	ma ju vi	OPC	B	12
fis	2010	8060	9	4 a 5:30	lu ju	MLF	O	130
act	987	1021	10	19 a 20	lu ma mi ju vi	AMR	O	110
bio	451	5010	9	16 a 19	lu ju	JMM	B	10
fis	322	8040	9	7 a 8:30	lu mi vi	AZR	O	130

Ejemplo 6.78. Si queremos obtener los cursos que tienen 10 créditos aplicamos la operación de selección $select(Cursos, creds=10)$, que construye una tabla que tiene únicamente a los renglones que tienen 10 en el atributo *creds* –ver tabla 6.9.

Tabla 6.9.

<i>select(Cursos, creds, 10)</i>								
<i>lice</i>	<i>clave</i>	<i>grupo</i>	<i>creds</i>	<i>horario</i>	<i>días</i>	<i>edificio</i>	<i>núm</i>	<i>prof</i>
mat	7	4010	10	10 a 11	lu ma mi ju vi	O	106	AMV
act	1167	6120	10	7 a 8	lu ma mi ju vi	O	106	JVA
act	1065	6041	10	8 a 9	lu ma mi ju vi	O	110	MCC
cc	287	7020	10	4 a 5:30	lu mi	T	218	FSC
act	987	1021	10	19 a 20	lu ma mi ju vi	O	110	AMR

Ejemplo 6.79. Queremos ahora obtener la información de cursos dada por número de grupo y los días de clase. Esto se obtiene mediante la operación de proyección $P_{3,6}(Cursos)$, puesto que el tercer atributo corresponde al número de grupo y el sexto a los días. Se encuentra en la tabla 6.10 a continuación.

Tabla 6.10. $P_{3,6}(Cursos)$

$P_{3,6}(Cursos)$			
<i>grupo</i>	<i>días</i>		<i>grupo</i> <i>días</i>
4010	lu ma mi ju vi		1010 ma ju vi
5050	lu ju		8060 lu ju
5010	ma vi		1021 lu ma mi ju vi
6120	lu ma mi ju vi		5010 lu ju
6041	lu ma mi ju vi		8040 lu mi vi
7020	lu mi		

Ejemplo 6.80. Ahora deseamos unir la información de las bases de datos *Cursos* y *Salones*, para obtener la información necesaria para publicar los horarios al inicio del semestre. Observamos que esto es posible, pues las últimas dos columnas de la primera base coinciden con las primeras dos columnas de la segunda, las cuales corresponden a los atributos *edificio* y *núm*. Esto implica que la operación *Join* se aplicará con el parámetro $p = 2$. El resultado es la tabla 6.11 en la siguiente página.

Tabla 6.11.

<i>Join₂(Cursos, Salones)</i>											
<i>lice</i>	<i>clave</i>	<i>grupo</i>	<i>creds</i>	<i>horario</i>	<i>días</i>	<i>prof</i>	<i>edificio</i>	<i>núm</i>	<i>tipo</i>	<i>capacidad</i>	<i>equipos</i>
mat	7	4010	10	10 a 11	lu ma mi ju vi	O	106	AMV	salón	64	F
bio	7230	5010	12	7 a 10	ma vi	ABR	B	10	lab	25	F
bio	7230	5010	12	7 a 10	ma vi	RRR	B	10	lab	25	F
act	1167	6120	10	7 a 8	lu ma mi ju vi	JVA	O	106	salón	64	F
fis	2010	8060	9	4 a 5:30	lu ju	MLF	O	130	salón	32	F
bio	451	5010	9	16 a 19	lu ju	JMM	B	10	lab	25	F
fis	322	8040	9	7 a 8:30	lu mi vi	AZR	O	130	salón	32	F

Terminamos con un ejemplo que combina varias operaciones.

Ejemplo 6.81. Queremos extraer información acerca del grupo, los días y horas a las que está asignado un salón registrado, junto con su capacidad, y si tiene o no computadoras. ¿Cuáles son las operaciones a realizar?

Solución: Primero hacemos una proyección en la tabla *Cursos* sacando las columnas que nos interesan, *grupo*, *horario*, *días* y *núm*. Esta operación nos arroja la tabla 6.12, que podemos ver a continuación.

Tabla 6.12.

<i>P_{3,5,6,8}(Cursos)</i>			
<i>grupo</i>	<i>horario</i>	<i>días</i>	<i>núm</i>
4010	10 a 11	lu ma mi ju vi	106
5050	12 a 15	lu ju	25
5010	7 a 10	ma vi	10
6120	7 a 8	lu ma mi ju vi	106
6041	8 a 9	lu ma mi ju vi	110
7020	4 a 5:30	lu mi	218
1010	12 a 14	ma ju vi	12
8060	4 a 5:30	lu ju	130
1021	19 a 20	lu ma mi ju vi	110
5010	16 a 19	lu ju	10
8040	7 a 8:30	lu mi vi	130

Elegimos también de la base de datos *Salones*, mediante proyección, las columnas correspondientes a *núm*, *capacidad* y *equipo*, que se encuentra en la tabla 6.13, a continuación.

Tabla 6.13.

$P_{2,4,5}(\textit{Salones})$		
<i>núm</i>	<i>capacidad</i>	<i>equipo</i>
106	64	F
130	32	F
215	30	T
8	70	F
25	25	F
10	25	F

Para nuestro propósito original necesitamos de la operación *Join* en las dos tablas con parámetro $p = 1$. El resultado es la tabla 6.14 en la página que sigue.

Tabla 6.14.

$Join_1(P_{3,5,6,8}(\textit{Cursos}), P_{2,4,5}(\textit{Salones}))$					
<i>grupo</i>	<i>horario</i>	<i>días</i>	<i>núm</i>	<i>capacidad</i>	<i>equipo</i>
4010	10 a 11	lu ma mi ju vi	106	64	F
5010	7 a 10	ma vi	10	25	F
5010	7 a 10	ma vi	10	25	F
6120	7 a 8	lu ma mi ju vi	106	64	F
8060	4 a 5:30	lu ju	130	32	F
5010	16 a 19	lu ju	10	25	F
8040	7 a 8:30	lu mi vi	130	32	F

Esta sección ha sido una pequeña muestra de la teoría de bases de datos relacionales. Para profundizar sugerimos consultar un libro particular de este tema, por ejemplo [11].

Ejercicios

6.8.1.- Tenemos cuádruplas en una relación de índice 4, que representan los atributos de los libros publicados: (título, ISBN, fecha de publicación, número de páginas). ¿Cuál es una posible llave primaria?

6.8.2.- Tenemos quintuplas en una relación de índice 5 que contiene (nombre, dirección, RFC, ciudad, estado) de todos los habitantes del país.

- ¿Cuál podría ser una llave primaria para esta relación?
- ¿Bajo qué condiciones la pareja (nombre, dirección) podría ser una llave compuesta?
- ¿Bajo qué condiciones la tripleta (nombre, dirección, ciudad) podrían configurar una llave compuesta?

6.8.3.- Tenemos la tabla 6.15 de los vuelos de compañías aéreas. Supongamos que el contenido ya no se va a modificar. Encuentre una llave compuesta de dos atributos que incluya al nombre de la Aerolínea.

Tabla 6.15. Ejercicio 6.8.3

Tabla: Vuelos				
Aerolínea	Núm. Vuelo	Puerta	Destino	Hora
Viva	122	12	Acapulco	08:10
Aerobús	221	22	Tapachula	08:17
Aerobús	122	11	Mazatlán	08:22
Aerobús	323	12	La Paz	08:30
Viva	212	08	Acapulco	08:47
Aerobús	122	22	Tapachula	09:10
Viva	212	08	Acapulco	09:44

6.8.4.- ¿Por qué para definir una llave primaria o compuesta hemos tenido que suponer que la extensión de una base de datos ya no se va a modificar?

6.8.5.- ¿Qué se obtiene cuando se aplica la proyección $P_{2,5}$ a la relación

$$R = \{(a, b, c, d, e), (f, g, h, i, j), (a, g, c, i, e)\}?$$

6.8.6.- Supongamos que tenemos una tabla de empleados de una empresa cuya intención es $N \times E \times G \times C \times Em \times EM$, donde los atributos significan lo siguiente:

N	Nombre del empleado	C	Número de hijos
E	Edad del empleado	Em	Edad mínima de los hijos
G	Género del empleado	EM	Edad máxima de los hijos

En el caso de edades mínimas y máximas, el atributo valdrá cero cuando no aplique.

Tenemos una segunda tabla, la cual cuenta con un registro por cada hijo de un empleado, con intención $n \times eh \times N$, donde el significado de los atributos es:

n	Nombre del hijo	N	Nombre del empleado
eh	Edad del hijo		

¿Qué operación u operaciones debemos realizar para obtener a todos los empleados hombres que tengan hijos menores de 10 años?

6.8.7.- Consideremos las tablas con las inscripciones de los estudiantes y la asignación de salones a los distintos grupos, que se encuentran en la figura 6.15.

Figura 6.15. Ejercicio 6.8.7: Tablas de inscripciones de estudiantes y de asignación de salones

Tabla: Inscripciones			Tabla: Asignación de salones			
Nombre	Carrera	Curso	Carrera	Curso	Salón	Hora
Avella	Matemáticas	3045	Actuaría	1333	P105	8:00
Avella	Matemáticas	3054	C. de la Comp.	7001	O127	8:00
Aguilar	Actuaría	1050	Matemáticas	3054	T001	10:00
Medina	Actuaría	1128	Actuaría	1128	P105	9:00
Medina	Actuaría	1217	Matemáticas	3045	O122	11:00
Medina	Actuaría	1333	Física	2223	O214	7:00
Morales	Física	2543	C. de la Comp.	7051	T212	10:00
Padilla	C. de la Comp.	7001	Actuaría	1231	P105	16:00
Padilla	C. de la Comp.	7051	Actuaría	1132	O127	18:00
Padilla	C. de la Comp.	7047	Matemáticas	3128	T001	15:00
Vázquez	Actuaría	1231	Física	2543	P105	14:00
Vázquez	Actuaría	1132	Actuaría	1217	O122	12:00
Zuazua	Matemáticas	3128				

- ¿Cuál es el resultado de aplicar la proyección $P_{1,2}$ a la tabla de inscripciones?
- ¿Cuál es el resultado de aplicar la unión $Join_2$ a las tablas de inscripciones y de asignación de salones?
- ¿Qué operación u operaciones debemos hacer para obtener los alumnos inscritos en cursos que se llevan a cabo antes de las 11:00 horas?
- ¿Qué operación u operaciones debemos realizar para obtener los estudiantes de Actuaría que toman clase en el salón P105?

6.8.8.- Consideremos las tablas de la figura 6.16 –ver siguiente página–, referentes al inventario de una bodega.

- Construye la tabla resultado de la operación $Join_2$.
- ¿Qué operación u operaciones debemos realizar para que tengamos una lista de aquellos proyectos a los que les faltan partes en existencia y a qué proveedor solicitarlas?
- ¿Qué operación u operaciones debemos realizar para saber cuántos y cuáles proyectos pueden ser completados con las partes que hay en existencia?

Figura 6.16. Ejercicio 6.8.8

Tabla: Partes necesarias				Tabla: Partes en existencia			
Proveedor	# Parte	Proyecto	# Nec.	# Parte	Proyecto	Cant.	Cód. color
23	1092	1	5	1001	1	14	8
23	1101	3	2	1092	1	2	2
23	9048	4	8	1101	3	1	1
31	4975	3	9	3477	2	25	2
31	3477	2	3	4975	3	6	2
32	6984	4	2	6984	4	10	1
32	9191	2	3	9048	4	12	2
33	1001	1	25	9191	2	80	4

6.8.9.- Sean R una relación n -aria y C_1, C_2 condiciones acerca de n -adas. Muestre que

a) $s_{C_1 \wedge C_2}(R) = s_{C_1}(s_{C_2}(R))$.

b) $s_{C_1}(s_{C_2}(R)) = s_{C_2}(s_{C_1}(R))$.

6.8.10.- Sean R, S relaciones n -arias y C una condición acerca de n -adas. Muestre que

$$s_C(R \setminus S) = s_C(R) \setminus s_C(S).$$

6.8.11.- Sean R, S relaciones n -arias y C una condición acerca de n -adas. Muestre que

$$s_C(R \cup S) = s_C(R) \cup s_C(S).$$

6.8.12.- ¿Será válida la propiedad del ejercicio anterior en el caso de la intersección de relaciones n -arias?

6.8.13.- Sean R, S relaciones n -arias. Muestre que

$$P_{i_1, \dots, i_m}(R \cup S) = P_{i_1, \dots, i_m}(R) \cup P_{i_1, \dots, i_m}(S)$$

6.8.14.- ¿Será válida la propiedad del ejercicio anterior en el caso de la intersección de relaciones n -arias?

Bibliografía

- [1] Kees Doets and Jan van Eijck. *The Haskell Road to Logic, Maths and Programming*. King's College Publications, London, 2004.
- [2] John A. Dossey, Albert D. Otto, Lawrence E. Spence, and Charles Vanden Eynden. *Discrete Mathematics*. Pearson/Addison-Wesley, 5th edition, 2006.
- [3] Rowan Garnier and John Taylor. *Discrete Mathematics, Proofs, Structures, and Applications*. CRC Press, 3rd edition, 2010.
- [4] Judith L. Gersting. *Mathematical Structures for Computer Science*. Computer Science Press, W.H. Freeman and Company, 6th edition, 2006.
- [5] Winifried Karl Grassman and Jean-Paul Tremblay. *Logic and Discrete Mathematics, A Computer Science Perspective*. Prentice-Hall Inc., 1996.
- [6] David Gries and Fred B. Schneider. *A Logical Approach to Discrete Mathematics*. Springer-Verlag, 1994.
- [7] Jerold W. Grossman. *Discrete Mathematics, An Introduction to Concepts, Methods and Applications*. Macmillan Publishing Company, 1990.
- [8] James L. Hein. *Discrete Structures, Logic, and Computability*. Jones and Bartlett Publishers, 3rd edition, 2010.
- [9] Thomas Koshy. *Discrete Mathematics with Applications*. Elsevier Academic Press, 2004.
- [10] Kenneth H. Rossen. *Discrete Mathematics and its Applications*. McGraw Hill, 7th edition, 2011.
- [11] Stephan Stanczyk, Bob Champion, and Richard Leyton. *Theory and Practice of Relational Databases*. Taylor & Francis, London, 2nd edition, 2001.

Índice analítico

- , 264
- $::=$, 9
- orden parcial
 - diagramas de Hasse, 303
- \models , 32
- activación, 150
 - por transición, 150
 - por valor, 150
- alcance, 48, 175
 - de cuantificación, 175
- álgebra
 - booleana, 104
- álgebra booleana, 325, 328
- algoritmo
 - para clasificar fórmulas, 99
 - para consecuencia lógica, 100
 - para satisfacibilidad, 100
 - para tautologías, 98
 - Warshall
 - relaciones binarias, 274
- análisis sintáctico, 45
- \wedge , 104
- antisimetría, 298
- árbol, 11
 - binario, 229, 247
 - resultado, 12
- archivo
 - directorio, 208
- arco, 261
- argumento
 - correcto, 33, 69, 203
 - deductivo, 33
 - inductivo, 33
 - lógico, 33
 - lógico, 17
 - sólido, 18
- aridad, 172, 328
- asociatividad, 30, 40
- atributo, 329
- autómatas finitos, 224
- axioma, 83
 - de inducción, 216
 - de Peano, 215
- base de datos, 329
 - relacional, 329
- bases de datos
 - relacionales, 328
- bicondicional, 28, 55
- buen orden, 317
- cadena, 10
- cálculos deductivos, 83
- capa, 141
- cardinalidad, 317
- casamiento de patrones, 230
- cerradura
 - definición en relaciones binarias, 270
 - reflexiva, 271
 - simétrica, 271
 - transitiva, 271
- circuito
 - estable, 144
 - inestable, 144
 - secuencial, 137
- circuito combinatorio
 - inversor, 137
 - multiplexor, 138

- circuitos
 - digitales, 104
- circuitos digitales, 325
 - combinatorios, 136
 - secuenciales, 142
- circuitos lógicos
 - simplificación, 113
- circuitos lógicos
 - tabletas (*chips*), 135
- circuitos secuenciales
 - flip flop, 151
 - latches, 143
- clase de equivalencia, 286
- codificador
 - de prioridad, 142
- comparables, 300
- completo
 - sistema de derivación, 87
- compuerta
 - nand, 115
 - nor, 115
- compuertas, 106
 - lógicas, 105
 - síncronas, 151
- compuertas lógicas, 106
- conclusión, 33
- conclusión, 17
- condición de carrera, 145
- condicional, 26
- condiciones *don't care*, 139
- conectivo, 21
 - principal, 48
- conexión
 - lema de, 321
- congruencia
 - relación de, 284
- conjunción, 25, 86
- conjunto cociente, 286
- $\wp(X)$, 306
- conjuntos infinitos numerables, 213
- conmutatividad, 29, 57
- consecuencia lógica, 69, 75, 80
- constante, 4
- contador digital, 158
- contadores
 - asíncronos, 164
 - tabletas para, 162
- contingencia, 31
- contradicción, 32, 68
- contradicción, 31
- contraejemplo, 78, 80
- contrapositiva, 28
- contrarrecíproca, 28
- coordenadas
 - orden parcial de, 307
- correcto, 18
- cuantificación
 - alcance, 175
 - existencial, 174
 - universal, 174
 - vacua, 201
 - variable de la, 175
- cuantificador, 171
- De Morgan
 - en latices, 324
- decodificador, 140
- deducción formal, 83
- definición
 - recursiva, 227
 - de funciones, 230
 - general, 227
- derivable, 83
- derivación, 10, 83
- descendiente
 - relación, 227
- diagrama de bloque
 - decodificador, 140
 - multiplexor, 139
 - semi sumador, 109
- digráfica, 261
- dilema constructivo simple, 77, 81
- distributividad

- en lógica de predicados, 201
- disyunción, 25, 86
- \div
 - cociente entero, 259
- divisibilidad
 - orden, 328
- dominante, 30
- dominio, 328
 - bien fundado, 213
 - de interpretación, 191
 - de la relación, 329
- don't care conditions, 130
- E^x_R , 41
- $E[x := R]$, 41
- edge triggered, 150
- elemento
 - extremo, 308
- elemento identidad, 30
- elemento neutro, 30
- eliminación
 - de \wedge , 84
 - de \leftrightarrow , 84
- emparejamiento de patrones, 230
- n -adas, 253
- enunciado, 176
- equivalencia, 28, 55, 87
 - lógica, 55, 199
 - álgebra de, 64
 - relación de, 282
- esquema, 46
 - básico, 49
 - instanciar un, 46
- estado, 20, 38, 67, 142, 143
 - evaluación en un, 38
- evaluación
 - de una expresión, 38, 44
- expresión
 - aritmética, 6, 228
 - lógica, 45
- extensión, 329
- extremo
 - elemento, 308
- $f^{(n)}$, 172
- factorial, 230
- falacia, 87
- flanco ascendente, 150
- flanco descendente, 150
- flip flop, 143, 151
 - tipo D , 154
 - tipo T , 157
 - tipo JK , 155
 - toggle, 157
- flip flop* (circuitos biestables), 143
- forma normal, 104
 - conjuntiva, 104
 - disyuntiva, 105
- formal normal
 - disyuntiva, 104
- fórmula, 45
 - asociada al argumento lógico, 33
 - bien construida, 228
 - cerrada, 176
 - cuantificada, 174
 - insatisfacible, 68
 - no satisfacible, 68
 - válida, 68
- fórmula atómica
 - con predicados, 173
- fuertemente tipificado, 208
- función sucesor, 214
- funcional, 172
- fórmula
 - bien formada, 45
- generalización
 - existencial, 204
 - universal, 203
- grado, 328
- gramática, 10
 - de la lógica
 - de predicados, 174, 175
 - de los términos, 172

- formal, 9
- gramática
 - de la lógica
 - proposicional, 22
- habilita, 137
- hojas, 12
- \mathcal{I} , 67
- I_A , 260
- idempotencia, 31
 - en latices, 324
- identificador, 46
- implicación, 26, 87
 - contrapositiva, 28
 - contrarrecíproca, 28
 - recíproca, 28
- implicante
 - función booleana, 131
- incógnitas, 38
- incomparables, 300
 - elementos
 - orden, 309
- indecidibilidad
 - teorema de, 203
- índice, 172, 328
- inducción, 213
 - cambio de base, 219
 - completa, 221
 - en fórmulas, 239
 - en listas, 237
 - en árboles, 242
 - estructural, 236
 - fuerte, 221
 - matemática, 213
- inferencia
 - regla de, 53, 74
- ínfimo, 314–316
 - $\sqcap P$, 314
 - $\inf P$, 314
- instanciación
 - existencial, 204
 - universal, 203
- instanciar
 - un esquema, 46
- intención
 - relación, 329
- intención, 329
- interpretación, 67, 74, 80
 - función de, 67
- interpretación, 67
- intratable, 272
 - algoritmo de Quine-McCluskey, 135
- introducción
 - de \wedge , 84
 - de \vee , 84
- inversa, 28
- join, 332
- juicio
 - afirmativo, 208
 - aristotélico, 182
 - existencial
 - afirmativo, 182, 207
 - negativo, 182
 - universal
 - afirmativo, 182, 207
 - negativo, 182
- latch, 143, 144
 - SR, 145
- latch–SR, 148
- latches* (circuitos de pestillo), 143
- latices
 - complementadas distributivas, 325
- latiz
 - acotada, 322, 323, 328
 - complementada, 323, 328
 - complemento, 323
 - completa, 327
 - conjunto potencia, 321
 - desigualdades distributivas, 327
 - distributiva, 324, 328
 - divisibilidad, 320

- ley de cancelación, 328
- orden total, 322
- propiedades, 322
- Leibniz
 - regla de, 57, 64
- lenguaje
 - de la lógica
 - de predicados, 171
 - de la lógica
 - proposicional, 17
 - formal, 3
 - fuertemente tipificado, 208
 - natural, 3
- level triggered, 150
- lista
 - concatenación, 231
 - finita, 228
 - longitud, 231
 - reversa, 231
 - tipo, 209
- literal, 90, 94
 - complementaria, 94
- llave compuesta, 330
- llave primaria, 330
- lógica
 - de predicados de primer orden, 171
- lógica proposicional
 - latiz, 326
- línea de dato, 137
- lógica
 - proposicional, 19
- mapas de Karnaugh, 118
 - n variables, 120
 - 2 variables, 118
 - 4 variables, 123
- matrices
 - cuadradas, 262
- máxima cota inferior, 314
- maximal
 - elemento, 308
- memoria, 142
- meta expresión, 32
- metalenguaje, 32
- método
 - analítico, 49
 - generador, 49
- micromundo, 185
 - de cubos, 186
 - de figuras geométricas, 186
- minimal
 - elemento, 308
- minimización, 131
- mintérmino, 105, 119
 - esencial, 130
- mintérmino, 112, 118, 119
- mod
 - residuo entero, 259
- modelo, 4, 68, 89
 - de una fórmula, 89, 98
 - matemático, 4
- modelo relacional, 329
- modus
 - ponens, 53, 74
 - tollens, 76
- máximo
 - elemento, 308
- mínima cota superior, 314
- mínimo
 - elemento, 308
- nand, 37
- naturales
 - definición recursiva, 214
 - exponenciación, 230
 - producto, 215
 - suma, 215
- n -adas, 253
- negación
 - en lógica de predicados, 184
- negación, 24, 87
- nivel
 - en un circuito, 139
- no determinismo, 80

- notación
 - infija, 5
 - prefija, 5
 - sufija o polaca, 5
- numerable, 213
- números naturales, 213
- o exclusivo (exclusive or), 257
- ocurrencia
 - presencia, 41
- operador, 4
 - binario, 5
 - de cuantificación, 171
 - n-ario, 5
 - unario, 4
- \vee , 104
- orden
 - lexicográfico, 306
 - lineal, 310
 - parcial
 - antilexicográfico, 307
 - de coordenadas, 307
 - suma, 307
 - suma lineal, 307
 - relación de, 297
 - total, 310
 - cadena, 310
- orden canónico, 251
- orden parcial
 - retícula (*lattice*), 319
- ordenamiento topológico, 311
- palabras, 10
- palíndroma, 244
- pareja, 252
- parejas, 253
 - ordenadas, 250
- partición, 285
- partición trivial, 286
- partición total, 286
- patrón, 230
- Peano
 - axioma de inducción, 216
 - axiomas de, 215
- pila
 - tipo, 209
- poliominó, 225
- precedencia, 39
- predicado, 168
 - calificador, 207
 - tablas para, 194
- predicados
 - lógica de, 167
- premisa, 17, 33
- prenexación, 202
 - de cuantificadores, 202
- presencia
 - de variables en expresiones, 41
- producciones, 9
- producto cartesiano, 250, 251
- propiedades, 170
- proposición, 4, 45
 - compuesta, 49
 - lógica, 4
- proposición, 19
 - atómica, 18, 21
 - compuesta, 21
 - definición, 21
- proyección
 - bases de datos, 331
- prueba, 83
- punto fijo
 - de una función, 266
 - potencias, 272
- Quine-McCluskey
 - minimización de circuitos, 130
- \mathcal{R} , 83
- raíz, 11
- rama
 - de un tableau, 93
- rango, 48
- razonamiento

- ecuacional, 57, 64
- recursión, 213
- recíproca, 28
- refinamiento, 296
- reflexividad, 57
- registro, 329
- regla
 - de reescritura, 9
 - recursiva, 227
 - sintáctica, 10
- regla de inferencia, 53
 - α , 95
 - β , 95
 - casos simple, 84
 - de Leibniz, 57
 - eliminación
 - de \wedge , 84
 - de \leftrightarrow , 84
 - generalización
 - existencial, 204
 - universal, 203
 - inconsistencia, 84
 - instanciación
 - existencial, 204
 - universal, 203
 - introducción
 - de \wedge , 84
 - de \vee , 84
 - introducción de \leftrightarrow , 84
 - modus
 - ponens, 84
 - tollens, 84
 - para tableaux, 95
 - σ , 96
 - silogismo
 - disyuntivo, 84
 - hipotético, 84
- relación
 - binaria, 255
 - asimétrica, 268
 - identidad, 260
 - de equivalencia, 282
 - n -aria, 255
 - terciaria, 255
- relaciones, 170
 - binarias, 258
 - cerradura, 270
 - propiedades, 266
 - composición, 264
 - diferencia, 257
 - intersección, 257
 - n -arias, 328
 - potencia, 264
 - unión, 257
 - XOR, 257
- relación, 255
 - binaria
 - antirreflexiva, 267
 - antisimétrica, 267
 - gráfica dirigida, 261
 - matriz booleana, 262
 - reflexiva, 266
 - simétrica, 267
 - tablas, 260
 - transitiva, 268
 - universal, 260
 - de equivalencia, 250
 - de orden, 250
 - estricta, 298
 - no estricta, 298
- relación identidad, 262
- relación universal, 262
- reloj, 149
 - pulso, 151
- representante
 - de una clase, 286
- reset
 - restablecer, 145
- restador completo, 116
- retardo, 141
- retardo (*delay*), 137
- retroalimentación (*feedback*), 142

- ripple counters, 165
- satisface, 68
- satisfacible, 68
- select
 - base de datos, 331
- semántica, 5, 23, 67, 214
- semi restador, 116
- semi sumador (*half adder*), 109
- set, 145
 - establecer, 145
- señal de control , 137
- silogismo
 - hipotético, 72, 81, 84
- símbolo
 - no terminal, 10
 - terminal, 10, 13
- sintaxis, 5, 214
- subexpresión, 10
- suma de productos, 105
- suma lineal
 - orden parcial, 307
- sumador completo (*full adder*), 111
- supC, 316
- supM, 316
- supremo, 314, 315
 - $\sqcup P$, 314
 - sup P , 314
- sustitución
 - por la izquierda, 10
 - propiedad de, 52
 - textual, 41, 57
 - variables escondidas en, 43
- sustitución textual, 42
 - de listas, 43
 - simultánea, 43
- síncrono, 150
- tabla
 - base de datos, 329
- tabla característica, 145
- tablas de verdad, 24
- tableaux, 90
 - algoritmos con, 98
 - contradicción, 95
 - fórmula contingente, 95
 - semánticos, 89
 - tautología, 95, 99
- tautología, 52, 55, 68
- tautología, 31
- teorema
 - de la deducción, 87
- teoría
 - de la demostración, 83
 - de modelos, 83
- tercero excluido, 32, 87
- término, 172
- terna, 252
- terna pitagórica, 257
- ternas, 253
- tipado, 208
 - dinámico, 208
- tipo, 207
 - abreviado, 208
- toggle, 155
- transitividad, 57, 297
- tripletas, 253
- tupla, 329
- universo
 - de discurso, 168, 191
- vacuidad, 267
- valor, 20
 - forzar un, 78
- variable, 4, 10
 - acotada, 176
 - libre, 176
 - ligada, 176
 - proposicional, 21
- variable artificial, 176
- verdad
 - noción de, 195

Warshall

algoritmo de
relaciones binarias, 274

XOR (o exclusivo), 257

x^*

en latices, 324

\mathbb{Z}_n

estructura de enteros módulo n , 288

álgebra relacional, 330

operaciones, 330

MATEMÁTICAS DISCRETAS
editado por la Facultad de Ciencias de la
Universidad Nacional Autónoma de México,
se terminó de imprimir el 20 de junio de 2016 en los
talleres de Impresos Vacha S. A. de C. V.
José María Bustillos 59, Col. Algarín
C.P. 06880. Ciudad de México.

El tiraje fue de 500 ejemplares en papel Cremy book de 60 g.
En su composición se utilizó tipografía Computer modern
de 16:18 y 12:14 puntos de pica.
Tipo de impresión: offset

El cuidado de la edición estuvo a cargo de
Patricia Magaña Rueda