Estructuras Discretas

Wilber Ramos Lovón

Prólogo

Escribí este libro con el fin de ofrecer una introducción de los Fundamentos Matemáticos para la Ciencia de la Computación, que estimula a pensar que las Matemáticas no son sólo una deducción de teoremas, sino también una metodología de modelamiento.

Por lo tanto, la cuestión es de artícular matemáticamente el fenómeno de la Computación. Se que ésta es una tarea compleja, que demanda reflexiones a propósito del fenómeno de la computación y de la metodología matemática; comprensiones difíciles, que en gran medida aún deben ser elaboradas; pero que hoy, cuando la computación precisa una Ingeniería de corte científico, son ineludibles.

El término "Matemáticas Discretas"se utiliza extensamente para describir un tipo de Matemáticas donde no tiene cabida propiedades tales como cercanía y suavidad, ideas fundamentales del cálculo. Se trata de un curso híbrido, cuyo contenido fundamental es de matemáticas, aunque muchas de sus aplicaciones pertenecen a la Ciencia de la Computación.

Para la comprensión integral de la ciencia de la computación se requiere comprender muchos tópicos de Matemáticas Discretas, entre los cuales, en un nivel de introducción, creo que los mas adecuados son: algoritmos, conjuntos, inducción matemática, relaciones y estructuras de orden.

Agradezco a la Universidad Católica San Pablo por su incondicional apoyo. Fue Luis Díaz Basurco, quien sugirió en primer lugar que debería escribir un texto de Matemáticas Discretas para estudiantes del segundo semestre del Programa Profesional de Ingeniería Informática de la USP que el entusiastamente dirige. Una vez puesta en marcha mi proyecto (por su puesto) duró mas de lo que se esperaba. La composición final fue efectuada por Abigail Parisaca Vargas y Liliana Mamani Sánchez, que trabajaron en LaTeX apoyados por Jesús Mena Chalco en la composición de gráficos. A todos ellos mi más sincero agradecimiento, agradecimiento que hago extensivo a todos mis alumnos que han - ¿debería decir sufrido o disfrutado? - de mis cursos de Matemáticas Discretas y que han contribuído decisivamente a su depuración y mejora.

Índice general

Capítulo 1

Lógica de predicados

1.1. Componentes sintácticos de la Lógica de Predicados

Observación 1.1.1. Observemos los siguientes items:

a) El desarrollo del cálculo proposicional se basa en entidades matemáticas representativas de unidades de información, cuya estructura se contempla como un todo, sin diferenciar sus componentes. Este planteamiento nos permite representar matemáticamente determinadas estructuras deductivas, que sin embargo son correctas en el lenguaje usual. Como ejemplo considere las siguientes frases:

   - Antonio no es tonto.
   - Sólo los tontos se dejan engañar por los vendedores ambulantes.
   - Antonio se deja engañar por Juan.

   De estas frases uno debería poder concluir que Juan no es vendedor ambulante.

b) Los predicados se utilizan para describir ciertas propiedades o relaciones existentes o individuos existentes, las entidades que se relacionan de esta forma se denominan términos. Además de términos y predicados se usan los cuantificadores.

c) La veracidad de la frase: Antonio no es tonto, sólo se puede ponderar en un cierto contexto.

Definición 1.1. El universo de discurso o dominio es la colección de todas las personas, ideas, símbolos, estructuras de datos y demás que afectan al argumento lógico que se esta considerando. Los elementos del dominio se denominan términos.

- La verdad de una frase puede depender del dominio seleccionado.
- Para evitar las cosas triviales se considera que el dominio es diferente del vacío.

- Para hacer alusión a un término en particular se emplean individuales.

Observación 1.1.2. En lógica de predicados

a) La simbolización, responde a las siguientes preguntas:

   - ¿Qué se afirma?
   - ¿De quién se afirma?

   La primera pregunta determina el predicado y la segunda los términos. Así la frase:

   - Mayte es bonita.

   |Es bonita| es el predicado y |Mayte| es el término. Puede haber proposiciones con varios términos, por ejemplo:

   - Ica está entre Lima y Arequipa.

   El predicado es |__ está entre __ y __| y los términos son |Ica , Lima, Arequipa|. El número de términos se denomina aridad del predicado. Los predicados de aridad n suelen denominarse predicados de n cifras. Los predicados de una sola cifra suelen denominarse propiedades.

b) Para simbolizar predicados se utiliza la notación funcional $P(t_1, t_2, \ldots, t_n)$ donde P es el nombre del predicado y $t_1, t_2, \ldots, t_n$, no debe confundirse con la notación de las variables de término, en este caso t_i representa la plaza y a ocupar en el predicado por una variable x_i o una constante a_i de término, se denominan por tanto variables nominales o de plaza

Ejemplo 1.1.1. En la oración: Mayte es bonita

Bonita (Mayte)	Bonita (t_1)	Bonita (x_1)
Constante	**Plaza**	**Variable**

$P(t_1, t_2, \ldots, t_n)$ no debe considerarse una expresión en tanto no se ocupen sus plazas por variables o constantes de términos.
Esto puede hacerse en dos formas:

- Por sustitución
- Por cuantificación

Por sustitución En este caso se asigna a cada plaza de predicado un símbolo de término que puede ser constante o variable.

Ejemplo 1.1.2. $P(x_1, x_2, a_3, \ldots, x_n)$ las variables que aparecen así se denominan variables libres.

Por cuantificación En este caso se asigna a cada plaza un conjunto de elementos del dominio, caben dos posibilidades:

1. Si se asigna a una plaza todos los elementos del dominio para todo x_i:

$$\forall x_i \ P(x_1, x_2, a_3 \dots x_1 \dots x_n)$$

2. Si se asigna a una plaza un subconjunto del dominio no especificado.

$$\exists x_i \ P(x_1, x_2, a_3 \dots x_i \dots x_n)$$

Las variables afectadas por cuantificadores se denominan variables ligadas.

Ejemplo 1.1.3. En el dominio de los alumnos de Est. Disc. Simbolizar en lógica de predicados las siguientes frases.

a) Ursula está entre Oscar y Sandra.

b) Un alumno cualquiera de E.D. está entre Manuel y Diana.

c) Un alumno de E.D. está entre Manuel y Diana

d) Todos los alumnos de E.D. están entre Manuel y Diana.

e) Algunos alumnos de E.D. están entre Manuel y Diana.

Solución

$$\begin{array}{ccc} P(t_1, t_2, t_3) = t_1 & \text{está entre} & t_2 \text{ y } t_3 \\ \underline{\quad\quad} & \text{está entre} & \underline{\quad\quad} \text{ y } \underline{\quad\quad} \end{array}$$

a) $P(U, O, S)$.

b) $P(X, M, D)$, Se puede cuantificar.

c) $P(X, M, D)$, No se puede cuantificar.

d) $\forall x \ P(X, M, D)$.

e) $\exists x \ P(X, M, D)$.

$\star$La aparición de x como libre en los casos d y c puede entenderse en dos sentidos:

1. En el sentido general (caso b), x representa un elemento genérico del dominio, en este sentido b y d tienen el mismo contenido de información.

2. En el sentido condicional (caso c), x es una forma de denominar en elemento incógnito del dominio. En este sentido x representa un elemento único desconocido.

Observación 1.1.3. El nombre de un predicado seguido por una lista de términos se denomina una formula atómica. Las formulas atómicas pueden combinarse mediante conexiones lógicas como si fuesen proposiciones. $\forall x$ y $\exists x$, deben tratarse como conexiones unarias los cuantificadores tienen una propiedad superior a la de todas las conexiones binarias.

Ejemplo 1.1.4. Simbolizar en lógica de predicados:

Todas las águilas vuelan alto.

Solución

- El dominio es el conjunto de todos los águilas
  $VA(t_1) = t_1$ vuela alto
  $\forall x VA(x)$
- El dominio es el conjunto de todas las aves águila $(t_1) = t_1$ es un águila.
  $\forall x(Aguila(x) \rightarrow VA(x))$

Definición 1.2. Sea A una expresión, y sea x una variable y t un término. Entonces $\int_t^x$ representa la expresión que se obtiene al sustituir todas las apariciones de x en A por t, esto se llama una particularización de A y se dice que t es un caso o instancia de x.

Ejemplo 1.1.5.

$$\int_a^x (P(x) \rightarrow Q(x)) \equiv P(a) \rightarrow Q(a)$$
$$\int_b^y (P(y) \vee Q(y)) \equiv P(b) \vee Q(b)$$
$$\int_y^a (Q(y)) \equiv Q(a)$$

Definición 1.3. Se dice que una expresión es una variante de $\forall x A$ si tiene la forma de $\forall y \int_y^x A$. Análogamente $\exists x A$ y $\exists y \int_y^x$ son variantes una de otra.

⋆Los cuantificadores pueden anidarse según se muestra en los ejemplos siguientes:

Ejemplo 1.1.6. Simbolizar en lógica de predicados:

a) Todo el mundo tiene alguien que sea su madre.

b) Hay alguien que conoce a todo el mundo.

c) Nadie es perfecto.

Solución
El dominio es el conjunto de todas las personas

a) $M(t_1, t_2) = t_1$ es madre de t_2.
   $\forall y,\ \exists x M(x, y)$

b) $K(t_1, t_2) = t_1$ conoce a t_2
   $\exists x\ \forall y K(x, y)$

c) $\forall x\ \neg P(x)$ $P(t_1)$ es perfecto
   $\neg \exists x P(x)$

- En español hay muchos cuantificadores tales como "unos pocos", "la mayoría"y "alrededor de la tercera parte"que resultan útiles en el lenguaje usual, pero que no son lo suficientemente precisos para utilizarlos en lógica de predicados.
- La particularización sólo afecta a variables libres. Concretamente si A es una expresión, particularización de A de x en t ($\int_y^x A$) sólo afecta a las apariciones libres de la variable x.

  Ejemplo 1.1.7.

  $$\int_y^x \forall x P(x) \equiv \forall x P(x)$$

- Dos cosas sólo serán idénticas si ambas se tratan idénticamente. Esto implica que si una variable aparece tanto libre como ligada, tenemos de hecho dos variables que casualmente poseen el mismo nombre.
- Podemos considerar que las variables ligadas son locales para el ámbito del cuantificador.

Ejemplo 1.1.8. Para el presente ejemplo:
Considere la frase: "Y tiene madre"
Dominio: Conjunto de todas las personas

$M(t_1, t_2) = t_1$ es madre de t_2
$\exists x M(x, y)$

No se puede particularizar cuando la variable esta ligada.

$$\int_y^x \exists x M(x, y) = \exists M(y, y)$$

Esto se llama una colisión de variables

Y es madre de y (absurdo)

Observación 1.1.4. En lógica de predicados

a) Si todos los argumentos de un predicado son constantes individuales, entonces la forma atómica resultante debe ser o bien verdadero o bien falso.

b) Todo método que asigne valores de verdad a todas las posibles combinaciones de individuos de un predicado se denomina una *Asignación del Predicado.*

Ejemplo 1.1.9. La tabla mostrada a continuación es una asignación para el predicado $M(t_1, t_2)$

$M(t_1, t_2)$	DAVID	JUANA	MARIA	PABLO
DAVID	F	F	F	F
JUANA	F	F	V	V
MARIA	F	F	F	F
PABLO	F	F	F	F

$M(J, M)$ es verdadero

Observación 1.1.5. Si todas las apariciones de x en una expresión A están ligadas, diremos que A no contiene a x libre. Si A no contiene a x libre, entonces el valor de verdad de A no cambia si x se particulariza con una constante individual. En este sentido A es independiente de x.

$$\frac{\forall P(x)}{P(Oscar)}$$

1.2. Interpretación y validez

Observación 1.2.1. Observemos que

a) Una interpretación de una frase debe contener información suficiente para determinar si la frase es V o F.

b) Una interpretación de una expresión lógica contiene los siguientes componentes.

- Tiene que haber un universo de discurso (dominio)
- Para cada individuo, tiene que haber una constante individual que se refiera exclusivamente a ese individuo y a ningún otro.
- Hay que asignar a cada variable libre una constante individual exclusiva. Tiene que haber una para cada predicado que se utilice en la expresión, incluyendo los predicados de aridad cero, una proposición tienen aridad cero.

Ejemplo 1.2.1.

Luis es estudioso	P	aridad cero
Luis es estudioso	$E(L)$	aridad uno
Llueve	$L()$	aridad cero

Que representan proposiciones

Ejemplo 1.2.2. Para las interpretaciones de:
$\forall x P(x)$, donde P es el predicado "ha pagado" se necesita una lista de clientes (dominio), supongamos que hay sólo 3 clientes, Franco, Monterrey y Alemán. Supongamos también la siguiente asignación para $P(x)$

	FRANCO	MONTERREY	ALEMÁN
$P(x)$	V	F	V

De donde el valor de $\forall x P(x)$ es F por que Monterrey no pago.

■

$\forall x P(x) \equiv P(a_1) \wedge P(a_2) \wedge \ldots \wedge P(a_n)$
$\exists x P(x) \equiv P(a_1) \vee P(a_2) \vee \ldots \vee P(a_n)$

Donde el dominio es finito con $a_1, a_2 \ldots\ldots$, en términos

- $$\begin{aligned}\neg\forall P(x) &\equiv \neg(P(a_1) \wedge P(a_2) \wedge \ldots \wedge P(a_n)) \\ &\equiv \neg P(a_1) \vee \neg P(a_2) \vee \ldots \vee \neg P(a_n) \\ &\equiv \exists x \neg P(x)\end{aligned}$$

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

Ejemplo 1.2.3. Consideremos la frase "Hay alguien que admira a todo el mundo".
El dominio consta de 3 personas: Rosa, Eduardo, Lourdes.

$A(t_1, t_2) = t_1$ admira a t_2
$\exists x \forall y A(x, y)$

Supongamos la siguiente asignación:

	R	E	L	$\forall y A(x,y)$	$\exists y A(x,y)$
R	V	V	F	F	V
E	F	F	V	F	V
L	F	V	F	F	V
$\forall y A(x,y)$	F	F	F	F	
$\exists y A(x,y)$	V	V	V	V	

Por lo tanto:

$$\exists x \forall y A(x, y)$$

Existe alguien que admire a todo el mundo: F

- Observe que las proposiciones

$\exists x \forall y A(x, y)$	Existe alguien que admire a todo el mundo
$\forall y \exists x A(x, y)$	Todos son admirados por alguien
$\forall y \exists x A(x, y)$	Todos admiran a alguien

  Son diferentes, esto es, el valor de verdad puede cambiar si se intercambian los cuantificadores universal y existencial.

- Aún cuando los cuantificadores universal y existencial no se pueden intercambiar, se tienen las siguientes leyes:

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

- Si R es cualquier proposición, la cual significa que es o bien verdadera o bien falsa sea cual fuera el individuo considerando, entonces

  $\forall x R \equiv \exists x R \equiv R$

Observación 1.2.2. Notemos lo siguiente:

a) Una expresión es válida si es verdadera en todas las interpretaciones. Una se denota con $\models A$.

b) Todas las tautologías son expresiones válidas

c) Una expresión A no es válida si existe una sola interpretación que haga a A falsa y "no A verdadera".

d) Si B es una expresión, entonces que toda interpretación que haga que B produzca verdadero, satisface B. Toda interpretación que satisface a B se denomina un modelo de B. Si B tiene un modelo, entonces se dice que b es viable. Por tanto una expresión A no es válida si $\neg A$ es viable

e) Una expresión B que no tenga ningún modelo se denomina CONTRADICTORIA.

f) Sean A y B expresiones, decimos que A es lógicamente equivalente a B si $A \leftrightarrow B$ es válida, en éste caso escribiremos $A \equiv B$ o $A \Leftrightarrow B$. Además decimos que A implica a B si $A \rightarrow B$ es válida y escribimos $A \Rightarrow B$.

g) Tal como en el cálculo proposicional, escribiremos $A_1, A_2, \ldots, A_n$ que en conjunto implica lógicamente a C.

h) Principales reglas de inferencia en lógica proposicional

$A, B \models A \wedge B$	Ley de combinación (LC)
$A \wedge B \models B$	Ley de simplificación (LS)
$A \Gamma A \wedge B$	Ley de Adición (LA)
$A, A \rightarrow B \models B$	Modus Ponens (MP)
$A \rightarrow B, B \rightarrow C \models A \rightarrow C$	Silogismo Hipotético (SH)
$A \vee B, \neg A \models B$	Silogismo Disyuntivo (SD)
$A \rightarrow B, \neg A \rightarrow B \models B$	Ley de Casos (LCs)
$A \leftrightarrow B \models A \rightarrow B$	Eliminación de equivalencia (EE)
$A \rightarrow B, B \rightarrow A \models A \leftrightarrow B$	Introducción a la equivalencia (IE)
$A, \neg A \models B$	Ley de inconsistencia (LT)

Ejemplo 1.2.4. Demostrar: $\forall x(P \rightarrow Q(x)) \equiv P \rightarrow \forall x Q(x)$

Solución

Demostración. Como P es una proposición entonces es verdadera o falsa, es decir debemos demostrar que :

■

$$\forall x(V \rightarrow Q(x)) \equiv V \rightarrow \forall x Q(x) \qquad \text{y}$$
$$\forall x(F \rightarrow Q(x)) \equiv F \rightarrow \forall x Q(x)$$

$$\boxed{\begin{array}{l} V \to A \equiv A \\ F \to A \equiv V \end{array}}$$

- $$\begin{aligned} \forall x(V \to Q(x)) &= \forall x Q(x) \\ &= V \to \forall x Q(x) \end{aligned}$$

- $$\begin{aligned} \forall x(F \to Q(x)) &= F \to \forall x(V) \\ &= V \\ &= F \to \forall x Q(x) \end{aligned}$$

Entonces la expresión es válida. □

Observación 1.2.3. Es claro que

a) Las tautologías se pueden transformar en esquemas en el sentido de que se puede hacer que cada variable lógica represente a una expresión.

b) Las expresiones válidas de la lógica de predicados se pueden transformar en esquemas salvo que hay que prestar especial atención a las variables libres y ligadas. En el ejemplo anterior P se puede sustituir por cualquier expresión, siempre y cuando esa expresión no contenga a x como variables libre y esta sustitución no afecte a su validez.

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

Válido

c) Los problemas indecibles no tienen una solución general, en el sentido de que no existe un método que pueda proporcionar de modo fiable una respuesta para el problema. De hecho el problema de demostrar la validez de una expresión es indecible.

Observación 1.2.4. Observemos que

a) Una expresión A es válida si no hay ninguna interpretación para la cual A produzca el valor falso. Para determinar que A no es válida, es entonces suficiente dar un único contraejemplo. Equivalentemente basta con hallar un único modelo para $\neg A$.

b) Para probar que una condicional no es válida es suficiente hallar un modelo que haga que el antecedente sea verdadero y el consecuente falso

$$\begin{array}{ll} A \to B & \text{No es válida} \\ V \quad F & \end{array}$$

Ejemplo 1.2.5. $\exists x P(x) \to \forall x P(x)$ no es válida

Demostración. Entonces debo encontrar una interpretación que haga a $\exists x P(x) \equiv V$ y $\forall x P(x) \equiv F$. Equivalentemente debo hallar un modelo para $\exists x P(x) \wedge \neg \forall x P(x)$.

Sean a, b el dominio y asignemos:

$$\begin{array}{rrl} P(a) \equiv V & P(b) \equiv F & \text{entonces:} \\ \exists x P(x) \equiv V & \forall x P(x) \equiv F & \\ \neg \forall P(x) \equiv V & & \end{array}$$

Entonces hemos encontrado un modelo □

c) Para demostrar que una expresión no es válida, también es posible utilizar la expresión y derivar de ella algo absurdo.

La expresión:

$$\forall x \exists y P(x, y) \to \exists y P(y, y) \qquad \text{no es válida}$$

$$\boxed{\begin{array}{c} P(t_1, t_2) = t_1 \text{ es madre de } t_2 \\ \text{absurdo} \end{array}}$$

d) El problema de demostrar si una expresión es o no válida es indecible.

Ejemplo 1.2.6. Demostrar que es válida la siguiente expresión:

$$\forall x P(x) \to \neg \forall x \neg P(x)$$

Demostración.

$$\begin{array}{rll} A \to B \text{ es válida} \Leftrightarrow \neg(A \to B) & & \text{no es válida} \\ \Leftrightarrow \neg(\neg A \vee B) & & \\ \Leftrightarrow A \wedge \neg B & & \text{no es válida} \end{array}$$

Debo hallar una interpretación que haga a:

$$\forall x P(x) \vee \forall x \neg P(x) \qquad \text{Falsa}$$

Esto es evidente, Si se cumple para todo , como se va a cumplir para su negación.

$$\begin{array}{rl} a, b & P(a) = V \\ & P(b) = V \\ & \forall x P(x) = V \\ & \forall x \neg P(x) = F \end{array}$$

□

1.3. Derivaciones

- La expresión $\forall x A \Rightarrow \int_t^x A$ es válida si la sustitución $x = t$ no da a lugar una colisión de variables. En otras palabras t no debe transformarse en una variable ligada de ningún cuantificador que pudiera quedar. Esta implicación se puede convertir en una regla de inferencia en la siguiente forma:

$$\frac{\forall x A}{\int_t^x A}$$

  Que llamaremos particularización universal (P.U)

- La particularización Universal nos permite concluir que

$$\frac{\forall x(\text{gato}(x) \rightarrow \text{tieneCola}(x))}{\text{gato(Tom)} \rightarrow \text{tieneCola}}$$

  Ejemplo 1.3.1. Tenemos las siguientes premisas:

  Todos los seres humanos son mortales
  Sócrates es un ser humano
  $\therefore$ Sócrates es mortal

  Solución

  Demostración.

$$H(t_1) = t_1 \text{ es humano}$$
$$M(t_1) = t_1 \text{ es mortal}$$

1. $\forall x(H(x) \rightarrow M(x))$	Premisa
2. $H(s)$	Supuesto
3. $H(s) \rightarrow M(s)$	P.U (1)
4. $M(s)$	MP (2,3)

□

- Si A es cualquier expresión y si x es una variable que no aparece libre en ninguna premisa, entonces:

$$\frac{A}{\forall x A}$$

  Esta regla de inferencia se denomina generalización Universal (G.U)

  La generalización universal está sometida a restricciones.

- Si se generaliza sobre x, entonces x no debe aparecer en ninguna premisa como variable libre. Si x aparece libre en alguna premisa, entonces x siempre se refiere al mismo individuo y está fijada en este sentido. Si x esta fijada, entonces no se puede generalizar sobre x. Las generalizaciones que parten de un sólo individuo para llegar a todo el dominio son incorrectas.

- Si x no aparece en ninguna premisa o si x esta ligada en todas las premisas, entonces se supone que x representa a cualquier individuo o se puede aplicar sin restricción la generalización universal

Ejemplo 1.3.2. Demostrar:

David es el padre de Pablo
Pablo no es la hija de David
Toda persona cuyo padre sea David debe ser o bien su hijo o bien su hija
$\therefore$ Pablo es hijo de David

Solución

Demostración.

$$\text{Padre } (t_1, t_2) = t_1 \text{ es padre de } t_2$$
$$\text{Hijo } (t_1, t_2) = t_1 \text{ es hijo de } t_2$$
$$\text{Hija } (t_1, t_2) = t_1 \text{ es hijo de } t_2$$

1. Padre (D, P)	Premisa
2. $\neg$Hija (P, D)	Premisa
3. $\forall x$Padre $(D, x) \rightarrow$ Hijo $(x, D) \vee$ Hija (x, D)	Premisa
4. Padre $(D, P) \rightarrow$ Hijo $(P, D) \vee$ Hija (P, D)	PU (3) $\int_P^x$
5. Hijo $(P, D) \vee$ Hija(P, D)	Modus Ponems (1,4)
6. Hijo (P, D)	Silogismo Disyuntivo (2,5)

□

Ejemplo 1.3.3. Demostrar

Todos los alumnos son de ciencias de la computación
A los alumnos de C.C les gusta la programación
$\therefore$ A todos les gusta la programación

Suponga que el dominio es el conjunto de alumnos de ciencias de la computación.

Solución

Demostración.

$$\text{CC } (t_1) = t_1 \text{ es alumno de CC}$$
$$\text{Programa } (t_2) = t_1 \text{ le gusta la programación}$$

1. $\forall x$ CC (x)	Premisa
2. $\forall x($ CC $(x) \rightarrow$ Prog $(x))$	Premisa
3. CC (x)	P.U (1) $\int_x^x$
4. CC $(x) \rightarrow$ Prog (x)	P.U (2) $\int_x^x$
5. Prog (x)	M.P (3,4)
6. $\forall x$Prog (x)	G.U.(5)

□

Ejemplo 1.3.4. Demostrar:

$$\forall x \forall y P(x,y) \models \forall y \forall x P(x,y)$$

Solución

Demostración.

1. $\forall x \forall y P(x,y)$	Premisa
2. $\forall y P(x,y)$	PU (1) $\int_x^x$
3. $P(x,y)$	PU (2) $\int_y^y$
4. $\forall x P(x,y)$	GU (3)
5. $\forall y \forall x P(x,y)$	GU (4)

□

1.4. Teorema de deducción

Se supone B, se demuestra C empleando B como premisa y se concluye $B \to C$. Una vez que se ha hecho esto, prescindimos de B.

- B se utiliza como suposición, esto es, mientras no prescindamos de B, hay que tratar a B como cualquier otra premisa.
- Si B contiene a x como variable libre, entonces no hay que generalizar sobre x. Sin embargo, en cuanto prescindamos de B, esto no será así. Una vez que hemos prescindido de B, no tiene efecto alguno sobre el estado de ninguna variable. Por tanto, si x no es libre en ninguna otra premisa, es posible generalizar sobre x aún cuando x aparezca como variable libre en B.

Ejemplo 1.4.1. Demostrar:

Todo el que ha estudiado ha aprobado
$\therefore$ Los que no hayan aprobado, no han estudiado.

Solución

Demostración. El Dominio es el conjunto de alumnos de estructuras Discretas

$E(t_1) = t_1$	ha estudiado
$A(t_2) = t_2$	ha aprobado

1. $\forall x(E(x) \to A(x))$	premisa
2. $E(x) \to A(x)$	PU (1)
3. $\neg A(x)$	Suposición
4. $\neg E(x)$	MT(2,3)
5. $\neg A(x) \to \neg E(x)$	T. de la deducción (3,4)
6. $\forall x(\neg A(x) \to \neg E(x))$	GU (5)

Para generalizar en la premisa la variable debe estar ligada.

□

Observación 1.4.1. Notemos que

a) Es frecuente omitir cuantificadores universales por ejemplo en la afirmación $x+y = y+x$, tanto x como y tienen implícitamente una cuantificación universal.

Esto da a lugar a problemas cuando se utilizan estas apreciaciones como premisas, porque de acuerdo con nuestra reglas toda variable que aparezca libre en una premisa es fija , en el sentido de que a lo largo de toda la demostración estará asociada a un único individuo. Para soslayar esta dificultad, aislamos ciertas variables en las premisas e indicamos explícitamente que estas variables no son fijas y se denominan variables verdaderas. Una variable se puede generalizar si y solo si es una variable verdadera.

b) Al utilizar variables verdaderas, se pueden omitir los cuantificadores universales. Además de ahora en adelante permitiremos que cualquier variable verdadera se puede particularizar como cualquier término.

c) En vez de usar $\int_y^x$ usaremos $x := y$

Ejemplo 1.4.2. Sea $P(t_1, t_2, t_3) = t_1 + t_2$ es igual a t_3 Demostrar que:

$$P(x, y, z) \to P(y, x, z)$$

$$\frac{P(x, 0, x)}{P(0, x, x)}$$

Solución

Demostración. Observar que x, y, z son variables verdaderas

1. $P(x, y, z) \to P(y, x, z)$	Premisa.
2. $P(x, 0, x)$	Premisa.
3. $P(x, 0, x) \to P(0, x, x)$	PU $y := 0, z := x$
4. $P(0, x, x)$	MP $(2, 3)$

□

⋆En el ejemplo la x de la línea 1 y la x de la línea 2 son variables diferentes. Cuando se esta haciendo la demostración, uno tiene que establecer, algún tipo de conexión entre ambas variables, esta se efectúa a través de la particularización.

Definición 1.4. Se dice que dos expresiones se unifican si existen particularizaciones legales que hagan idénticos a las expresiones en cuestión. El acto de unificarlas se denomina unificación. La particularización que unifica a las expresiones en cuestión se denomina un unificador.

Ejemplo 1.4.3. $Q(a, y, z)$ y $Q(y, b, c)$ son expresiones que aparecen en líneas diferentes. Mostrar que estas dos expresiones se unifican y dar un unificador.

Observar que en realidad y representa diferentes variables.

$$\begin{aligned} Q(a, y, z), \qquad & \{y := b, z := c, y_1 := a\} \text{ Unificador} \\ Q(y_1, b, c), & \\ \therefore Q(a, y, z) \text{ y } Q(y, b, c) \qquad & \text{se unifica en } Q(a, b, c) \end{aligned}$$

Ejemplo 1.4.4. Demostrar:

Si x es la madre de y y si z es la hermana de x entonces z es la tía de y
La madre de Braulio es Juana
Lola es hermana de Juana
$\therefore$ Lola es tía de Braulio

Demostración.

$$\begin{aligned} M(t_1, t_2) &\equiv t_1 \text{ es la madre de } t_2. \\ H(t_1, t_2) &\equiv t_1 \text{ es la hermana de } t_2. \\ T(t_1, t_2) &\equiv t_1 \text{ es la tía de } t_2. \end{aligned}$$

1. M $(x, y) \wedge$ H $(z, x) \rightarrow$ T (z, y)	Premisa
2. M (J, B)	Premisa
3. H (L, J)	Premisa
4. M $(J, B) \wedge H(L, J)$	LC $(2, 3)$
5. M $(J, B) \wedge$ H $(L, J) \rightarrow$ T (L, B)	Unificación $(1, 4)$ $x := J\, y := B\, z := L$
6. T (L, B)	MP $(4, 5)$

□

Definición 1.5. La regla de inferencia

$$\frac{\int_t^x A}{\exists x A}$$

se denomina Generalización Existencial (G.E)

Ejemplo 1.4.5. Demostrar:

$$\frac{\begin{array}{c}\text{Toda persona que ha ganado cien millones es rica}\\ \text{María ha ganado cien millones}\end{array}}{\therefore \text{ Hay alguien que es rico}}$$

Demostración.

G $(t1) \equiv t_1$ ha ganado cien millones
R $(t1) \equiv t_1$ es rico.

1. $\forall x(G(x) \to R(x))$	Premisa
2. $G(M)$	Premisa
3. $G(M) \to R(M)$	PU(1), $x := M$
4. $R(M)$	MP (2,3)
5. $\exists x R(x)$	GE (4)

□

Observación 1.4.2. Si es cierto que existe un x que verifica A entonces tiene que haber algún termino t que satisfaga A. El problema es que no sabemos para que termino. Para hacer esto, seleccionamos una nueva variable, digamos b para denotar este individuo desconocido. Esto da a lugar a la regla de inferencia

$$\frac{\exists x A}{\int_b^x A}$$

llamada particularización existencial (P.E). La variable que se introduce está fijada en el sentido de que no se puede aplicar una generalización universal a esa variable. Además, no debe aparecer en la conclusión una variable que tenga valor desconocido y dado que toma variable introducida por la particularización existencial es desconocida, tampoco deberá aparecer en la conclusión.

Ejemplo 1.4.6. Demostrar:

$$\frac{\begin{array}{c}\text{Alguien ha ganado cien millones}\\ \text{Cualquiera que haya ganado cien millones es rico}\end{array}}{\therefore \text{ Alguien es rico}}$$

Demostración.

1. $\exists x(G(x)$	Premisa
2. $\forall x(G(x) \to R(x))$	Premisa
3. $G(b)$	PE (1)$x := b$
4. $G(b) \to R(b)$	PU (2)$x := b$
5. $R(b)$	MP (3,4)
6. $\exists x R(b)$	G.E. (5)

□

Ejemplo 1.4.7. Demostrar;

$$\forall x \neg P(x) \models \neg \exists x P(x) \qquad \text{Por el absurdo}$$

Demostración.

1.	$\exists x P(x)$	Suposición
2.	$\forall x \neg P(x)$	Premisa
3.	$P(b)$	P.E (1) $x := b$
4.	$\neg P(b)$	P.U (2) $x := b$
5.	$P(b) \vee \neg P(b)$	L.C (3,4) ¡Contradicción!, Se rechaza la suposición
6	$\neg \exists x P(x)$	

□

Observación 1.4.3. Veámos lo siguiente

a) Si A es una expresión, es posible sustituir cualquier sub-expresión B de A por una Sub – expresión C; siempre y cuando B y C sean equivalentes.

b) Equivalencias que implican cuantificadores

1.	$\forall x A \equiv A$	si x no es libre en A
1*d*.	$\exists x A \equiv A$	si x no es libre en A
2.	$\forall x A \equiv \forall y \int_y^x A$	si y no es libre en A
2*d*.	$\exists x A \equiv \exists y \int_y^x A$	si y no es libre en A
3.	$\forall x A \equiv \int_t^x A \wedge \forall x A$	para cualquier termino t
3*d*	$\exists x A \equiv \int_t^x A \vee \exists x A$	para cualquier termino t
4.	$\forall x (A \vee B) \equiv A \vee \forall x B$	si x no es libre en A
4*d*.	$\exists x (A \wedge B) \equiv A \wedge \exists x B$	si x no es libre en A
5.	$\forall x (A \wedge B) \equiv \forall x A \wedge \forall x B$	
5*d*.	$\exists x (A \vee B) \equiv \exists x A \vee \exists x B$	
6.	$\forall x \forall y A \equiv \forall y \forall x A$	
6*d*.	$\exists x \exists y A \equiv \exists y \exists x A$	
7.	$\neg \exists x A \equiv \forall x \neg A$	
7*d*.	$\neg \forall x A \equiv \exists x \neg A$	

c) Se llama distinguir las variables por estandarización a cambiar los nombres de una expresión de tal manera que las variables distintas tengan distintos nombres

Ejemplo 1.4.8. Distinguir por estandarización todas las variables de la expresión siguiente:

$$\forall x (P(x) \to Q(x)) \wedge \exists x Q(x) \wedge \exists z P(z) \wedge \exists z (Q(z)) \to R(x))$$
$$\forall y (P(y) \to Q(y)) \wedge \exists u Q(u) \wedge \exists v P(v) \wedge \exists z (Q(z)) \to R(x))$$

Ejemplo 1.4.9.

$$\begin{aligned}
&\neg\forall z(\exists x P(x,z) \wedge \neg\forall x Q(x,z)) \\
&\exists z\neg(\exists x P(x,z) \wedge \neg\forall x Q(x,z)) \\
&\exists z(\neg\exists x P(x,z) \vee \forall x Q(x,z)) && \text{Ley de Morgan} \\
&\exists z(\forall\neg x P(x,z) \vee \forall x Q(x,z))
\end{aligned}$$

Observación 1.4.4. Observemos claramente que:

a) Resulta ventajoso eliminar los cuantificadores universales. Sin embargo los cuantificadores universales no se pueden eliminar a no ser que se encuentren el principio de la expresión.

b) Las equivalencias siguientes resultan de especial interés.

$$\begin{aligned}
\forall x(P(x) \vee \forall y Q(y) &\equiv \forall x\forall y(P(x) \vee Q(y)) \\
\forall x(B \to A) &\equiv \exists x B \to A && x \text{ no esta libre en } A
\end{aligned}$$

$\forall x(B \to A)$	$\equiv$	$\forall x(\neg B \vee A)$	equivalencia de $\to$
	$\equiv$	$\forall x\neg B \vee A$	(4)
	$\equiv$	$\neg\exists x B \vee A$	(7)
	$\equiv$	$\exists x B \to A$	equivalencia de $\to$

Considere la frase:

"si alguien habla, mañana estará en los periódicos"

$$\begin{aligned}
\text{Habla } (t_1) &\equiv t_1 \text{ habla} \\
P &\equiv \text{mañana estará en los periódicos} \\
\exists x \text{ habla } (x) \to P &\equiv \forall x(\text{habla } (x) \to P)
\end{aligned}$$

1.5. Lógica de ecuaciones

Observación 1.5.1. No olvidemos que

a) La igualdad también es importante en lógica, en particular para indicar que solamente existe un elemento que satisfaga una cierta propiedad.

b) Dos términos t y r se dicen iguales si hacen alusión al mismo individuo y para expresar esto, se escribe $t = r$.

Por ejemplo: Superman = Clark Kent

c) Una forma de decidir la igualdad es mediante una asignación explícita. En lugar de una asignación explícita, se da un cierto número de axiomas que deben satisfacer el predicado igual.

Ejm. Igual $(t, r) \equiv t = r$

d) $\forall x(x = x)$ se denomina axioma de la reflexividad.

e) Si A es cualquier expresión, entonces $R(n)^t_r\ A$ es la expresión que se obtiene a partir de A sustituyendo la n-ésima aparición del término t por r. Si t aparece en menos de n ocasiones en A. Entonces, $R(n)^t_r A \equiv A$

Ejemplo 1.5.1.

$$R(1)^x_y(x = x) \equiv y = x$$
$$R(2)^x_y(x = x) \equiv x = y$$
$$R(3)^x_y(x = x) \equiv x = y$$

1.5.1. Regla de sustitución

Si A y $t = r$ son dos expresiones que se han derivado, se permite concluir que $R(n)^t_r A$ para todo $n > O$. En este caso diremos que sustituimos t a partir de $t = r$ en A.

Ejemplo 1.5.2. Sustituir $x + 1$ de $x + 1 = 2y$ en $x < x + 1$

Solución

$$R(1)^{x+1}_{2y}(x < x + 1) \equiv x < 2y$$

1. $\forall x(x = x)$	Axioma reflexividad
2. $x = y$	Supuesto
3. $x = x$	PU (1) $x := x$
4. $y = x$	$R(1)^x_y\ (x = x)$
5. $x = y \to y = x$	Teorema de la Deducción (2,4)

⋆La expresión resultante se puede cuantificar en forma universal, lo cual produce:

$$\forall x \forall y(x = y \to y = x)$$

Por lo tanto, dado $t = u$, no solamente es posible sustituir t por u, sino que además podemos sustituir u por t.

Ejemplo 1.5.3. Demostrar que: $\vdash x = y \wedge y = z \to x = z$
Solución

$$x = y \wedge y = z \vdash x = z \qquad \text{Teorema de la Deducción}$$

1. $x = y \wedge y = z$	Premisa
2. $x = y$	L.S (1)
3. $y = z$	L.S (1)
4. $x = z$	$R(1)^y_x(y = z)$ del ejemplo anterior es conmutativa
5. $x = y \wedge y = z \to x = z$	con un supuesto previo

$A \vdash B \to C$	T. Deducción
$A, B \vdash C$	
$A, B \vdash C$	$A \vdash B \to C$

Ejemplo 1.5.4. Después de todas las interacciones de un cierto bucle, se verifica la conclusión $s > 3i$ Además una vez que termina el bucle i es 10. Demostrar que $s > 30$ cuando termina el bucle.

Solución

1. $s > 3i$	Premisa
2. $i = 10$	Premisa
3. $s > 3x10$	$R(1)^i_{10}$ $(s > 3i)$

⋆La regla de sustitución se puede aplicar a subexpresiones que estén incluidas en otras expresiones, en este sentido la regla de sustitución es muy eficiente.

Observación 1.5.2. Se hace hincapié

α) En general, el signo igual no se puede utilizar, si el lado izquierdo de la expresión puede hacer alusión a distintos objetos. Si $x_1 = y$ y $x_2 = y$ entonces, uno siempre puede concluir que $x_1 = x_2$. De allí que la igualdad hace necesaria a la unicidad.

β) A la inversa, para expresar la unicidad, se utiliza la igualdad. La afirmación de "el termino t es el único elemento existente con la propiedad P"

Se puede parafrasear diciendo:

"Si x no es t, entonces $P(x)$ no puede ser cierto", esto se simboliza en la forma:

$$\forall x\{\neg(x = t) \to \neg P(x)) \equiv \forall x(P(x) \to x = t)\}$$

γ) El hecho de que ahora existe un individuo, pero uno solo que tiene la propiedad p se puede expresar de la forma:

$$\exists x(P(x) \wedge \forall y(P(y) \to y = x) \equiv \exists! x P(x)$$
$$\exists x P(x) \wedge \forall x \forall y(P(x) \wedge P(y) \to x = y) \equiv \exists! x P(x)$$

Ejemplo 1.5.5. Para las siguientes expresiones.

a) "Sólo Juan podría haber olvidado la reunión"

b) "la universidad tiene exactamente un rector"

c) "Todas los países tienen exactamente una capital"

solución

a) Olvidado $(t_1) \equiv t_1$ podría haber olvidado la reunión

$$\forall x \text{ (olvidado}(x) \to x = J)$$

b) Rector $(t_1) \equiv t_1$ es el rector de la universidad.

$$\exists! x \text{ Rector}(x) \equiv \exists x \text{ Rector}(x) \wedge \forall x \forall y \text{ (Rector}(x) \wedge \text{ Rector}(y) \to x = y)$$

c) $\forall x(\text{País}(x) \to \exists y\ (\text{Ciudad}(y) \wedge \text{capital}(y))\ )$

Observación 1.5.3. Notemos que

a) Las letras de función representan las formas abstractas representativas de cualquier aplicación de $D^n \to D$, siendo D el Dominio de referencia.

b) El concepto de término se define de la forma recursiva siguiente:

b_1 Son términos las letras de variables y constantes.

b_2 Si y es una letra e función, son términos las expresiones del tipo $\psi(t_1, t_2, \ldots, t_n)$ donde $(t_1, t_2, \ldots, t_n)$ son términos.

c) El concepto de función permite simplificar las estructuras de las fórmulas.

Ejemplo 1.5.6. Simbolizar en lógica de predicados las frases:

a) "Algunos siguientes a los alumnos de la clase son aficionados al cine".

b) "Ningún producto de dos números naturales es primo".

Solución

a. $D =$Alumnos

$$Cine(t_1) = t_1 \text{ es aficionado al cine}$$
$$Siguiente(t_1, t_2) = t_1 \text{ es siguiente a } t_2$$

$$\exists x \exists y Siguiente(x, y) \wedge Cine(x)$$

b. $D = \mathbb{N}$

$$Producto(t_1, t_2, t_3) = t_1 * t_2 \text{ es igual a } t_3$$
$$Primo(t_1) = t_1 \text{ es primo}$$

$$\forall x \forall y \forall z(Producto(x, y, z) \to \neg Primo(z))$$

Sin embargo si tenemos en cuenta el concepto de función:

a) $\psi(x) = siguiente(x)$
$\exists x\ cine(\psi(x))$

b) $\psi(x, y) = x * y$
$\forall x \forall y(\neg Primo(\psi(x, y)))$

Observación 1.5.4. Tenga presente

a) Si los cuantificadores se aplican exclusivamente a las variables, el sistema lógico planteado se denomina: Lógica de predicados de 1^{er} orden; sin embargo se presentan en el lenguaje usual afirmaciones del tipo: Existe al menos una función ψ tal que $\psi(a) = \psi(b)$. La formulación de esta frase exige un dominio para las letras a y b, y un dominio par la letra de función ψ. La estructura de la frase sería:

$$\exists \psi\ \text{Igual}(\psi(a), b)$$

b) Análogamente pueden plasmarse frases que requieren una cuantificación de letras de predicados. Por ejemplo: Algunas relaciones entre pares de alumnos son simétricas.

$$\exists R \forall x \forall y \, (R(x,y) \rightarrow R(y,x)$$

Donde R es el predicado que significa que x esta en relación con y.

c) La lógica de predicados planteada en a y b se denomina lógicos de 2^{do} orden.

1.6. Forma normal PRENEX

Definición 1.6. La forma PRENEX constituye una estructura de formulas en cálculo de predicados caracterizada por las siguientes propiedades:

a) Todas los cuantificadores aparecen en cabeza de una expresión a la que afectan en su totalidad.

b) La expresión afectada por el conjunto de cuantificadores se denomina matriz de la forma y esta constituida por predicados y conectivas, de esta últimas solo aparecen conjunción, disyunción y negación, esta última aplicada solo a predicados atómicos.

Ejemplo 1.6.1.

$$\forall x \forall y \forall z \, (\neg A(x,y,z) \wedge (R(x,w) \vee P(x,w))$$

Teorema 1.1. *Dada una formula (expresión cualquiera A) en lógica de predicado existe otra formula $FP(A)$ tal que $A \equiv FP(A)$.*

Prueba Es suficiente definir un procedimiento que se aplique sucesivamente incorporando características de la forma normal Prenex.

¤ Eliminación de la condicional ($\rightarrow$)

$$A \rightarrow B = \neg A \vee B$$

¤ Eliminación del signo $\neg$ a fórmulas compuestas

$$\begin{aligned} \neg(A \wedge B) &\equiv \neg A \vee \neg B \\ \neg(A \vee B) &\equiv \neg A \wedge \neg B \\ \neg \forall x A(x) &\equiv \exists x \neg A(x) \\ \neg \exists x A(x) &\equiv \forall x \neg A(x) \end{aligned}$$

¤ Situación de los cuantificadores con cabeza de la formula

$*_1$ La fórmula parcial conectada no contiene libre la variable cuantificada:

$$A \vee \forall x P(x) \equiv \forall x(A \vee P(x))$$
$$A \wedge \forall x P(x) \equiv \forall x(A \wedge P(x))$$
$$A \vee \exists x P(x) \equiv \exists x(A \vee P(x))$$
$$A \wedge \exists x P(x) \equiv \exists x(A \wedge P(x))$$

x no esta libre en A.

$*_2$ La formula parcial conectada contiene libre la variable cuantificada. En este caso, las expresiones serían del tipo:

$$A(x) \vee \forall x P(x) \equiv \forall x(A \vee P(x))$$
$$A(x) \wedge \forall x P(x) \equiv \forall x(A \wedge P(x))$$
$$A(x) \vee \exists x P(x) \equiv \exists x(A \vee P(x))$$
$$A(x) \wedge \exists x P(x) \equiv \exists x(A \wedge P(x))$$

Antes de aplicar $*_1$ hacer un cambio de variable.

Ejemplo 1.6.2.

$$A(y) \vee \forall x P(x) \equiv \forall x(A(y) \vee P(x))$$

Esto puede aumentar el número de cuantificadores en una fórmula que se puede evitar en los casos:

$$\forall x(A(x) \wedge B(x)) \equiv \forall x A(x) \wedge \forall x B(x)$$
$$\exists x(A(x) \vee B(x)) \equiv \exists x A(x) \vee \exists x B(x)$$

Ejemplo 1.6.3. Encontrar la forma normal Prenex equivalente a la expresión:

$$\neg\exists x(P(x) \to \forall y Q(y))$$

- Eliminar las condicionales

$$\neg\exists x(\neg P(x) \vee \forall y Q(y))$$

- Eliminación de la negación

$$\forall x\neg(\neg P(x) \vee \forall y Q(y))$$
$$\forall x(P(x) \wedge \neg\forall y Q(y))$$
$$\forall x(P(x) \vee \exists y\neg Q(y))$$

- Cuantificadores en cabeza de formula

$$\forall x\exists y(P(x) \wedge \neg Q(y))$$

1.7. Lista de ejercicios

1. Formular las siguientes sentencias en lógica de predicados:

   a) El profesor dio una tarea el lunes, y la siguiente el miércoles. Todos los alumnos se quejaron y algunos no pudieron finalizar su trabajo. (Seres humanos)

   b) Se puede engañar a algunas personas algunos días, pero no se puede engañar a todas las personas todos los días. (Seres humanos, Días).

2. Sea un universo de discurso que consta de tres personas solamente, a saber, Juan, María y Juana. Los tres son alumnos, y ninguno de ellos es rico. Juan es varón, mientras que María y Juana son hembras. S, F, M y R denotan respectivamente las propiedades alumno, hembra, varón y rico:

   a) Presentar la asignación de los predicados S,F,M y R.

   b) Calcular: $\forall x S(x)$, $\forall x F(x) \vee \forall x M(x)$, $\exists x(F(x) \rightarrow R(x))$

3. Supóngase que $VC(x, y)$ representa el hecho consistente en que x e y viven en la misma ciudad. Claramente, $\forall x \forall y \forall z(VC\ (x, y) \wedge VC\ (y, z) \rightarrow VC\ (x, z))$. Empleando esto como una premisa, dar una demostración formal de que si Pedro vive en la misma ciudad de María, Y María vive en la misma ciudad que Beni, entonces Pedro vive en la misma ciudad que Beni.

4. Dar una derivación formal para $(P(a) \wedge \forall x P(x)) \leftrightarrow \forall x P(x)$

5. Simbolizar en lógica de predicados y demostrar la siguiente deducción:

   - Eduardo pudo haber visto al asesino.
   - Antonio fue el primer testigo de la defensa.
   - Eduardo estaba en clase o Antonio dio testimonio falso.
   - Nadie en clase pudo haber visto al asesino.

   Luego : El primer testigo de la defensa dio testimonio falso.

6. Hallar la forma normal Prenex, de la siguiente fórmula:

$$A(x, y) \wedge (\exists x B(x) \rightarrow \forall y \forall z C(x, y, z))$$

7. Demostrar:
   $(P(x) \rightarrow Q(y)) \wedge (Q(y) \rightarrow R(z)) \rightarrow (P(z) \rightarrow Q(z))$ no es válida.

8. En el dominio de los animales, ¿Cómo traduciría las expresiones siguientes?

   a) Todos los leones son depredadores.

   b) Algunos leones viven en África.

   c) Sólo rugen los leones.

   d) Algunos leones comen cebras.

   e) Algunos leones sólo comen cebras.

9. Demuestre lo siguiente por derivación:

$$\forall x P(x) \vdash \forall x (P(x) \vee Q(x))$$

10. Poner en la forma normal Prenex:

$$A \equiv \forall x [P(x) \to \neg \forall y (Q(x,y) \to \exists z P(z)) \wedge \forall y (Q(x,y) \to R(t))]$$

Capítulo 2

Razonamiento: Lógico - Matemático

2.1. Introducción

Comenzaremos con el concepto más fundamental en computación: el de algoritmo.

El estudio de los algoritmos comenzó como una tarea de matemáticas; de hecho, la búsqueda de algoritmos era una actividad importante de las matemáticas mucho antes de que se inventaran las computadoras actuales, y el objetivo principal de dicha búsqueda era decubrir un conjunto único de dichas instrucciones para resolver cualquier problema de cierto tipo.

El nombre algoritmo proviene del matemático persa del siglo IX Alkhowãrizmi. El algoritmo más famoso de la historia procede de un tiempo anterior al de los griegos; se trata del algoritmo de Euclides para calcular el máximo común divisor de dos enteros.

Una vez descubierto un algoritmo para resolver un problema, el siguiente paso es representar el algoritmo de modo que se pueda comunicar a una máquina o a otras personas. Esto significa que debemos transformar el algoritmo conceptual en un conjunto claro de instrucciones y representar ésta última sin ambigúedad. Los estudios que se ocupan de esto se basan en nuestro conocimiento del lenguaje y la gramática, y ha dado lugar a un gran número de esquemas para representar algoritmos (llamados lenguajes de programación) fundados en diversos enfoques del proceso de programación (denominados paradigmas de programación).

La figura 2.1 representa la evolución histórica de los paradigmas de programación:

El Paradigma por Procedimiento También conocido como paradigma imperativo, representa el enfoque tradicional del proceso de programación. El paradigma por procedimiento define al proceso de programación como el desarrollo de procedimientos que, al seguirse, manipulan los datos para producir el resultado deseado. Así, el paradigma por procedimientos nos dice que abordemos un problema tratando de hallar un método para resolverlo.

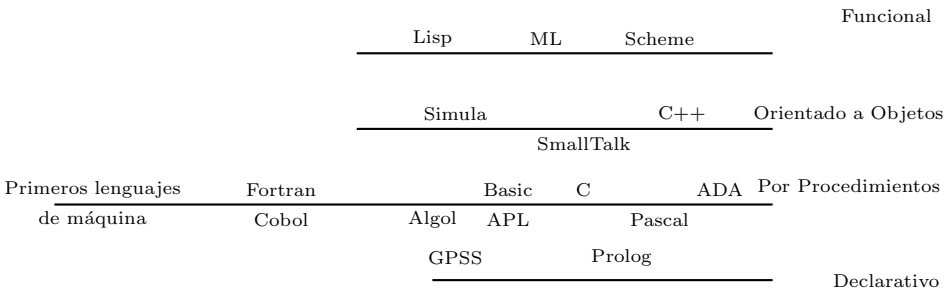


Figura 2.1: Evolución de los Paradigmas de Programación

El Paradigma Declarativo Hace hincapié en la pregunta ¿Cuál es el problema ?, en vez de ¿Qué procedimientos necesitamos para resolver el problema?

El Paradigma Funcional Contempla el proceso de creación de programas como la construcción de "cajas negras", cada una de las cuales acepta entradas y produce salidas. Los matemáticos llaman funciones a tales "cajas ", y es por ello que este enfoque se denomina paradigma funcional.

El Paradigma Orientado a Objetos Conduce al proceso denominado programación a objetos. Con este enfoque, las unidades de datos se consideran "objetos"activos, en vez de las unidades pasivas contempladas por el paradigma por procedimientos. Para aclarar esto, consideremos como lista de nombres. En el paradigma por procedimientos, esta lista no es más que una colección de datos, y cualquier programa que obtenga acceso a esta lista deberá incluir los algoritmos para realizar las manipulaciones requeridas.

Así, la lista es pasiva en el sentido de que la mantienen un programa controlador, en vez de tener responsabilidad de mantenerse ella misma. En cambio, en el enfoque orientado a objetos la lista se considera como un objeto formado por la lista misma y que por una serie de rutinas para manipularla. Por ejemplo, pueden ser rutinas para insertar una nueva entrada en la lista, para detectar si la lista está vacía o para ordenar la lista.

De este modo, un programa que obtiene una acceso a esta lista no necesita contener algoritmos para efectuar esas tareas; simplemente aprovecha las rutinas suministradas en el objeto. En cierto sentido, en vez de ordenar la lista como el paradigma por procedimientos, el programa pide a la lista que se ordene a sí misma.

Muchas ventajas del paradigma orientado a objetos son consecuencias de la estructura modular, que surge como subproducto natural de la filosofía orientada, bien definido, cuyas características son en gran medida independientes del resto del sistema. Así, una vez desarrollado un objeto que representa a una determinada entidad, es posible reutilizar ese objeto cada vez que se requiera

dicha cantidad. De hecho, un rasgo primordial de los lenguajes orientados a objetos es la capacidad de representar definiciones de objeto a modo de esqueletos que pueden usarse una y otra vez para construir nuevos objetos con propósitos similares.

Es evidente que la creación de Lenguaje de Programación es un proceso contínuo, y nuevos lenguajes evolucionan al tiempo que buscamos formar más conveniente de comunicarnos con las máquinas. Por tanto, es inevitable que los lenguajes de programación que hoy representan la frontera del conocimiento serán criticados en el mañana por ser arcaicos.

2.2. Algoritmos

Definición 2.1. Un algoritmo es una secuencia finita de pasos ejecutables que, de seguirla, debe terminar en un momento.

- ♠ El empleo del término *no ambiguo*, significa que, en cada paso, la acción que debe realizarse a continuación debe quedar determinada de manera única por la construcción por sí sola debe bastar para determinar la acción deseada.
- ♠ El requisito de que todos los pasos del algoritmo sean ejecutables descarta la posibilidad de que un algoritmo contenga pasos cuya ejecución sea imposible. Los científicos de la computación emplean el término efectivo para capturar este concepto de ser ejecutable.
- ♠ El requisito de que una secuencia de pasos terminará no es superflúo pues que la lista de instrucciones contenga sólo un número finito de pasos no significa necesariamente que la tarea concluirá alguna vez si se siguen las instrucciones.
- ♠ El concepto de calcular la solución de un problema implica la capacidad de reconocer cuándo ya se obtuvo la solución, para entonces terminar el proceso de cómputo. Así pues, en este sentido, sólo nos interesan los procesos de cómputo que tarde o temprano terminan.

Observación 2.2.1. Recuerde que

a) Un algoritmo en si es algo puramente conceptual, de modo que para comunicar un algoritmo a una persona o a un computador debemos de hallar una forma de representarlo.

b) La distinción entre sintaxis y semántica es primordial en el tema de la representación de algoritmos. El término *sintaxis* se refiere a la representación, en tanto que *semántica* se refiere al concepto que se representa. La estructura sintáctica "aire"sólo un conjunto de letras; pero el objeto representado, la semántica, es una sustancia gaseosa que rodea el planeta.

   Un objetivo importante de la representación de algoritmos es asegurar que la sintaxis refleje con precisión la semántica deseada. Otro aspecto importante es que la sintaxis debe reflejar la semántica subyacente de manera accesible. La búsqueda de estructuras sintácticas para representar algoritmos de manera accesible y no ambigua es una labor continua en las ciencias de la computación.

c) Las ciencias de la computación atacan estos dos requisitos mediante el uso de promitivas. Una primitiva consiste en una estructura semántica bien definida junto con una sintaxis no ambigua para representarla. Con este enfoque se resuelve el problema de representación de algoritmos; estableciendo primero una colección suficientemente rica de primitivas para que cualquier algoritmo se pueda expresar mediante combinaciones de las mismas; y expresando después todos los algoritmos en éstos términos. Tal colección de primitivas, junto con las reglas de combinación para representar estructuras más complejas, constituye un lenguaje de programación.

d) Nos abstendremos de definir formalmente lo que es un lenguaje de programación, en vez de ello presentaremos un sistema de notación menos formal, **un pseudocódigo es un sistema de notación en el que podemos expresar informalmente ideas durante el proceso de creación de algoritmos**, por tanto el pseudocódigo presenta estructuras sintaácticosemántica similares, pero menos formales, a las empleadas en el lenguaje de programación objetivo.

e) La necesidad de seleccionar una de dos posibles actividades dependiendo de que se cumpla o no una condición es una estructura algorítmica común, que es pseudocódigo, y se expresa de la siguiente manera:

```
1 if (condicion)
2       actividad - verdadero
3 else
4       actividad -falso
```

f) Los bloques verdadero y falso (que deben ser realizados si al condición es verdadera o falsa, respectivamente) pueden contener cualquier pseudocódigo válido, incluyendo selecciones e interacciones. En ocasiones se omitirá el enunciado 2.

g) Una forma fundamental para expresar que hay que seguir ejecutando un enunciado o secuencia de enunciados, en tanto se cumpla cierta condición es la forma:

```
1 while (condicion)
2       repetir-actividad
```

En este paso, se revisa la CONDICIÓN; de ser cierta, se realiza el bloque de pseudocódigo que la sigue. Este proceso se repite mientras la condición sea verdadera. Si en algún momento se detecta que la CONDICIÓN es falsa, se pasará a la siguiente instrucción.

h) Una modificación sencilla de la primitiva WHILE, es la primitiva UNTIL (HASTA) y son completamente equivalentes.

i) Recorrer una sucesión utilizando una variable i, la cual toma valores enteros de 1 hasta “n”, la podemos expresar con la estructura:

FOR VAR = inicial **TO** final **DO** repetir - actividad

j) Hasta aquí, hemos estado analizando el papel de las interrelaciones sintáctico-semánticas en referencia al objetivo de producir programas no ambigúos. Sin embargo, la eliminación de la ambigúedad no es lo único que nos interesa al representar algoritmos; igual de importante es que le programa final sea fácil de entender y modificar. Esto es, de hecho la razón por la que usamos sangrías en nuestro pseudocódigo.

Una técnica para obtener una representación bien organizada consiste en construírla en forma modular; esto es, dividir la tarea que realiza el algoritmo en unidades pequeñas (módulos) y representar cada módulo por separado.

Luego, estos módulos podrán usarse como herramientas abstractas para construir otros módulos que se encarguen de porciones mayores de la tarea global. El resultado es una representación organizada en niveles variables de detalle. En el nivel más alto, encontramos los pasos del algoritmo expresados en términos de unidades grandes, cuyas funciones están muy relacionadas con la tarea global por realizar.

Al dirigir nuestra atención a niveles que están más abajo en la jerarquía, observamos que cada tarea individual está definida con mayor detalle y en terminología más parecida a las primitivas con las cuales todas las tareas se definirń en última instancia, De éste modo, todo el que examine semejante representación podrá entender su composición con relativa facilidad y evaluarla (total o parcialmente) en el nivel de detalle deseado.

k) Con la anterior observaci ón en mente, entenderemos nuestro pseudocódigo, a fin de poder representar algoritmos en forma modular. Lo primero que haremos será establecer mecanismos para asignar nombres a los módulos comenzando cada módulo con un enunciado de la forma:

procedure NOMBRE (a, b,. . . , W) donde NOMBRE es el nombre que daremos al módulo compuesto por las instrucciones que siguen a este enunciado y A, B, ..., W son los parámetros del procedimiento, los cuales describen los datos, variables, arreglos, etc., que se encuentran disponibles para el procedimiento. La última línea de un procedimiento tiene la palabra “END ”seguida por el nombre del procedimiento.

Además de usar las sangrías para identificar las progresiones que conforman una determinada estructura podemos usar las palabras BEGIN y END. Las dos diagonales “// ”indican el incio de un comentario, el cual se extiende hasta el final de la línea. Los comentarios ayudan al lector a entender el algoritmo, pero no se ejecutan. La proposición “RETURN(X)”concluye con un procedimiento y regresa el valor “x”a quien llamó el procedimiento.

Si no existe una proposición “return”el proceso termina justo antes de la línea de “END”. Un procedimiento que contiene una proposición RETURN (x) es una función. El dominio consta de todos los valores válidos para los parámetros y el rango es el conjunto de valores que puede regresar al procedimiento.

l) Al utilizar un pseudocódigo, utilizamos las expresiones aritméticas comunes, +, -, *, /; así como los operadores de relación $=$, $\notin$, $<$, $>$, $\leq$, $\geq$ y los

operadores lógicos “and, or y not”. Utilizaremos el operador “:=”para la asignación.

m) Si se quiere dar mensajes, utilizar: “PRINT (’mensaje’) ”.

Ejemplo 2.2.1. El siguiente algoritmo permite determinar el máximo de los números “n_1, n_2, n_3 ”.
Entrada: n_1, n_2, n_3
Salida: max

PROCEDURE-MAXIMO(n_1, n_2, n_3)

```
1  max := n1
2  if n2 > max
3        max := n2
4  if n3 > max
5        max := n3
6  return (max)
```

Ejemplo 2.2.2. El siguiente algoritmo calcula el área de un triángulo de lados “a, b, c ”Entrada: a, b, c
Salida: $Area$

PROCEDURE AREA-TRIANGULO(a, b, c)

```
1  p := (a + b + c)/2 // Semiperímetro
2  if (p > a) and (p > b) and (p > c)
3        Area := (a × (p − a) × (p − b) × (p − c))^(1/2)
4  else
5        print ("Datos no forman un triángulo")
6  return (Area)
7  end Area-Triangulo
```

Ejemplo 2.2.3. El siguiente algoritmo calcula el cuadrado de un número entero positivo N por adiciones sucesivas.
Entrada: n
Salida: x

PROCEDURE CUADRADO(n)

```
1  x := n
2  i := 1
3  while (i ≠ n)
4        x := x + n
5        i := i + 1
6  return (x)
7  end Cuadrado
```

Ejemplo 2.2.4. El siguiente algoritmo hace exactamente lo mismo que “cuadrado”sólo que utiliza UNTIL.

PROCEDURE CUADRADO2(N)

```
1  x := n
2  i := 1
3  until (i < n)
4        x := x + n
5        i := i + 1
6  return (x)
7  end Cuadrado2
```

Ejemplo 2.2.5. El siguiente algoritmo determina la suma de los primeros "N"términos de la serie: $\sum_{n=1}^{\infty} 1/(8n+1)$
Entrada: n
Salida: Sum

PROCEDURE SUMA(n)

```
1  Sum := 0
2  for i := 1 to n
3        Sum := Sum + 1/(8 * i + 1)
4  return (Sum)
5  end Suma
```

Ejemplo 2.2.6. El siguiente algoritmo muestra el método de ordenación "burbuja".
Entrada: A
Salida: El arreglo A con elementos ya ordenados.

PROCEDURE ORDENACION-BURBUJA(A)

```
1  for i := 1 to length(A) − 1
2        for j := 1 to length(A) − i
3              if A[j] > A[j + 1]
4                    intercambiar A[j] con A[j + 1]
5  end Ordenacion-Burbuja
```

Ejemplo 2.2.7. El algoritmo siguiente describe el método de *búsqueda lineal*:

Entrada: x (Dato que desea encontrar en el arreglo), A (Lista con elementos).
Salida: Posición de x en la lista.

PROCEDURE BUSQUEDA-LINEAL (A, x)

```
1  i := 1
2  while i ≤ n and x ≠ A[i]
3        i := i + 1
4  if i ≤ n
5        pos := 1
6  else
7        pos := 0
8  return (pos)
9  end Busqueda-Lineal
```

Ejemplo 2.2.8. El algoritmo siguiente describe el método de *búsqueda binaria*:
Entrada: "x"(Dato que desea encontrar en el arreglo), A (Lista de elementos ya ordenados en forma creciente).
Salida: Posición de a en la lista, si a esta presente en la misma.

PROCEDURE BUSQUEDA-BINARIA (A, x)

```
1  i := 1
2  j := length(A)
3  while i < j
4        m := ⌊(i + j)/2⌋
5        if x > A[m]
6              i := m + 1
7        else
8              j := m
9  if x = A[i]
10       localizacion := i
11 else
12       localizacion := 0
13 return (localizacion)
14 end Busqueda-Binaria
```

Observación 2.2.2. En los algoritmos:

1. Para llamar a un procedimiento que regrese un valor, escribir CALL NOMBRE (A, B, C, ..., W)

2. Hasta aquí, nos hemos ocupado de problemas de representación de algoritmos sin considerar la cuestión de cómo se obtienen los algoritmos por primera vez. Sin embargo, el descubrimiento de algoritmos suele ser el paso más difícil en la creación de software; después de todo, descubrir un algoritmo es equivalente a encontrar un método para resolver el problema cuya solución el algoritmo va a calcular. Por tanto, entender cómo se descubren los algoritmos es entender el proceso de resolución de problemas.

3. Las técnicas para resolver problemas y la necesidad de saber más acerca de ellos no son exclusivas de las ciencias de la computación, sino que son temas pertinentes para casi cualquier campo. Así, la capacidad para resolver problemas sigue siendo más una aptitud artística que debe desarrollarse, que una ciencia precisa por aprender.

4. Como prueba de la naturaleza imaginativa y artística de la resolución de problemas, las fases definidas a grandes rasgos por el matemático G. Polya a finales de los años cuarenta del siglo pasado siguen siendo los principios fundamentales en que se basan los intentos actuales de enseñar a resolver problemas.

5. Los científicos de la computación suelen citar a Polya porque sus trabajos abordaron la resolución de problemas en el contexto matemático, pariente cercano de la computación. Las fases de Polya para resolver problemas son:

Fase 1: Entender el problema

Fase 2: Idear un plan para resolver el problema

Fase 3: Llevar a cabo el plan.

Fase 4: Evaluar la solución en cuanto a su exactitud y a su potencial como herramienta para resolver otros problemas.

Traducidas el contexto de creación de programas, estas fases se convierten en:

Fase 1: Entender el problema.

Fase 2: Pensar cómo un algorítmo podría resolver el problema.

Fase 3: Formular un algoritmo y representarlo en forma de programa.

Fase 4: Evaluar el programa en cuanto a su exactitud y a su potencial como herramienta para resolver otros problemas.

6. No olvidemos que estamos hablando de la forma en cómo se resuelven los problemas, no de cómo nos gustaría que se resolvieran. Muchas personas hábiles en resolver problemas a menudo comienzan formulando estrategias para resolver un problema (Fase 2) antes de entender totalmente el problema (Fase 1). Después si fallan tales estrategias (Fase 3 y 4), la persona comprende con mayor profundidad los detalles del problema y, con base en ésta mejor comprensión, puede volver a diseñar otras estrategias que, con suerte, tendrán éxito.

   Idealmente querríamos eliminar el desperdicio inherente al proceso de ensayo y error que acabamos de escribir. En el caso de la creación de sistemas de software de gran tamaño, descubrir una mala comprensión cuando ya se está en la **fase 4** puede representar una tremenda prédida de recursos. Evitar tales catástrofes es un objetivo primordial en los ingenieros de software.

7. Otra irregularidad en la elaboración de enfoques sistemáticos para resolver problemas, es la misteriosa inspiración que puede llegar a un potencial solucionador de problemas, quien después de trabajar con un problema sin éxito aparente, puede ver de repente la solución mientras realiza otra tarea.

   El científico H. Von Helmontz identificó este fenómeno ya desde 1896, y el matemático Henri Poincare lo analizó en una conferencia ante la Sociedad Psicológica de París. Ahí Poincare describió su experiencia de encontrar la solución a un problema con el que había trabajado después de haberlo dejado de lado e iniciado otros proyectos.

   Es como si una parte subconsciente del intelecto siguiera trabajando y, en caso de tener éxito, obligaría a la mente consciente a reconocer la evolución. Hoy día, al período entre el trabajo consciente con un problema y la llegada de la inspiración repentina se le llama **período de incubación**, y su comprensión es una meta de varias investigaciones actuales.

8. Por última instancia, pues el descubrimiento de algoritmos sigue siendo un difícil arte que se debe desarrollar con el tiempo, y no enseñarse sólo como una materia aislada que abarque metodologías bien definidas.

Por supuesto, hay muchas estrategias para resolver problemas y cualquiera de ellas puede ser muy fructífera en una situación en la que parece haber un hilo común en todas ellas, y que en términos sencillos puede expresarse como dar el *"primer paso"*.

Ejemplo 2.2.9. A continuación se muestra las 4 Fases de Polya que resuelve el problema de hallar la solución a la ecuación cuadrática: $ax^2 + bx + c = 0$.

Fase 1: La ecuación es cuadrática si $a \neq 0$ y tiene dos soluciones reales diferentes o dos soluciones reales iguales o dos soluciones complejas diferentes.

Fase 2: Las soluciones de la ecuación cuadrática se llaman raíces y se calculan por:

$$R_1, R_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

donde el discriminante $\triangle = b^2 - 4ac$ determina como son las raíces:

$$\begin{aligned} &\text{Si } \triangle > 0, \text{ raíces reales diferentes.} \\ &\text{Si } \triangle = 0, \text{ raíces reales iguales.} \\ &\text{Si } \triangle < 0, \text{ raíces complejas.} \end{aligned}$$

Fase 3:

Entrada: a, b, c
Salida: R_1, R_2

```
PROCEDURE RAICES (a, b, c)
 1  if a = 0
 2       print ("Ecuación Lineal")
 3  △ := b^2 − 4 * a * c
 4  if △ > 0
 5       print ("Raíces reales diferenciales")
 6       R1 := (−b + √△)/2 * a
 7       R2 := (−b − √△)/2 * a
 8  else
 9       if △ = 0
10            print ("R1 raíz de multiplicidad 2")
11            R1 := −b/2 * a
12       else
13            print ("Raíces complejas conjugadas")
14            R1 := (−b + √△ * i)/2 * a
15            R2 := (−b − √△ * i)/2 * a
16  return (R1, R2)
17  end Raices
```

Fase 4: El procedimiento Raíces resuelve la ecuación $ax^2 + bx + c = 0$ cualesquiera fueran los coeficientes: a, b, c y es una herramienta muy útil en muchos campos de la matemática, la ingeniería, la economía, etc.

2.3. Conjuntos

Muchos conceptos en la matemática y en la ciencia de la computación pueden ser convenientemente expresados en el lenguaje de los conjuntos.

Las definiciones son importantes en cualquier ciencia, pues contribuyen a una comunicanión sin ambigúedad. Por lo tanto, tener un punto de partida para nuestras definiciones en el cual los significados estén claros es de suma importancia, nuestro punto de partida será la idea de un conjunto.

Aún cuando revisemos aquí los principales conceptos que se utilizan en la Teoría de Conjuntos, suponemos que el lector ya está familiarizado con el lenguaje de los conjuntos.

Un conjunto es *una colección ordenada de objetos distintos que por lo general se denominan elementos o miembros.* Por ejemplo tenemos los conjuntos:

$$
\begin{aligned}
A &= \{a, b, c\} \\
B &= \{x/x \text{ es la capital del Perú}\} \\
\mathbb{N} &= \{0, 1, 2, 3, 4, \ldots\} \\
\mathbb{Z} &= \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\} \\
\mathbb{Z}^+ &= \{1, 2, 3, 4, \ldots\} \\
\mathbb{Q} &= \text{Conjunto de todos los números racionales.} \\
\mathbb{R} &= \text{Conjunto de todos los números reales.} \\
\mathbb{C} &= \text{Conjunto de todos los números comlejos.} \\
\phi &= \{\} \text{ “Conjunto vacío”.}
\end{aligned}
$$

Si A es un conjunto, $a \in A$, significa que a es un elemento de A. Escribimos $a \notin A$ cuando a no sea un elemento de A. *Dos conjuntos son iguales si contienen los mismos elementos.*

Si A y B son dos conjuntos, $A \subseteq B$ significa que todos los elementos de A pertenecen también a B; y se lee A subconjunto de B. La notación $A \subset B$ significa que $A \subset B$ y que $A = B$ y se lee A *subconjunto propio de* B.

Dado un conjunto A, podemos crear un nuevo conjunto cuyos elementos sean todos los subconjuntos de A. Este nuevo conjunto que denotaremos $P(A)$ es llamado *conjunto potencia* o conjunto de partes de A. $P(A)$ tendrá, por lo menos como elemento al ϕ y al propio A, pues $\phi \subseteq A$ y $A \subseteq A$ es siempre verdadero. Para calcular el número de elementos de $P(A)$ usamos:

$$|P(A)| = 2^n$$

donde n es el número de elementos del conjunto A o cardinalidad de A

Dado un conjunto arbitrario U, podemos definir algunas operaciones en el conjunto $P(A)$. U en este caso, es llamado el conjunto universal o “universo de discurso”. El conjunto universal define el contexto de los objetos en cuestión

Si $A, B \in P(U)$ se define la:

$A \cup B = \{x \in U / x \in A \vee x \in B\}$	Unión
$A \cap B = \{x \in U / x \in A \wedge x \in B\}$	Intersección
$\overline{A} = \{x \in U / x \in \overline{A} \notin A\}$	Complemento
$A - B = A \cap \overline{B}$	Diferencia
$A \oplus B = (A - B) \cup (B - A)$	Diferencia simétrica
$A \triangle B = (A \cup B) - (A \cap B)$	

Las operaciones definidas para realizarse con conjuntos satisfacen las siguientes propiedades:

Propiedades Idempotentes

$$\begin{aligned} A \cup A &= A \\ A \cap A &= A \end{aligned}$$

Propiedades Conmutativas

$$\begin{aligned} A \cup B &= B \cup A \\ A \cap B &= B \cap A \end{aligned}$$

Propiedades Asociativas

$$\begin{aligned} A \cup (B \cup C) &= (A \cup B) \cup C \\ A \cap (B \cap C) &= (A \cap B) \cap C \end{aligned}$$

Propiedades Distributivas

$$\begin{aligned} A \cap (B \cup C) &= (A \cap B) \cup (A \cap C) \\ A \cup (B \cap C) &= (A \cup B) \cap (A \cup B) \end{aligned}$$

Propiedades del Complemento

$$\begin{aligned} \overline{\overline{A}} &= A \\ A \cup \overline{A} &= U \\ A \cap \overline{A} &= \phi \\ \overline{\phi} &= U \\ \overline{U} &= \phi \end{aligned}$$

Leyes de Morgan

$$\left.\begin{aligned} \overline{A \cup B} &= \overline{A} \cap \overline{B} \\ \overline{A \cap B} &= \overline{A} \cup \overline{B} \end{aligned}\right\} \text{Leyes de Morgan}$$

Propiedades del Conjunto Universal

$$\begin{aligned} A \cup U &= U \\ A \cap U &= A \end{aligned}$$

Propiedades del Conjunto Vacío

$$A \cup \phi = A$$
$$A \cap \phi = \phi$$

Propiedades de Absorción

$$A \cup (A \cap B) = A$$
$$A \cap (A \cup B) = A$$

Observe que los nombres y las formas de éstas propiedades son muy parecidas con las equivalencias lógicas de la lógica de proposiciones. Veremos más adelante que ésta semejanza no es una coincidencia.

El producto cartesiano de dos conjuntos A y B, representado por $A \times B$, es el conjunto de todos los pares ordenados de la forma (a, b), donde $a \in A$ y $b \in B$, observe que por lo general, $A \times B \neq B \times A$ ya que la disposición de los componentes de un par ordenado es significativa.

Es posible generalizar el concepto del producto cartesiano de conjuntos para obtener el producto de más de dos conjuntos. Por ejemplo, $A \times B \times C = \{(a, b, c)/a \in A, b \in B, c \in C\}$. Es común utilizar la abreviatura A^2 para representar $A \times A$. Así, de manera general A^n representará la colección de todas las n-tuplas de elementos de A.

Dados los conjuntos A y B, una función de A en B es un subconjunto de $A \times B$, tal que cada elemento de A aparece como el primer elemento de uno y sólo uno de los pares ordenados de la función. Si f es una función de A en B diremos, que A es el dominio de f. El conjunto de elementos que aparece como la segunda componente de los pares ordenados de una función se denomina contradominio o rango de f. Si cada elemento de B se halla en el rango de f, diremos que f es una función sobreyectiva. Si cada miembro del rango de f está asociado a sólo un elemento del dominio, diremos que f es una función inyectiva. Las funciones que sean inyectivas y sobreyectivas las llamaremos biyectivas.

Por lo general, los elementos del dominio de una función se consideran como entradas para la función y los elementos del rango como los valores de salida. Este concepto de entrada y salida origina la notación $f : A \to B$, lo cual representa una función f de A en B. Dada una función $f : A \to B$, representamos como $f(x)$ al valor de salida asociado a la entrada x. Por ejemplo, suponga que $f : N \to N$ es la función que consiste en los pares ordenados de la forma (n, n^2), es decir $f(n) = n^2$ para cada $n \in N$. Esta función es inyectiva pero no es sobreyectiva.

Consideremos ahora el concepto intuitivo del número de elementos en un conjunto.

Es obvio que el conjunto (a, b, c) tiene 3 elementos y que la gente diría que el conjunto N tiene una infinidad de elementos, pero esto, para nuestros propósitos, es demasiado simplista. Pues, como veremos enseguida los conjuntos con una infinidad de elementos, llamados conjuntos infinitos, no contienen la misma cantidad de elementos. El número de elementos de los conjuntos infinitos pueden variar, al igual que el número de elementos de los conjuntos finitos. Por esto, existen varios niveles de infinidad, donde algunos conjuntos infinitos son mayores que otros.

Ya que nuestro sistema contable tradicional no existen números que representen el número de elementos de un conjunto infinito, es necesario ampliar nuestro sistema para tener en cuenta estos conjuntos. En este sistema ampliado nos referimos al número de elementos de un conjunto como su cardinalidad. La

cardinalidad de un conjunto finito corresponde al concepto tradicional del número de elementos de un conjunto, pero la cardinalidad de un conjunto infinito no es un número, en el sentido habitual. Más bien, la cardinalidad de un conjunto infinito es un número, llamado **número cardinal** que existe únicamente en nuestro sistema de numeración ampliado. La cardinalidad de un conjunto A se representa con $\mid A \mid$.

Para completar nuestro sistema de cardinalidad es necesario establecer una forma de comparar las cardinalidad de dos conjuntos diferentes, es decir, se requiere definir qué significa el hecho de que una cardinalidad sea mayor que otra. Para esto se establece: $\mid A \mid \leq \mid B$ si y sólo si existe una función inyectiva de A en B. Observe que si A y B son conjuntos finitos, esta definición equivale a la definición tradicional: $\mid a, b \mid \leq \mid a, y, z \mid$ ya que puede establecerse la función inyectiva $\{(a, x), (b, y)\}$.

A continuación, se dice: $\mid A \mid = \mid B \mid$ si y sólo si existe una función biyectiva de A en B. El teorema de Schrörder-Bernstein, cuya demostración no es importante ahora, confirma que estas definiciones están de acuerdo con nuestra intuición:

$\mid A \mid = \mid B \mid$ si y sólo si: $\mid A \mid \leq \mid B \mid$ y $\mid B \mid \leq \mid A \mid$

Por último definimos: $\mid A \mid < \mid B \mid$ si y sólo si $\mid A \mid \leq \mid B \mid$ y $\mid A \mid \not\leq \mid B \mid$.

Si A es un conjunto finito a nuestro lector seguramente ya le han mostrado que $\mid P(A) \mid = 2^{|A|}$ y por tanto podrá aceptar que: $\mid A \mid < \mid P(A) \mid$. Resultado que se puede demostrar, que se extiende a conjuntos infinitos.

Ahora nos encontramos en un punto donde podemos apoyar nuestra afirmación de que los conjuntos infinitos puede tener cardinalidades distintas. Basta seleccionar un conjunto infinito A y comparar su cardinalidad con la de su conjunto potencia P(A).

Es decir:

$$\mid A \mid < \mid P(A) \mid < \mid P(P(A)) \mid < ...$$

Una consecuencia de esta jerarquía es que no existe ningún cardinal infinito mayor. Existe, no obstante, un cardinal infinito menor. El conjunto $\mathbb{N}$ es uno de los conjuntos infinitos mś pequeños ($\mid \mathbb{N} \mid = X$ "alef cero"), Decimos que es uno de los más pequeños ya que existen varios conjuntos con cardinalidad X. Por ejemplo: $\mid \mathbb{N} \mid = \mid \mathbb{Z} \mid$, la biyección $f\colon \mathbb{N} \to \mathbb{Z}$ definida por:

$$f(n) = \begin{cases} \frac{n}{2} & \text{si n es par} \\ \\ \frac{-1+n}{2} & \text{si n es impar} \end{cases}$$

establece dicho resultado (considerando a 0 como par).

A quí resulta útil recordar el proceso al que normalmente nos referimos como contar. Contar los elementos de un conjunto significa asignar los valores $1, 2, 3, \ldots, a$ los elementos del conjunto, incluso podemos pronunciar las palabras "uno, dos, tres,..."conforme señalamos los objetos durante el recuento. En consecuencia, un conjunto A es contable (o enumerable) si existe una biyección de $\mathbb{Z}^+$ en A.

Sin embargo, hemos visto que existen conjuntos infinitos con más elementos que $\mathbb{Z}^+$. Por consiguiente, no es posible contar estos conjuntos ya que no hay enteros positivos suficientes. Se dice que estos conjuntos son incontables. Por otra parte, se dice que un conjunto es contable si su cardinalidad es menor o igual que la de los enteros positivos.

El conjunto $\mathbb{N}$ es contable; no así su conjunto potencia, pues

$$| \mathbb{Z}^+ | = | \mathbb{N} | (| P(\mathbb{N} |))$$

De hecho, el conjunto potencia de cualquier conjunto infinito es incontable.

Una forma de mostrar que un conjunto infinito es contable consiste en demostrar que existe un proceso para representar una lista en secuencia de sus elementos de manera que cada elemento del conjunto aparezca en la lista. Como ejemplo se mostrará que $\mathbb{Q}^+$ es contable, describiendo la elaboración de una lista de todos los elementos de $\mathbb{Q}^+$, que la haremos a partir de la siguiente matriz:

$$\begin{array}{cccccc} 1/1 & 1/2 & 1/3 & 1/4 & 1/5 & \dots \\ 2/1 & 2/2 & 2/3 & 2/4 & 2/5 & \dots \\ 3/1 & 3/2 & 3/3 & 3/4 & 3/5 & \dots \\ 4/1 & 4/2 & 4/3 & 4/4 & 4/5 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{array}$$

Para obtener la lista de todos los elementos de $\mathbb{Q}^+$, usaremos la enumeración determinada arriba, teniendo presente que debemos eliminar las fracciones para que no estén escritas en su forma ampliada, esto evita el problema de listar $\frac{1}{2}$ y $\frac{2}{4}$ por ejemplo, que representan el mismo número racional.

Relacionemos estos resultados con un ambiente de computación típico. Supongamos que tenemos un programa listo para ejecutarse en un computador. El programa está diseñado para aceptar cierta entrada y producir cierta salida. Sin importar cuáles sean la entrada y la salida, desde el punto de vista de la máquina no son más que una cadena de bits.

Por tanto, podemos considerar a la entrada y a la salida como números naturales representados en notación binaria (podemos considerar un 1 adicional colocado en el extremo izquierdo de las cadenas de bits, de modo que los 0 iniciales de las cadenas originales sean siginificativos). Desde esta perspectiva, el programa acepta un número natural como entrada y produce un número natural como salida. Es decir, la acción del programa es calcular una función de N en N. A la vez, que cada función de N en N sugiere un programa que en alguna ocasión podríamos escribir.

¿Cuántas funciones de éstas existen? Obviamente existen por lo menos tantas funciones de N en N como elementos tiene P(N). Por lo tanto, existe una cantidad incontable de funciones por las cuales podríamos escribir programas.

Ahora considere un lenguaje de programación preferido y establezca que el conjunto de los programas que pueden escribirse en ese lenguaje es contable ("es un reto").

¿Qué hemos mostrado con esto? Sólo existe una cantidad contable de programas que pueden escribirse en su lenguaje de programación preferido, pero hay una cantidad incontable de funciones, por ende problemas que se quisieran resolver utilizando un computador, por ende hay problemas para los cuales no pueden escribirse un programa en su lenguaje favorito.

¿Cuáles son algunos de éstos problemas?
¿Es posible resolver alguno de ellos cambiando a otro lenguaje de programación?
¿Podrían resolverse estos problemas si construimos un computador más grande?
¿Éstas son algunas de las preguntas que trata directa o indirectamente las ciencias de la computación?

2.4. Conjuntos en la Programación

El concepto de conjuntos es un concepto útil en una noción general que figura como tipo de datos en algunos lenguajes de programción, tales como el C++. En esos lenguajes el concepto universal U precisa ser especificado y entonces las variables que representan subconjuntos de U, esto es, elementos de P(U), pueden ser definidas.

Existe un límite de tamaño para el conjunto universal de forma que los elementos de $P(U)$ no puedan ser arbitrariamente grandes; es decir, el conjunto universal precisa ser contable.

Si el conjunto no fuera establecido como un tipo de dato básico en un lenguaje de programación entonces el programador puede implementar las ideas de conjuntos teóricos, implementando un tipo abstracto de datos para "conjuntos". Un tipo abstracto de datos es nada más que un modelo mental de una colección de ítems relacionados, juntamente con las operaciones apropiadas a ser realizadas entre instancias de esas colecciones.

En este caso, el modelo mental es un conjunto, y las operaciones apropiadas son las operaciones entre conjuntos, tales como unión, intersección y complemento. El programador debe usar las facilidades por el lenguaje para simular los conjuntos y las operaciones.

Algunos de los conjuntos más importantes se originan en relación con la sucesión. *Una sucesión es una lista de objetos dispuestos en orden creciente de los enteros positivos.* Así por ejemplo:

$$1, 0, 1, 1, 1, 1, 0, 1, 1, 2, 4, 9, 16, 25, ...$$

En el primer caso, se dice que la sucesión es finita y en el segundo caso, que la sucesión es infinita.

El concepto relacionado con sucesión en Ciencias de la Computación es el arreglo lineal o lista. Se hará una distinción simple pero útil entre sucesión y arreglo y se usará una notación ligeramente diferente. Si se tiene una sucesión $S : S_1, S_2, S_3, \ldots$, se piensa que todos los elementos de S están plenamente determinados.

Por otra parte, si se cambia cualesquiera de los elementos S_i, se tiene una nueva sucesión y se le llamará de alguna manera de S. Un arreglo, en cambio, puede considerarse como una sucesión de posiciones formada por casillas o celdas, como se muestra en la figura:

S =

$S_{[1]}$	$S_{[2]}$	$S_{[3]}$	$S_{[4]}$	$S_{[5]}$	...	$S_{[N]}$

Los elementos de algún conjunto pueden ser asignados a las posiciones del areglo S. El elemento asignado a la posición i será denotado por $S_{[i]}$, y a la sucesión $S_{[1]}, S_{[2]}, \ldots, S[n]$ se le llamará sucesión de valores del arreglo S y el número N se denominará el tamaño del arreglo. El punto por observar es que el arreglo S se considera bien definido, aún cuando a algunas posiciones no se les haya asignado o si se cambian algunos valores durante la discusión.

Un concepto muy útil para los conjuntos es la **Función Característica**. Si A es un conjunto de un conjunto universal U, la función característica F_A de A se define con:

$$f_A(x) = \begin{cases} 1 & \text{si} \quad x \in A \\ 0 & \text{si} \quad x \notin A \end{cases}$$

Teorema 2.1. *Si $A, B \in P(U)$, entonces:*

a) $f_{A\cap B} = f_A \times f_B$

b) $f_{A\cup B} = f_A + f_B - (f_A \times f_B)$

c) $f_{A\oplus B} = f_A + f_B - 2f_A \times f_B$

Demostración. Tenemos:
a.

$$f_A \times f_B = \begin{cases} 1 & \text{si} \quad x \in A \text{ y } x \in B \\ 0 & \text{si} \quad x \notin A \text{ o } x \notin B \end{cases} = \begin{cases} 1 & \text{si} \quad x \in (A \cap B) \\ 0 & \text{si} \quad x \notin (A \cup B) \end{cases}$$

entonces

$$f_{A\cap B} = f_A \times f_B$$

b. Se tiene los siguientes casos:

$$\text{Si } x \in A \text{ entonces : } \begin{cases} f_{A\cup B}(x) = 1 \\ f_A(x) + f_B - f_A \times f_B(x) = 1 + f_A(x) - f_A(x) = 1 \end{cases}$$

$$\text{Si } x \in B \text{ entonces : } \begin{cases} f_{A\cup B}(x) = 1 \\ f_A(x) + f_B - f_A \times f_B(x) = f_A(x) - f_A(x) = 1 \end{cases}$$

$$\text{Si} x \notin A \text{ y } x \notin B \text{ entonces : } \begin{cases} f_{A\cup B}(x) = 0 \\ f_A(x) + f_B - f_A \times f_B(x) = 0 + 0 - 0 = 1 \end{cases}$$

Entonces

$$f_{A\cup B} = f_A + f_B - (f_B f_A \times f_B)$$

□

Para representar conjuntos en una computadora, debe disponerse los elementos del conjunto Universal en una sucesión. La sucesión particular seleccionada no tiene importancia alguna. Por ejemplo, si:

$$\begin{aligned} U &= \{a, c, e, i, m, t\} \\ A &= \{c, e, m, t\} \\ B &= \{a, i, m\} \end{aligned}$$

entonces:
f_A corresponde a la sucesión 0, 1, 1, 0, 1, 1

f_B corresponde a la sucesión 1, 0, 0, 1, 1, 0

Este hecho permite representar nuestros conjuntos como arreglos:

$$U = \boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{1}$$

$$A = \boxed{0}\boxed{1}\boxed{1}\boxed{0}\boxed{1}\boxed{1}$$

$$B = \boxed{1}\boxed{0}\boxed{0}\boxed{1}\boxed{1}\boxed{0}$$

y recíprocamente, el arreglo:

$$C = \boxed{1}\boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{1}$$

representa al conjunto $C = \{a, i, t\}$.

La representación en arreglos de bits toma las operaciones de conjuntos más simples de ser implementados, como veremos en el siguiente ejemplo:

Ejemplo 2.4.1. El siguiente algoritmo especifica un arreglo C que representa al conjunto $C = A \oplus B$ en el contexto de un conjunto universal de N elementos:

Entrada: Arreglos A y B de igual tamaño
Salida: Arreglo C de tamaño igual a B y A

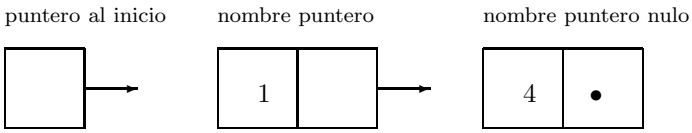
PROCEDURE DIFERENCIA-SIMETRICA(A, B)

```
1  for i := 1 to length(A)
2        if A[i] ≠ B[i]
3              C[i] := 1
4        else
5              C[i] := 0
6  return (C)
7  end Diferencia-Simetrica
```

Otra posibilidad es usar listas enlazadas para representar a los conjuntos. Una lista enlazada es una lista con punteros que conectan cada ítem de la lista al siguiente. Un puntero nulo (un puntero que no dá para ningún lugar) marca el fín de una lista. Algunos lenguajes de programación ofrecen un tipo de dato de puntero, en otros lenguajes, los punteros precisan ser simulados usando los recursos del lenguaje de programación.

La representación a través de una lista enlazada presenta la ventaja de que no importa una limitación a priori del tamaño del conjunto, pero contiene la importancia del orden en sus elementos. Las operaciones entre los conjuntos se realizan a través de recorrer, comparar, insertar y deleteo en las listas enlazadas.

Ejemplo 2.4.2. El subconjunto $A = \{1, 4\}$ puede ser representado por la lista enlazada mostrada en la figura de abajo, donde el "punto"representa el puntero nulo.



La programación orientada a objetos se basa en la idea de un tipo abstracto de dato un poco más elaborado a través de la posibilidad de encapsulamiento; es la posibilidad de definir un tipo de dato junto con las operaciones apropiadas para manipulación de sus instancias, todo en un paquete con sección de código que todo el resto del programa no logra accesar.

Un tipo de objeto de conjunto, SET, podría ser definido y encapsulado juntamente con los procedimientos pata hallar el complemento de un conjunto o la unión e intersección de dos conjuntos. Variables (objetos) que fuesen instancias del tipo objeto SET pueden llamar esos procedimientos.

Otra idea que es importante en la programción orientada a objetos es la herencia. Una vez que el tipo objeto haya sido definido se puede crear un tipo "hijo"que herede todas sus propiedades y operaciones y adicione propiedades u oraciones locales especializadas conforme a necesidades. De esta forma, el código que ya estaba escrito en el tipo "padre"es un código reutilizable. Esto reduce la cantidad de código que precisa ser escrito. Lo importamte es que el tipo "hijo"es esencialmente un subconjunto del tipo "padre", entonces todos los elementos del tipo "hijo"heredan las propiedades del tipo "padre", asimismo poseen alguna propiedad local.

2.5. Inducción

Uno de los conceptos más útiles de las matemáticas básicas en el diseño y análisis de algoritmos es la inducción matemática. No sólo nos permitirá demostrar propiedades interesantes acerca de la corrección y eficiencia de los algoritmos, sino que además puede incluso utilizarse para determinar que propiedades es preciso probar.

Antes de discutir la técnica, se ha de indicar una digresión acerca de la naturaleza del decubrimiento científico. En la ciencia hay dos enfoques opuestos fundamentales: **Inducción y deducción. La inducción consiste en inferir una ley general a partir de casos particulares, mientras que una deducción es una inferencia de lo general a lo particular.**

Veremos, que aún cuando la inducción puede dar lugar a conclusiones falsas, no se puede despreciar. La deducción, por otra parte; siempre es válida con tal de que sea aplicada correctamente.

En general, no se puede confiar en el resultado del razonamiento inductivo. Mientras que haya casos que no hayan sido considerados, sigue siendo posible que la regla general inductiva sea incorrecta. Por ejemplo si se observa que:

$$\begin{aligned} 1^3 &= 1 &&= 1^2 \\ 1^3 + 2^3 &= 9 &&= 3^2 \\ 1^3 + 2^3 + 3^3 &= 36 &&= 6^2 \\ 1^3 + 2^3 + 3^3 + 4^3 &= 100 &&= 10^2 \\ 1^3 + 2^3 + 3^3 + 4^3 + 5^3 &= 225 &&= 15^2 \end{aligned}$$

No podemos convencernos inductivamente que la suma de los cubos de los números enteros positivos es siempre un cuadrado perfecto. Resulta en este caso que el razonamiento inductivo proporciona una ley correcta. Si todavía somos más perceptivos quizá nos demos cuenta de que esta suma de cubos es precisamente el cuadrado de la suma de los primeros números enteros positivos.

En contraste, se observa que, la expresión de todos los días no puede haber convencido inductivamente que siempre es posible ingresar una persona mas en un autobús. Pero unos instantes de pensamiento nos muestra que la regla es absurda.

Como ejemplo más matemático, de inducción incorrecta considérese el polinomio:

$$p(n) = n^2 + n + 41$$

y observe que:

$$\begin{aligned} p(0) &= 41 \text{ Es un número primo} \\ p(1) &= 43 \text{ Es un número primo} \\ p(2) &= 47 \text{ Es un número primo} \\ p(3) &= 53 \text{ Es un número primo} \\ p(4) &= 61 \text{ Es un número primo} \\ p(5) &= 71 \text{ Es un número primo} \\ p(6) &= 83 \text{ Es un número primo} \\ p(7) &= 97 \text{ Es un número primo} \\ p(8) &= 113 \text{ Es un número primo} \\ p(9) &= 131 \text{ Es un número primo} \\ p(10) &= 151 \text{ Es un número primo} \end{aligned}$$

Por tanto es natural inferir por indución que $p(n)$ es primo para todos los valores de n. Pero de hecho $p(40) = 1681 = 41^2$ no es primo.

Un ejemplo más sorprendente de inducción es el dado por una conjetura de Euler, que formuló en 1789. ¿Es posible que la suma de tres cuartas potencias sea una cuarta potencia? Formalmente, es posible encontrar enteros positivos a, b, c tales que:

$$a^4 + b^4 + c^4 = d^4$$

Al no poder encontrar un ejemplo de este comportamiento, Euler conjeturó que ésta ecuación nunca se podría satisfacer. (Esta conjetura está relacionada con el último teorema de Fermat). Transcurrieron más de dos siglos antes

que Elkies en 1987 descubriese el primer contraejemplo, que implicaba números de siete y ocho cifras. Posteriormente ha sido demostrado por parte de Frye, utilizando cientos de horas de tiempo de computación en varias máquinas conectadas, que el único contraejemplo en el que d es menor que un millón es :

$$(95800)^4 + (217519)^4 + (414560)^4 = (422481)^4$$

La ecuación de Pell proporciona un caso todavía más extremo de razonamiento inductivo tentador pero incorrecto. Considérese el polinomio:

$$p(n) = 991 * n^2 + 1$$

La pregunta es si existe un entero positivo "n"tal que $p(n)$ es un cuadrado perfecto. Si uno va probando valores para n, resulta cada vez más tentador suponer inductivamente que la respuesta es negativa. Pero de hecho se puede obtener un cuadrado perfecto en este polinomio.

La solución más pequeña se obtiene cuando:

$$n = 12055735790331359447442538767$$

Es muy probable que el lector se pregunte en este momento porqué es importante utilizar inducción, si es proclive a errores, en lugar de utilizar deducción que es una prueba de bombas.

Hay razones fundamentales para el uso de la inducción en el proceso del descubrimiento científico. Por ejemplo, los físicos precisan utilizar el enfoque inductivo para determinar las leyes fundamentales que gobiernan el universo a partir de datos reales obtenidos de experimentos. Aun cuando uno sea físico teórico, tal como Einstein, sigue siendo necesario experimentos reales que otras personas hayan efectuado. Por ejemplo, Halley predijo el retorno de su cometa homónimo por razonamiento inductivo, y también por razonamiento inductivo Mendelev predijo no sólo la existencia de elementos químicos todavía no decubieros, sino también sus propiedades químicas.

Con seguridad, ¿sólo será legítimo en matemáticas y en las rigurosas ciencias de la computación la deducción? Después de todo las propiedades matemáticas como el teorema consistente en que existen infinitos números primos se puede demostrar de una forma deductiva rigurosa, sin datos experimentales. Los razonamientos inductivos deberían eliminarse de las matemáticas "verdad? ¡Pues no! En realidad la matemática puede ser también una ciencia experimental.

No es infrecuente que un matemático descubra una verdad matemática considerando varios casos especiales e infiriendo a partir de ellos por inducción una regla general que parece plausible. Sin embargo, independientemente de lo tentadora que sea la evidencia, esa verdad matemática no se puede aceptar basándose en la evidencia inductiva solamente. La diferencia entre la matemática y las ciencias inherentemente experimentales es que una vez que se haya descubieto por inducción una ley matemática general debemos demostrarla rigurosamante aplicando el proceso deductivo. Así el proceso inductivo tiene su lugar en las matemáticas. En caso cotrario , ¿Como podríamos esprerar comprobar rigurosamente un teorema cuyo enunciado ni siquiera ha sido formulado? Por lo tanto, la inducción es necesaria para formular conjeturas, y la deducción

es igualmente necesario para demostrarlas, o a veces para demostrar su falsedad. Ninguna de éstas técnicas puede ocupar el lugar de la otra. La deducción sólo es suficiente para las "matemáticas muertas"o congeladas, tal como los elementos de Euclides (que quizás sea el mayor monumento de la historia a las matemáticas deductivas, cuando no cabe duda de que gran parte de su material fue descubierto por razonamiento inductivo). Pero se necesita la inducción para mantener vivas a las matemáticas.

Tal como Polya dijo una vez *"las matemáticas presentadas sin rigor son una ciencia deductiva sistemática, pero las matemáticas que se están haciendo son una ciencia inductiva experimental".*

Por último, anecdóticamente una de las técnicas deductivas más útiles que están disponibles en matemáticas tiene la mala fortuna de llamarse inducción matemática. Esta terminología resulta confusa, pero tenemos que vivir con ella.

Para ilustrar el uso de ésta técnica, imagine que usted está subiendo en una escalera sin fín. Como usted puede saber si será capaz de alcanzar una grada arbitrariamente alta. Suponga que usted tiene las siguientes afirmaciones sobre sus habilidades de subir escaleras:

1. Usted puede alcanzar la primera grada

2. Si usted puede alcanzar una grada, usted puede siempre pasar a la grada siguiente

Si la sentencia 1 como la condicional 2 son verdaderas, por la sentencia 1 usted puede llegar a la primera grada y por la sentencia 2 usted puede llegar a la segunda; nuevamente por 2 usted puede llegar a la tercera y así sucesivamente, usted puede subir tan alto como usted quiera. En esta caso, ambas aserciones son necesarias. Si apenas la sentancia 1 es verdadera, usted no tiene garantías de que podrá ir más allá de la primera grada y si apenas la segunda sentencia es verdadera, usted no podría llegar a la primera grada a fin de iniciar el proceso de subida de la escalera.

A este tipo particular de deducción, en general se le llama principio de *Inducción Matemática*; que de manera formal enunciamos a continuación:

2.5.1. Principio de Inducción Matemática

Sea P(n) una proposición en términos de n, con $n \in \mathbb{Z}^+$ tal que se cumple:

a. P(1) es verdadera

b. Si P(k) es verdadera para k¿1 implica que P(k+1) sea verdadera.

Entonces $\forall n P(n)$ es verdadera.

La condición "a"se llama paso base y la condición "b"paso inductivo. De aquí en adelante, significa inducción matemática. Ahora podemos aplicar la técnica de demostración por inducción en problemas menos obvios.

Ejemplo 2.5.1. Consideramos un juego de vídeo que empieza con una nave espacial a mitad de la pantalla. En 5 segundos aparece un extraterrestre. Cinco segundos después el extraterrestre se divide en dos, y estos dos extraterrestres se subdividen en dos, y así sucesivamente. Es decir, cada cinco segundos el número

de extraterrestres se duplica. La tarea del jugador es eliminarlos antes de cubran la pantalla.

Supongamos que el jugador no es muy hábil y que todos sobreviven.

¿Cuántos habrá 30 segundos después de empezado el juego?

Calculando el número de extraterrestres en intervalos de 5 segundos, obtenemos la tabla siguiente:

Tiempo	5	10	15	20	25	30
Extraterrestres	1	2	4	8	16	32

Por lo que la respuesta a nuestra pregunta es 32 extraterrestres en 30 segundos.

Ahora supongamos que queremos saber el número de estos seres en 5 minutos. Podemos simplemente ampliar la tabla hacia la derecha. Pero hay evidentemente un camino más fácil. Sea $P(n)$ el número de extraterrestres en la pantalla después de n intervalos de 5 segundos. Entonces $P(1) = 1$, $P(2) = 2$, $P(3) = 4$, $P(4) = 8$, $P(5) = 16$, $P(6) = 32$ que al razonar en forma inductiva conjeturamos que:

$$P(n) = 2^{n-1}, \forall n \in \mathbb{Z}^+$$

Y evidentemente se verifica para los datos de la tabla. ¿cómo podemos estar seguros de que lo que esperamos es correcto sin hacer todos los cálculos?

Como vimos, la respuesta a esta pregunta es utilizando el Principio de Inducción. Así:

Demostración. $P(n) = 2^{n-1}, \forall n \in \mathbb{Z}^+$

Paso Base

$$\begin{aligned} P(1) &= 2^{t-1} \\ &= 2^0 \\ &= 1 \text{ es verdadera} \end{aligned}$$

Paso Inductivo

Se debe demostrar ahora que $P(k)$ es verdadera para $k > 1$, entonces $P(k+1)$ también debe ser verdadera.

Se supone que para algún valor fijo $k > 1$, $P(k) = 2^{k-1}$ llamada también hipótesis de inducción es verdadera.

Ahora se debe demostrar que $P(k+1)$ sea verdadera:

$$\begin{aligned} P(k+1) &= 2 * P(k), \text{ pues el número de extraterrestres se duplica} \\ &= 2 * 2^{k-1}, \text{ por hipótesis de inducción} \\ &= 2^{(k+1)-1} \end{aligned}$$

Es decir, $P(k+1)$ es verdadera. Por lo tanto por el principio de Inducción $P(n) = 2^{n-1}$ es verdadera. $\forall n \in \mathbb{Z}^+$; entonces:

$$\begin{aligned} P(96) &= 2^{96-1} \\ &= 2^{95} \end{aligned}$$

es la respuesta a nuestro problema. □

Ejemplo 2.5.2. Ilustramos qué la inducción es útil en matemáticas. Observemos que:

$$\begin{aligned} 2 &= 1*2 \\ 2+4 &= 2*3 \\ 2+4+6 &= 3*4 \\ 2+4+6+8 &= 4*5 \end{aligned}$$

Al seguir estos cálculos el razonamiento inductivo nos hace conjeturas:

$$P(n) = 2+4+6+8+\ldots+2n = n(n+1), \forall n \in \mathbb{Z}^+$$

que demostraremos por el principio de inducción:

Demostración. $P(n) = 2+4+6+8+\ldots+2n = n(n+1), \forall n \in \mathbb{Z}^+$
Paso Base

$$P(1) = 2 = 1(1+1)\ , \quad \text{es verdadera}$$

Paso Inductivo

Supongamos que para algún valor fijo $k > 1$

$$P(k) = 2+4+6+8+...+2k = k(k+1), \text{ es verdadera}$$

Ahora debemos demostrar que P(k+1) es verdadera:

$$\begin{aligned} P(k+1) = 2+4+6+8+\ldots+2k+2(k+1) &= 2+4+6+8+...+2(k+1) \\ &= (2+4+6+...+2k)+2(k+1) \\ &= k(k+1)+2(k+1)\ , \text{ por H.I} \\ &= (k+1)(k+2) \\ &= (k+1)((k+1)+1) \end{aligned}$$

Es decir, $P(k+1)$ es verdadera.
Por lo tanto por el principio de inducción

$$2+4+6+8+\ldots+2n = n(n+1)\ \ \forall n \in \mathbb{Z}^+$$

□

Hasta el momento, la base de la inducción ha sido 1, pero esto no es realmente un riquisito. De hecho, se puede comenzar la inducción con cualquier elemento n_0, tomando $P(n_0)$ como base de inducción.

Ejemplo 2.5.3. Al intentar hallar una relación entre 2^n y $n!$:

$$\begin{aligned} 2^1 &> 1! & \qquad 2^5 &< 5! \\ 2^2 &> 2! & \qquad 2^6 &< 6! \\ 2^3 &> 3! & \qquad 2^7 &< 7! \\ 2^4 &< 4! & \qquad 2^8 &< 8! \end{aligned}$$

por un razonamiento inductivo, podemos afirmar que:

$$P(n) = 2^n < n! \forall n \in \mathbb{Z}^+, n \geq 4$$

que demostraremos por el principio de inducción:

Demostración. $P(n) = 2^n < n! \forall n \in \mathbb{Z}^+, n \geq 4$

Paso Base

$$2(4) = 2^4 < 4! \text{ es verdadera}$$

Paso Inductivo
Supongamos que para algún valor fijo $k > 4$, $P(k)$ es verdadera.

Ahora demostraremos que $P(k+1)$ es verdadera:
Sabemos que:

$$\begin{aligned} &2^k < k! \text{ ; por hipótesis de inducción, entonces} \\ &2 * 2^k < 2 * k! < (k+1) * k!, \text{ pues } k > 4 \\ &2^{k+1} < (k+1)! \end{aligned}$$

entonces $P(k+1)$ es verdadera.

Por lo tanto $\forall P(n),\ n \geq 4$ es verdadera. □

El principio de inducción descrito hasta el momento es adecuado para demostrar muchas afirmaciones interesantes. Existen casos, sin embargo, en los cuales es preferible un principio algo más potente, llamado *Principio de Inducción Matemática Generalizada*. La situación se ilustra mediante el siguiente ejemplo: supongamos que se desea demostrar, por el principio de inducción, que para todo entero positivo $n \geq 2$, n es un número primo o es el producto de primos. Si saltamos directamente al paso inductivo, cuando se intenta demostrar que k+1 se puede expresar como un producto de números primos (suponiendo que sea producto de números primos), la hipótesis de inducción sería que también se puede descomponer en producto de números primos el número k.

Sin embargo, no podremos encontrar algo en la decomposición de k en números primos que pueda ser útil o relevante para la descomposición de k+1 en primos. Lo que se necesita realmente es una hipótesis más fuerte, consistente en que todo número que es producto de primos menos que k+1 se puede dscomponer en un producto de números primos.

Al formalizar estos conceptos se tiene el principio de Inducción Matemática Generalizada lo que enumeramos a continuación:

2.5.2. Principio de Inducción Matemática Generalizada

Sea P(n) una proposición enunciada en términos de n, con $n \in \mathbb{Z}^+$ tal que se cumple:

a. P(1) es verdadera

b. P(k) es verdadera par todo n $K < n$ implica que $P(n)$ sea verdadera

Entonces $\forall n P(n)$ es verdadera.

Ejemplo 2.5.4. Mostraremos como el principio de inducción matemática generalizada permite concluir que:

Demostración. P(n) = n es un número primo o es el producto de números primos, $\forall n \in \mathbb{Z}^+$
Paso Base

P(1) = 1 es un número primo, es verdadera.

Paso Inductivo

Supongamos que para todo k, $1 \leq k < n, P(k)$ es verdadera.

Ahora debemos demostrar que$P(n)$ es verdadera.

Si n es primo, $P(n)$ es verdadera:

Si no n es primo, entonces n es un número compuesto y puede ser escrito como $n = a * b$, donde $a < n$ y $b < n$ entonces por hipótesis de inducción a y b son primos o producto de primos. Luego n es el producto de números primos, entonces $P(n)$ es verdadera. □

Ejemplo 2.5.5. Consideremos un grupo de n personas, $n \geq 1$.

Observemos que:

$n =$ 1 , implica que no hay ningún apretón de manos de saludo
$n =$ 2 , implica que hay un apretón de manos de saludo
$n =$ 3 , implica que hay 2+1 apretones de manos de saludo
$n =$ 4 , implica que hay 3+2+1 apretones de manos de saludo

Luego, haciendo un razonamiento inductivo, se puede conjeturar:

$P(n) = (n+1)+(n+2)+\ldots+2+1$, es el número de apretones de manos de saludo, $\forall n \in \mathbb{Z}^+$ que demostraremos por el principio de inducción generalizada:

Demostración. $P(n) = (n+1)+(n+2)+\ldots+2+1$
Paso Base

$$\begin{aligned} P(1) &= 1-1 \\ &= 0 \text{ apretones de manos de saludo, es verdadera} \end{aligned}$$

Paso Inductivo

Supongamos que para todo k, $k < n$, $P(k)$ es verdadera.

Ahora demostraremos que $P(n)$ es verdadera:

Como $n-1 < n$, por hipótesis de inducción $P(n-1)$ es verdadera, entonces:

$$\begin{aligned} P(n) &= (n-1) + P(n-1)\ , \text{ apretones de manos de saludo} \\ &= (n-1) + (n-2) + \ldots + 2 + 1, \text{ apretones de manos de saludo.} \end{aligned}$$

Entonces P(n) es verdadera.
Por lo tanto $\forall n P(n)$ es verdadera.

□

Para que usted pueda probar la equivalencia entre estos dos principios de inducción, le presentamos otro principio llamado **principio del buen orden**. Toda colección de números enteros que contenga por lo menos un elemento tiene un elemento mínimo. Debemos entonces verificar que las siguientes sean verdaderas.

Inducción matemática generalizada	$\Longrightarrow$	Inducción matemática
Inducción matemática	$\Longrightarrow$	Principio del buen orden
Principio del buen orden	$\Longrightarrow$	Inducción matemática generalizada

2.6. Inducción en la verificación de programas

La construcción de programas se basa en argumentos relacionales y por lo tanto parece razonable aplicar técnicas de comprobación formales que demuestren si un programa funciona correctamente. Las técnicas formales que se utilizan para demostrar que los programas son correctos y son rentables, pues reduce el número de errores lo que a su vez reduce el esfuerzo que se necesita en la depuración. Lo esencial en la verififcación de programas son las precondiciones y postcondiciones. Las precondicones indican las condiciones que deben satisfacer los datos de entrada para que el programa pueda cumplir su tarea. Las postcondiciones, indican que condiciones de salida son aceptables como soluciones correctas del problema en cuestión. Por ejemplo, en el programa que calcule una raíz cuadrada: la precondición es que el argumento de la raíz cuadrada no debe ser negativo y la postcondición que el resultado final sea la raíz cuadrada deseada.

Se utilizará la palabra código o partes de un código para designar cualquier parte de un código, desde una simple sentencia hasta el programa entero.

Para demostrar que las diferentes partes del código funcionan correctamente, se deben asociar las precondiciones y las postcondiciones que corresponden a cada parte del código. Despuś se debe demostrar que las precondiciones y postcondiciones de todas las partes del código son compatibles entre sí y a su vez, con las precondiciones y postcondiciones de todo el programa. De hecho, cuando el programa es ejecutado, las postcondiciones de cada parte del código deben incluir en esencia las precondiciones de la siguiente parte. Esto debe realizarse sin importar lo largas que resulten las partes del código.

De hecho, estas consideraciones son todavía más importantes en la construcción y correción de programas grandes: en los programas grandes hay una probabilidad mayor de cometer errores y es más difícil localizarlos.

Cuando se realiza la verificación de programas, es necesaria una regla de inferencia para tratar con cada tipo de sentencia ejecutable. Concretamente, nos encontraremos con una regla para las sentencias de asignación, otra regla para las sentencias if y otra para *while*, en la que no sólo hay que probar que el programa es correcto, si termina y cuando termina, sino también hay que probar que el programa realmente termina.

Los programas se dividen en partes de un programa que son secuencias de sentencias que transforman el estado inical en un estado final. El estado inicial es el estado enterior a la ejecución de la parte de un código. El estado del sistema a su vez viene dado por los valores de la variable tal y como aparecen en las declaraciones.

Cada sentencia lógica relativa a un estado se llama *aserción*. Las dos aserciones más importantes son las precondiciones y postcondiciones. Una precondición es una aserción que hace referencia al estado inicial de la parte de un código y una postcondición es una aserción de su estado final.

Invariablemente, las postcondiciones de un trozo de código ha sido ejecutado, y las precondiciones proporcionan las condicones que han de ser satisfechas para que el código termine con las postcondiciones estipuladas.

La parte de un código se considera correcta si una vez el código ha sido ejecutado, todos los estados que satisfacen las precondiciones llevan a estados que satisfacen las postcondiciones.

Nótese, sin embargo, que esta discución no aborda la cuestión si las precondiciones y postcondiciones realmente reflejan lo que el programador tenía en mente.

Además, dado el diseño esta cada vez más formalizado, a menudo es posible obtener unas precondiciones y postcondiciones rigurosamante definidas.

En esta sección se supone que el lenguaje para escribir un código contine todos los operadores aritméticos, una amplia biblioteca de funciones, palabras clave para el inicio y final de bloques de programa, las construcciones *if-then-else* y la construcción de lazo *while*.

No se emplean declaraciones. En todos los casos el tipo de variable se deduce del contexto. Además no se utilizan sentencias de entrada-salida. La entrada puede tener lugar antes de la primera sentencia de código que se está considerando y la salida puede tener lugar tras la última sentencia.

Como ya se ha mencionado, los programas trabajan con estados de programa y estos estados de programa se modifican por la acción de las partes del código, las partes del código son funciones que proyecta el estado inicial sobre el estado final, que es el estado al finalizar la parte del código.

Varias partes de código pueden estar concatenadas, lo que simplemente significa que se ejecutan secuencialmente. Concretamente, si C_1 y C_2 son dos partes del programa, su concatenación se representa como C_1; C_2. La concatenación es realmente la composición de funciones. La concatención de dos o más partes del código también constituyen una parte del código.

Para indicar que una sentencia A es una aserción, a menudo se encierra entre llaves $\{A\}$. Si las llaves dan lugar a confusiones éstan obviamente se omiten.

Definición 2.2. Si C es una parte del código, entonces cualquier aserción de

$\{P\}$ se denomina precondición de C si $\{P\}$ sólo implica el estado inicial. Cualquier aserción $\{Q\}$ se denomina postcondición si $\{Q\}$ sólo implica el estado final. Si C tiene como precondición a $\{P\}$ y como postcondición a $\{Q\}$, se escribe $\{P\}C\{Q\}$. La terna $\{P\}C\{Q\}$ se denomina **Terna de Hoare**.

Ejemplo 2.6.1. Consideremos el código que consiste en una sóla sentencia $x := 1/y$.

Una precondición de esta sentencia es $\{y\}$ y una postcondición es $\{x := 1/y\}$.

Ejemplo 2.6.2. La sentencia a:=b, determina la terna de Hoare: $\{\}$ $a := b$ (a=b) Donde la aserción vacía $\{\}$, se puede leer como "verdadero para todos los estados posibles".

Definición 2.3. Si C es un código en la precondición $\{P\}$ y la postcondición $\{Q\}$, entonces $\{P\}C\{Q\}$ se dice que es parcialmente correcto si el estado final de C satisface $\{Q\}$ siempre que el estado inicial satisfaga $\{P\}$. C también sería parcialmente correcto si no hay estado final debido a que el programa no termina. Si $\{P\}C\{Q\}$ es parcialmente correcto y C termina, entonces se dice que $\{P\}C\{Q\}$ es totalmente correcto.

En otras palabras, un código no es totalmente correcto si existen algunos estados iniciales que satisfacen P que conducen a un bucle infinito. Sin embargo tales programas pueden ser parcialmente correctos. El código sin bucles siempre termina y la corrección parcial entonces implica una corrección total. De ahí que la distinción entre corrección parcial y total sea sólo esencial en códigos que contengan bucles o procedimientos recursivos.

Como se ha indicado anteriormente, las precondiciones y postcondiciones son importantes para realizar verificaciónes. Sin embargo, la importancia de éstas condiciones es aún más general. Éstas le ayudan a uno a clarificar los objetos del código que se estén investigando. Además insertando precondiciones y postcondiciones a las partes del código de un programa, la verficación se puede realizar de un manera más sistemática: si se han satisfecho las precondiciones de algún código, pero sin haberse cumplido las postcondiciones, entonces el error debe haberse producido dentro de ese código de cuestión. Eso permite simplificar la búsqueda de errores.

Ahora explicaremos cómo verificar un código que contenga una sentencia *while*. Primero sólo estudiaremos la corrección parcial. En otras palabras, demostraremos que si el código termina su estado final satisface la postcondición.

Las características que no varían son capturadas por un invariante, el invariante del bucle.

La negación de la condición de entrada es la condición de salida. Una vez que se ha satisfecho la condición de salida, el bucle termina. La variante del bucle es una expresión que mide el progreso realizado hasta satisfacer la condición de salida.

Para ilustrar los conceptos principales, consideremos el código del siguiente algoritmo.

Entrada: m, n enteros no negativos.
Salida: x

```
Procedure Producto(m, n)
1  j := 1
2  x := n
3  while (j < n)
4        x := x + n
5        j := j + 1
6  return (x)
7  end producto
```

La condición de entrada de este bucle es $J \neq M$. Si ésta expresión es falsa al comienzo del bucle termina. Por esta razón a la negación de la condición de entrada se le llama condición de salida. En nuestro caso, la condición de salida es $(J \neq M)$ o $J = M$. Al final de la parte del código, $X = M * N$, y esta condición es la postcondición. Durante cada iteración se progresa hasta la postcondición. Al final de cada iteración, se tiene $X = J * N$, esta aserción resulta ser la invariante del bucle.

La variante del bucle es M-J. La variante del bucle decrece con cada iteración y tan pronto como llega a cero el bucle termina. A la salida del bucle, J=M y J*N = M*N.

Por lo tanto, después de su terminación X=M*N que es la postondición. La regla de inferencia para demostrar la corrección de los bucles *while* se puede representar de un modo más formal en el cuadro siguiente:

$$\frac{\{I\}C\{I\}}{\{I\}\ \textit{while}\ D\ \text{do}\ C\{\ D \wedge I\}}$$

Donde I es el invariante de un código C; D es la condición de entrada y D es la condición de salida.

Para utilizar esta regla, primero se tiene que demostrar que I es verdadera al final de cada iteración, esto es demosrtar: $\{I\}C\{I\}$. En este punto es que la inducción entra en escena. Demostraremos con f(n) la sentencia f: y debemos demostrar que la invariante del ciclo es verdadera para n-iteraciones de ciclo. Como no sabemos necesariamente cuantas iteraciones de ciclo puede ejecutarse (o sea por cuantos pasos la condición D permanece verdadera), nosotros debemos demostrar que I(n) es verdadera para todo $n \geq 0$.

Por lo tanto, el bucle no termina hasta que la condición de entrada D no sea falsa al final de una iteración, ya que I se cumple al final de cada iteración incluyendo la iteración que provoca la terminación (demostrada previamente por inducción), tanto I como D deben mantenerse cuando el bucle haya terminado. Esto demuestra la validez de la regla *while*.

No demostremos el hecho que el ciclo *while* tenga un fin de ejecución. Lo que demostraremos es su corrección parcial: el programa produce el resultado deseado, suponiendo que su ejecución llega al fín.

Demostración. $I(n) = X_n = J_n * N, n \geq 0$

Paso Base

$$\begin{aligned} I(0) &= X_0 = 0 \\ &= J_0 * N \ \text{ es verdadera} \end{aligned}$$

Paso Inductivo

Supongamos que para algún valor fijo k¿0, I(k) es verdadera, es decir: $I(k) = X_k * N$ Ahora debemos demostrar que I(k+1) es verdadera:

$$\begin{aligned} I(k+1) = X_{k+1} &= X_k + N \\ &= J_k * N + N, \text{ por hipótesis de inducción} \\ &= (J_k + 1) * N \\ &= Jk + 1 * N \end{aligned}$$

Entonces, I(k+1) es verdadera. Por lo tanto I(n) es verdadera, para todo $n \geq 0$, es decir $\{I\}C\{I\}$. □

Lo más habitual es que el invariante del bucle sea precursor de la postcondición, lo que significa de que alguna manera debe ser parecido a la postcondición. Además de esto, en cada iteración debe progresarse hacia la postcondición, lo que significa que el invariante del bucle debe contener todas la variables que son modificadas dentro del bucle.

El concepto de invariante del bucle también es útil en la construcción de programas. En la construcción de programas normalmente hay un objetivo que indica lo que hay que realizar al final de cada bucle. Este objetivo se puede formalizar y esta formalización dá como resultado el invariante del bucle. Por lo tanto, la verificación y la programación van codo a codo. En particular, si el objetivo que hay que alcanzar al final de cada bucle se especifica claramente, es mucho más fácil hallar el invariante del bucle. En este sentido el invariante del bucle solamente es una formalización de los objetivos del programador.

2.7. Lista de Ejercicios

1. Describa un algoritmo que calcule las suma de todos los enteros de una lista.

2. Describa un algoritmo que tenga como entrada una lista de n enteros distintos y devuelva la posición del mayor entero par de la lista o devuelva 0 si no hay enteros pares.

3. Escriba un algoritmo en pseudocódigo cuya salida sea el menor elemento de la sucesión $a_1, a_2, \ldots, a_n$

4. Un palíndromo es una cadena que se lee igual de izquierda a derecha que al revés. Describa un algoritmo para determinar si una cadena de n caracteres es un palíndromo.

5. Describa un algoritmo para calcular x^n, donde x es un número real y n es un entero. (*indicación:* primero desarrolle un procedimiento para calcular x^n cuando n no es negativo mediante multiplicaciones sucesivas de x, comenzando por 1. Luego extiende el procedimiento y utiliza el hecho de que $x^{-n} = 1/x^n$ para calcular x^n cuando n es negativo.)

6. Describa un algoritmo que intercambie los valores de las variables x e y utilizando sólo asignaciones. ¿Cuál es el número mínimo de asignaciones necesarias?

7. Enumere los pasos realizados en la búsqueda del número 9 en la sucesión $1, 2, 3, 4, 5, 6, 8, 9, 11$ utilizando:
   a) una búsqueda lineal
   b) una búsqueda binaria
8. Describa un algoritmo que localice la primera aparición del mayor elemento de una lista finita de enteros, donde los enteros de la lista no son necesariamente distintos.
9. Describa un algoritmo para obtener el mayor y el menor valor de una sucesión finita de enteros.
10. Describa un algoritmo que cuente los bits 1 que aparecen en una cadena de bits examinando cada bit para determinar si es o no un 1.
11. El **algoritmo búsqueda ternaria** localiza elementos en una lista de enteros dados en orden creciente dividiendo sucesivamente la lista en tres sublistas de igual tamaño (o de tamaño tan parecido como sea posible) y restringiendo la búsqueda al fragmento apropiado. Especifique los pasos de este algoritmo.
12. Describa un algoritmo que obtenga todos los términos de una sucesión finita de enteros que sean mayores que la suma de todos los términos previos.
13. Utilice el método de la burbuja para ordenar $d, f, k.m.a, b$ mostrando las listas obtenidas en cada caso.
14. Escriba un algoritmo en pseudocódigo que determine el máximo de tres números.
15. Escriba un algoritmo en pseudocódigo que determine el máximo de una sucesión finita.
16. Escriba un algoritmo en pseudocódigo que verifique si un entero positivo es primo.
17. Escriba un algoritmo en pseudocódigo que determine un primo mayor que un entero dado
18. Escriba un algoritmo en pseudocódigo cuya salida sea el índice de la última ocurrencia del elemento máximo de la sucesión $a_1, a_2, \ldots, a_n$.
19. Escriba un algoritmo que invierta la sucesión $a_1, a_2, \ldots, a_n$
20. Suponga que el arreglo A consta de los números reales $A[1], A[2], \ldots, A[n]$ y que el arreglo B consta de los números reales $B[1], B[2], \ldots, B[n]$. Escriba un algoritmo en pseudocódigo que calcule$A[1]B[1] + A[2]B[2] + \ldots + A[n]B[n]$
21. Sea $A[1]$, $A[2]$,..., $A[n]$ el arreglo asociado a los coeficientes $a_1, a_2, \ldots, a_n$ de un polinomio $\sum_i^N a_1 x^i$. Escriba un algoritmo en pseudocódigo con el arreglo A y las variables n y x como entradas y como salida proporcione el valor del polinomio en x.

22. Sean A y B arreglos de longitud N con ceros y unos, y suponga que representan subconjuntos, que también llamaremos A y B, de algún conjunto universal U con N elementos. Escriba un algoritmo en pseudocódigo para especificar el arreglo C que represente al conjunto $C = A \cap (A \oplus \overline{B})$

23. Obtenga una fórmula para la suma de los n primeros enteros positivos pares.

24. Use la inducción matemática para demostrar la fórmula obtenida en el problema anterior.

25. Sea n un entero positivo. Dibujar un círculo y marcar n puntos espaciados regularmente alrededor de la circunferencia. Ahora dibujar una cuerda dentro del círculo entre cada pareja de puntos. En el caso n=1 no hay parejas de puntos y por tanto no se dibuja ninguna cuerda. Denotaremos con c(n) el número de secciones que se construyen así dentro del círculo.

    a. Hallar c(1), c(2), c(3), y c(4)

    b. Por inducción, ¿cuál cree usted que será la fórmula general para c(n)?

    c. Determine c(5) dibujando y contando. ¿Era correcta la fórmula hallada inductivamente? Vuelva a intentarlo con c(6)

    d. ¿Qué pasa si se permite que los puntos se espacien irregularmente?

    e. Determinar la fórmula correcta para c(n), y demostrar que es correcta.

26. Demuestre por inducción matemática la validez de las siguientes conjeturas:

    a) $2^0 + 2^1 + 2^2 + 2^3 + \cdots + 2^n = 2^{n+1} - 1$; $\forall n \in \mathbb{N}$

    b) $1^2 + 3^2 + 5^2 + \cdots + (2n-1)^2 = \frac{n(2n+1)(2n-1)}{3}$; $\forall n \in \mathbb{Z}^+$

    c) $1 + a + a^2 + \cdots + a^{n-1} = \frac{a^n - 1}{a-1}$; $\forall n \in \mathbb{Z}^+$

    d) $a + ar + ar^2 + \cdots + ar^{n-1} = \frac{a(1-r^n)}{1-r}$; $r \neq 1$; $\forall n \in \mathbb{Z}^+$

    e) $1 + 2^n < 3^n$; $n \geq 2$

    f) $n < 2^n$; $n > 1$

    g) $1 + 2 + 3 + \cdots n < \frac{(2n+1)^2}{8}$; $\forall n \in \mathbb{Z}^+$

    h) $3^n < n!$; $n \geq 6$

    i) $n! < n^n$; $\forall n \in \mathbb{Z}^+$

    j) $1^2 - 2^2 + 3^2 - \cdots + (-1)^{n-1} n^2 = (-1)^{n-1} \frac{n(n+1)}{2}$; $\forall n \in \mathbb{Z}^+$

    k) $1 + \frac{1}{4} + \frac{1}{9} + \cdots + \frac{1}{n^2} < 2 - \frac{1}{n}$; $n > 1$

    l) Si un conjunto A tiene n elementos, entonces $\mathcal{P}(A)$ tiene 2^n elementos.

    m) $3 | \left(n^3 - n\right)$; $\forall n \in \mathbb{Z}^+$

n) Si $A_1,\ A_2\ ,\ \cdots\ ,\ A_n$ son n conjuntos cualesquiera, entonces $\overline{\left(\bigcap_{i=1}^{n} A_i\right)} = \bigcup_{i=1}^{n} \overline{A_i}$.

o) Si $A_1,\ A_2\ ,\ \cdots\ ,\ A_n$ son n conjuntos cualesquiera, entonces $\overline{\left(\bigcup_{i=1}^{n} A_i\right)} = \bigcap_{i=1}^{n} \overline{A_i}$.

p) Si $A_1,\ A_2\ ,\ \cdots\ ,\ A_n\ ,\ y\ B$ son conjuntos cualesquiera, entonces $\left(\bigcap_{i=1}^{n} A_i\right) \bigcup B = \bigcap_{i=1}^{n} (A_i \bigcup B)$.

q) Si p es un número primo y $p|a^n$ para $n \geq 1$, entonces $p|a$.

r) Si $MCD\,(a,b) = 1$, entonces $MCD\,(a^n, b^n) = 1$; $\forall n \in \mathbb{Z}^+$.

27. Encuentre el entero positivo más pequeño n_0 tal que $2^{n_0} > n_0^2$. Luego demuestre por inducción matemática que $2^n > n^2$ para todos los valores $n \geq n_0$.

28. ¿Para que enteros no negativos n se cumple que $2n + 3 \leq 2^n$? Demuestra tu respuesta por inducción matemática.

29. Sea $P(n)$ la propiedad $2|\,(2n-1)$.

    a) Demuestre que $P(k) \to P(k+1)$ es valida.

    b) Demuestre que no existe un entero n para el cual $P(n)$ sea verdadera.

    c) ¿Contradicen los resultados de las partes (a) y (b) el principio de inducción matemática?. Explique

30. Use inducción matemática para demostrar que n líneas rectas en el plano lo dividen en $\frac{(n^2+n+2)}{2}$ regiones. Suponga que no hay dos líneas paralelas y que no hay tres líneas con un punto en común.

31. Demuestre que las regiones del ejercicio anterior pueden colorearse de rojo y verde de modo que no haya dos regiones que comparten una orilla que sean del mismo color.

32. Suponga que $n > 1$ personas se colocan en un campo (plano euclidiano) de manera que cada una tiene un vecino más cercano único. Aún más, suponga que cada persona tiene un pastel que lanza a su vecino más cercano. Un sobreviviente es una persona a la que no le pega un pastel.

    a) Proporcione un ejemplo para mostrar que si n es par, puede no haber un sobreviviente.

    b) De un ejemplo para demostrar que puede haber más de un sobreviviente.

    c) Use inducción matemática sobre n para demostrar que si n es impar, siempre habrá al menos un sobreviviente.

d) Pruebe o desapruebe: Si n es impar, una de las personas más separadas es un sobreviviente.

e) Pruebe o desapruebe: Si n es impar, una de las dos personas más separadas es un sobreviviente.

33. Demuestre por inducción matemática, que el siguiente algoritmo usado correctamente produce el resultado establecido.

```
Procedure Potencial (x, y)
 1  z := 0
 2  w := y
 3  while w > 0
 4        z := z + x
 5        w := w − 1
 6  w := y − 1
 7  u := z
 8  while w > 0
 9        z := z + u
10        w := w − 1
11  return (z)
12  end Potencial
```

34. Demuestre por inducción matemática, que el siguiente algoritmo usado correctamente, produce el resultado establecido.

```
Procedure Potencia(n, m)
1  r := n
2  p := 2 ∗ m
3  while p > 0
4        r := r ∗ n
5        p := p − 1
6  return (r)
7  end Potencia
```

Capítulo 3

Relaciones

3.1. Relaciones

Los elementos de un conjunto o los elementos de conjuntos diferentes frecuentemente presentan comparaciones especiales entre si, que pueden ser descritas como una relación.

Definición 3.1. Una relación binaria R del conjunto A en el conjunto B es un subconjunto de $A \times B$.

- Si R es una relación binaria en $A \times B$, entonces R consistirá en un conjunto de pares ordenados de la forma (a, b) y escribiremos $R : A \leftrightarrow B$.
- si $(a, b) \in R$ diremos que a esta relacionada con b bajo la relación R y escribiremos aRb.
- la relación $R \subseteq A \times B$ define dos subconjuntos, uno de A y otro de B, estos son:

$$Dom(R) = \{a/a \in A, (a, b) \in R \text{ para algún } b \in B\}$$
$$Ran(R) = \{b/b \in A, (a, b) \in R \text{ para algún } a \in A\}$$

  y se conocen como el dominio y rango de R, respectivamente.
- la relación de *uno_a_uno* (o inyectiva o biunívoca) si cada primer componente a y cada segunda componente b aparecen a lo sumo una vez en la relación. La relación es *uno_para_varios* si alguna primera componente a aparece más de una vez, es decir, si a fuera par con más de un b. La relación es *varios_para_uno* (o unívoca) si alguna segunda componente b fuera par con más de una a. Finalmente la relación es llamada *varios_para_varios* si por lo menos un b fuera par con mas de un a.
  La figura 3.1 ilustra esas cuatro posibilidades.

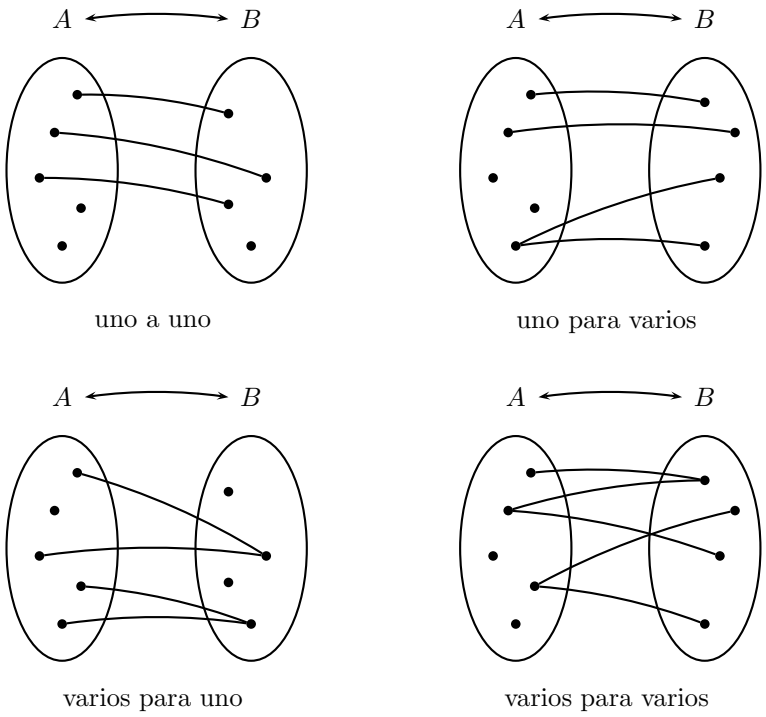


Figura 3.1: Tipos de Relaciones

Ejemplo 3.1.1. Sean los conjuntos $A = \{2, 5, 7, 9\}$ y $B = \{3, 4, 5\}$ y las relaciones:

$$R = \{(5, 3), (7, 5), (9, 3)\}$$
$$S = \{(2, 4), (5, 5), (7, 3)\}$$
$$T = \{(7, 4), (2, 5), (9, 4), (2, 3)\}$$

Entonces:

$$Dom(R) = \{5, 7, 9\}$$
$$Dom(S) = \{2, 5, 7\}$$
$$Dom(T) = \{7, 2, 9\}$$
$$Ran(R) = \{3, 5\}$$
$$Ran(S) = \{4, 5, 3\}$$
$$Ran(S) = \{4, 5, 3\}$$

Observemos que R es *varios_para_uno*, S es *uno_a_uno* y T *varios_para_varios*

Definición 3.2. Una relación n_aria en los conjuntos A_1, A_2, ..., A_n es un subconjunto de $A_1 \times A_1 \times \ldots \times A_1$

- Una relación n_aria es un conjunto de n_tuplas. Estas relaciones se utilizan en todas las bases de datos relaciones para el almacenamiento y acceso a datos.

- Aquí nos concentraremos en las relaciones binarias. De aquí en adelante, la palabra relación, cuando sea utilizada, se referirá por defecto a las relaciones binarias.

- Existen muchos métodos para representar relaciones binarias. Los métodos gráficos son particularmente útiles para visualizar relaciones. Por otra parte, es frecuente que para realizar operaciones matemáticas que incluyan relaciones sea más conveniente representarla como matrices. Por supuesto, las relaciones son conjuntos, y los métodos para representar conjuntos se pueden utilizar también para representar relaciones.

- Por ejemplo, suponga que A es el conjunto de proveedores y B es el conjunto de productos. Específicamente $A = \{a, b\}$ y $B = \{p, q, r\}$. Ahora se puede definir una relación $B : A \leftrightarrow B$ afirmando que: $xRy \Leftrightarrow x$ tiene en existencia al producto y.

$$R = \{(a, p), (a, r), (b, q), (b, r)\}$$

Significa (en particular) que el proveedor a tiene existencia del producto p.
Esta Relación queda definida plenamente por la tabla.

R	p	q	r
a	1	0	1
b	0	1	1

El número 1 de una celda particular de la tabla indica que el suministrador tiene el producto, mientras que un 0 indica que no lo tiene. las tablas están estrechamente relacionadas con las matrices, para convertir una tabla en una matriz, simplemente se eliminan los encabezamientos. En nuestro caso esto produce

$$M_r = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Formalicemos esta idea, si R es una relación, de $A = \{a_1, a_2, \ldots, a_m\}$ en $B = \{b_1, b_2, \ldots, b_n\}$: $M_r = [M_{ij}]_{m \times n}$ denota la matriz de esta relación, donde:

$$M_r = \begin{cases} 1, & \text{si } a_i R b_j; \\ 0, & \text{si } a_i \not R b_j. \end{cases}$$

- las relaciones se pueden representar gráficamente. Para representar una relación A en B, dibujamos un círculo para cada elemento de A a la izquierda, y dibujamos un círculo para cada elemento de B a la derecha. Si el par $(x, y) \in R$ los círculos correspondientes que llamaremos nodos o vértices, serán conectados entre si con una flecha que va de nodo x hacia nodo y, a estas flechas las denominaremos arcos. La figura resultante la se denomina grafo dirigido o digrafo. Por ejemplo, la siguiente figura representa la relación Proveedor $\leftrightarrow$ Producto. Se representa gráficamente en la figura 3.2.

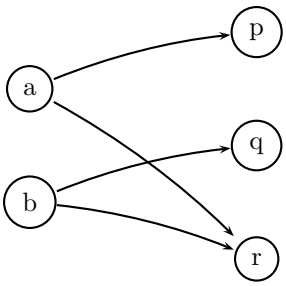


Figura 3.2: Digrafo de R

- Una relación R de A en A se denomina una relación en A. El hecho de que R está sobre A hace posible dibujar el digrafo de R como sigue:

  Cada elemento de A se representa mediante un nodo, y dos nodos cualesquiera están conectados por un arco si $(x, y) \in R$. Considere, por ejemplo, la relación $\leqq$ en $A = \{1, 2, 3, 4\}$, que se representa por el siguiente digrafo

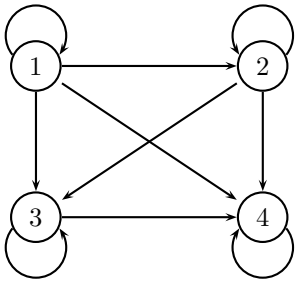


Figura 3.3: Un digrafo

- Generalmente los arcos que comienzan y terminan en el mismo nodo se denominan bucles o giros (slings).

3.2. Manipulación de relaciones

Al igual que puede manipularse o trabajarse con los números y formulas aplicando las reglas del álgebra, también pueden definirse operaciones que permitan manipular las relaciones. Con estas operaciones se puede cambiar, combinar y refinar relaciones para producir otras nuevas.

3.2.1. Relación Inversa

Definición 3.3. Si $R : A \leftrightarrow B$ es una relación inversa $R^{-1} : B \leftrightarrow A$ se define por $aR^{-1}b \Leftrightarrow bRa$. Todas las propiedades de conjuntos pueden aplicarse a las relaciones, Para ser formal presentamos la siguiente definición.

Definición 3.4. sean R y S relaciones de A en B, entonces la relación:

a) Complemento $\overline{R} : A \leftrightarrow B$ y puede expresarse:

$$a\overline{R}b \Leftrightarrow a \not R b$$

b) Unión $R \cup S : A \leftrightarrow B$ y puede expresarse:

$$a(R \cup S)b \Leftrightarrow aRb \vee aSb$$

c) Intersección $R \cap S : A \leftrightarrow B$ y puede expresarse:

$$a(R \cap S)b \Leftrightarrow aRb \wedge aSb$$

Ejemplo 3.2.1. Sea $A = \{\alpha, \beta, \gamma, \theta\}$ y $B = \{1, 2, 3\}$
Sean:

$$R = \{(\alpha, 1), (\alpha, 2), (\beta, 2), (\beta, 3), (\gamma, 2), (\theta, 1)\}$$
$$S = \{(\alpha, 2), (\beta, 3), (\gamma, 2), (\theta, 2)\}$$

Entonces:

$$R^{-1} = \{(1, \alpha), (2, \alpha), (2, \beta), (3, \beta), (2, \gamma), (1, \theta)\}$$
$$\overline{S} = \{(\alpha, 1), (\alpha, 3), (\beta, 1), (\beta, 2), (\gamma, 1), (\gamma, 3), (\theta, 1), (\theta, 3)\}$$
$$R \cup S = \{(\alpha, 1), (\alpha, 2), (\beta, 2), (\beta, 3), (\gamma, 2), (\theta, 1), (\theta, 2)\}$$
$$R \cap S = \{(\alpha, 2), (\beta, 3), (\gamma, 2)\}$$

3.2.2. Composición de Relaciones

Definición 3.5. Sea R una relación de A en B y sea S una relación de B en C. La composición de relaciones R con S es la relación $R \circ S$ de A en C definida por:

$$a(R \circ S)c \Leftrightarrow \exists b \in B / aRb \text{ y } bSc$$

.

Ejemplo 3.2.2. Si $A = \{1, 2, 3, 4, 5\}$, $B = \{a, b, c, d\}$, $C = \{x, y, z\}$,

$R = \{(1, b), (2, d), (4, c), (5, d)\}$, $S = \{(b, x), (d, y), (b, z)\}$,

entonces:

$$R \circ S = \{(1, x), (1, z), (2, y), (5, y)\}$$

Observación 3.2.1. En las relaciones:

a) Las matrices $m \times n$ cuyas entradas son cero y uno se denominarán booleanas, y en ellas es natural definir operaciones booleanas que tiene aplicaciones útiles en la manipulación de realaciones.

b) Si $A = [a_{ij}]$ y $B = [b_{ij}]$ son matrices booleanas $m \times n$, se define:

- Conjunción de matrices booleanas.
  $A \wedge B = C = [c_{ij}]$

$$c_{ij} = \begin{cases} 1, & \text{si } a_{ij} = b_{ij} = 1; \\ 0, & \text{si } a_{ij} = 0 \vee b_{ij} = 0. \end{cases}$$

- Disyunción de matrices booleanas.
  $A \vee B = D = [d_{ij}]$

$$d_{ij} = \begin{cases} 1, & \text{si } a_{ij} = 1 \vee b_{ij} = 1; \\ 0, & \text{si } a_{ij} = b_{ij} = 0. \end{cases}$$

c) Si $A = [a_{ij}]$ es una matriz booleana $m \times p$ y $B = [b_{ij}]$ es una matriz booleana $p \times n$, se define:

- Producto de matrices booleanas.
  $A \odot B = C = [c_{ij}]_{m \times n}$

$$c_{ij} = (a_{i1} \wedge b_{1j}) \vee (a_{i2} \wedge b_{2j}) \vee \ldots \vee (a_{ip} \wedge b_{pj})$$

Ejemplo 3.2.3. Dadas las matrices booleanas:

$$A = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

se tiene:

$$A \vee B = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

$$A \wedge B = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

$$A \odot B = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Observación 3.2.2. Podemos notar:

a) El cálculo de operaciones con matrices booleanas se puede efectuar fácilmente haciendo comparaciones en las posiciones adecuadas a cada operación.

b) Si A, B y C son matrices booleanas de tamaños compatibles, se tiene las propiedades:

1) Conmutatividad

$$A \vee B = B \vee A$$
$$A \wedge B = B \wedge A$$

2) Asociatividad

$$A \vee (B \vee C) = (A \vee B) \vee C$$
$$A \wedge (B \wedge C) = (A \wedge B) \wedge C$$

3) Distributividad

$$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$$
$$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$$

4) Asociatividad

$$A \odot (B \odot C) = (A \odot B) \odot C$$

5) $(A^T)^T = A$
6) $(A \vee B)^T = A^T \vee B^T$
7) $(A \wedge B)^T = A^T \wedge B^T$
8) $(A \odot B)^T = B^T \odot A^T$

c) *Una matriz es denominada simétrica si* $A^T = A$. En consecuencia, si A es simétrica, debe ser cuadrada. Es fácil demostrar que A es simétrica si sólo si $a_{ij} = a_{ji}$. Es decir, A es simétrica si y solamente si las entradas de A son simétricas con respecto de la diagonal principal de A.

Observación 3.2.3. Se debe enfatizar:

a) Sean R y S dos relaciones de A en B, entonces;

$$M_{R \cap S} = M_R \cap M_S$$
$$M_{R \cup S} = M_R \cup M_S$$
$$M_{R^{-1}} = (M_R)^T$$

b) Sea M una matriz booleana, el complemento $\overline{M}$ de M es la matriz que se obtiene al remplazar cada uno por cero y cada cero por uno.

entonces:

$$M_{\overline{R}} = \overline{M_R}$$

c) Sean A, B y C conjuntos finitos y sea R una relación de A en B y S una relación de B en C entonces:

$$M_{R \circ S} = M_R \odot M_S$$

Ejemplo 3.2.4. Sean $A = \{1, 2, 3\}$ y $B = \{a, b, c, d\}$. Sean R y S las relaciones de A en B cuyas matrices son:

$$M_R = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \; M_S = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

Entonces,

- $$M_{\overline{S}} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$
$$\overline{S} = \{(1,a),(1,d),(2,b),(2,c),(3,c),(3,d)\}$$

- $$M_{R\cup S} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$
$$R \cup S = \{(1,a),(1,b),(1,c),(1,d),(2,a),(2,d),(3,a),(3,b),(3,c)\}$$

- $$M_{R^{-1}} = (M_R{}^T) = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$
$$R^{-1} = \{(a,1),(a,3),(b,1),(b,3),(c,3),(d,1),(d,2)\}$$

Ejemplo 3.2.5. Sean las relaciones:

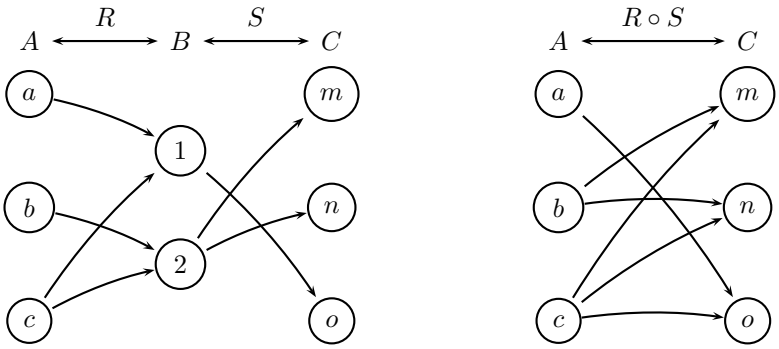


Figura 3.4: R, S y $R \circ S$

Observe que:

$$M_{R\circ S} = M_R \odot M_S$$

Además, es claro que:

$$R \circ S = \{(a,o),(b,m),(b,n),(c,m),(c,n),(c,o)\}$$

Teorema 3.1. *Sean R y S relaciones de A en B, entonces:*

a)

$$R \subseteq S \Rightarrow R^{-1} \subseteq S^{-1}$$

b)

$$R \subseteq S \Rightarrow \overline{S} \subseteq \overline{R}$$

c)

$$(R \cup S)^{-1} = R^{-1} \cup S^{-1}$$
$$(R \cap S)^{-1} = R^{-1} \cap S^{-1}$$

d)

$$\overline{(R \cup S)} = \overline{R} \cup \overline{S}$$
$$\overline{(R \cap S)} = \overline{R} \cap \overline{S}$$

Presentaremos la demostración de la parte 3.1-a, las demás se dejan para el lector.

Demostración.

$$\begin{aligned}(b,a) \in R^{-1} &\Longrightarrow (a,b) \in R \\ &\Longrightarrow (a,b) \in S \qquad \text{(por hipótesis)} \\ &\Longrightarrow (b,a) \in S^{-1}\end{aligned}$$

$\therefore R^{-1} \subseteq S^{-1}$ □

Teorema 3.2. *Sean A, B, C y D conjuntos finitos, R una relación de C en D, entonces:*

a) $(R \circ S) \circ T = R \circ (S \circ T)$

b) $(R \circ S)^{-1} = S^{-1} \circ R^{-1}$

Demostración.

a)

$$\begin{aligned}M_{(R\circ S)} \circ T &= M_{R\circ S} \odot M_T \\ &= (M_R \odot M_S) \odot M_T \\ &= M_R \odot (M_S \odot M_T) \\ &= M_R \odot M_{S\circ T} \\ &= M_{R\circ(S\circ T)}\end{aligned}$$

$\therefore (R \circ S) \circ T = R \circ (S \circ T)$

b)

$$\begin{aligned}M_{(R\circ S)^{-1}} &= (M_{R\circ S})^T \\ &= (M_R \odot M_S)^T \\ &= (M_S)^T \odot (M_R)^T \\ &= M_{S^{-1}} \odot M_{R^{-1}} \\ &= M_{S^{-1}\circ R^{-1}}\end{aligned}$$

$\therefore (R \circ S)^{-1} = S^{-1} \circ R^{-1}$ □

Observación 3.2.4. En casos futuros:

a) El teorema 3.2 es valido para cualquiera de los conjuntos A, B, C y D.

b) La operación de composición de relaciones no es conmutativa.

Definición 3.6. Sea R una relación en A. Una trayectoria de de longitud n de a a b, es una sucesión finita $\prod = a, x_1, x_2, \ldots, x_{n-1}, b$; que inicia en a y termina en b, tal que:

$$aRx_1, x_1Rx_2, \ldots, x_{n-1}Rb$$

Observación 3.2.5. Una trayectoria de longitud n involucra $n+1$ elementos, aunque no necesariamente distintos.

3.3. Trayectorias

Definición 3.7. Sea R una relación en A. Una trayectoria de de longitud n de a a b, es una sucesión finita $\prod = a, x_1, x_2, \ldots, x_{n-1}, b$; que inicia en a y termina en b, tal que:

$$aRx_1, x_1Rx_2, \ldots, x_{n-1}Rb$$

Observación 3.3.1. Una trayectoria de longitud n involucra $(n+1)$ elementos, aunque no necesariamente distintos.

Ejemplo 3.3.1. Con respecto a la relación asociada al siguiente digrafo:

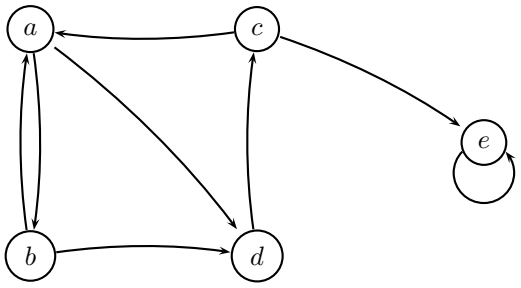


Figura 3.5: Digrafo de alguna relación R

Se tienen las trayectorias:

$$\begin{array}{lcll} \prod_1 & = & a,\ d,\ c,\ e & \text{trayectoria de longitud 3} \\ \prod_2 & = & d,\ c,\ e,\ e & \text{trayectoria de longitud 3} \\ \prod_3 & = & a,\ b,\ a & \text{trayectoria de longitud 2} \end{array}$$

Definición 3.8. Una trayectoria que se inicia y termina en el mismo vértice se le llama circuito.

Definición 3.9. Para cada entero positivo n, se define la relación R^n en A de acuerdo con:

$$aR^n b \iff \exists \text{ una trayectoria de longitud } n \text{ de } a \text{ a } b \text{ en } R$$

Definición 3.10. La relación de conectividad R^∞ en A, se define de acuerdo con:

$$aR^\infty b \iff \exists \text{ una trayectoria de } a \text{ a } b \text{ en } R$$

Ejemplo 3.3.2. Sea A el conjunto de todos los estudiantes de la USP y sea R la relación en A, definida con:

$$aRb \iff a \text{ y } b \text{ se conocen mutuamente.}$$

entonces:

aR^2b	significa que a y b tienen un conocido en común.
aR^nb	significa que a conoce algún x_1, que conoce a x_2, que conoce a $x_3, \ldots,$ que conoce a x_{n-1} que conoce a b.
aR^nb	significa que existe alguna cadena de conocidos la cual se inicia en a y termina en b.

¿Todo par de estudiantes de la USP están en la relación de conectividad?

Definición 3.11. Dadas las trayectorias $\prod_1 = a, x_1, x_2, \ldots, x_{n-1}, b$ y $\prod_2 = b, y_1, y_2, \ldots, y_{m-1}, c$ de longitudes n y m respectivamente, se define:

a) La composición de trayectorias $\prod_1 \circ \prod_2$ con:

$$\prod_1 \circ \prod_2 = a, x_1, x_2, \ldots, x_{n-1}, b, y_1, y_2, \ldots, y_{m-1}, c$$

b) La trayectoria inversa con:

$$\prod_1^{-1} = b, x_{n-1}, \ldots, x_2, x_1, a$$

Observación 3.3.2. El inverso de una trayectoria en R no necesariamente esta en R.

3.4. Propiedades de las relaciones

Definición 3.12. Diremos que una relación de R en A es:

a) Reflexiva $\iff aRa, \forall a \in A$

b) Irreflexiva $\iff a \not R a, \forall a \in A$

c) Transitiva $(\iff aRb \wedge bRc \implies aRc)$

d) Simétrica $\iff aRb \implies bRa)$

e) Asimétrica $\iff (aRb \implies b \not R a)$

f)

$$\begin{aligned}\text{Antisimetrica } \iff & (aRb \wedge bRa \iff a = b) \\ & (a \neq b \implies a \not R b \vee b \not R a)\end{aligned}$$

Observación 3.4.1. Es fácil notar:

a) La matriz de una relación reflexiva deberá tener unos en toda su diagonal principal, en cambio la matriz de una relación irreflexiva deberá tener ceros.

b) La relación reflexiva tiene un circuito de longitud 1 en cada vértice, en cambio la relación irreflexiva no tendrá circuitos de longitud 1.

c) El dominio y el rango de una relación reflexiva en A es el conjunto A.

d) La matriz $M_R = [m_{ij}]$ de una relación simétrica es simétrica, es decir: $[m_{ij}] = 1 \Longrightarrow [m_{ji}] = 1$.

e) Si R es una relación simétrica; entonces, si dos vértices están conectados por una arista, entonces deberán siempre estar conectados en ambas direcciones.

f) La matriz $M_R = [m_{ij}]$ de una relación asimétrica satisface que: $m_{ij} = 1 \Longrightarrow m_{ji} = 0$. Es importante observar que $m_{ii} \neq 1$ ¿por qué?

g) La matriz $M_R = [m_{ij}]$ de una relación antisimétrica satisface que: $i \neq j \Longrightarrow m_{ij} = 0 \vee m_{ji} = 0$. Es importante observar que no hay ninguna restricción para la diagonal principal de M_R.

h) la matriz $M_R = [m_{ij}]$ de una relación transitiva satisface que: $m_{ij} = 1 \wedge m_{jk} = 1 \Longrightarrow m_{ik} = 1$.

i) Una relación R en A es transitiva si y sólo si $R^n \subseteq R$ para todo entero positivo.

Ejemplo 3.4.1. Determine si la relación R en A es reflexiva, irreflexiva, simétrica, asimétrica o antisimétrica. Donde:
$A = \{1, 3, 6, 9, 12\}$, $aRb \Longleftrightarrow a|b$ "a divide a b"

Solución

$$M_R = \begin{array}{c} \\ 1 \\ 3 \\ 6 \\ 9 \\ 12 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 3 & 6 & 9 & 12 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

R es reflexiva

Demostración. M_R tiene unos en toda su diagonal principal. □

R no es irreflexiva
Contraejemplo

$m_{11} = 1$

R no es simétrica
Contraejemplo

$m_{12} = 1$ no implica que $m_{21} = 1$

R no es asimétrica
Contraejemplo

$m_{22} = 1$

R es antisimétrica

Demostración. Haciendo una inspección en M_R, se constata que:

$i \neq j \Longrightarrow m_{ij} = 0 \vee m_{ji} = 0$

□

3.5. Particiones

Una partición o conjunto cociente de un conjunto no vacío A es la colección de subconjuntos no vacios $\{A_1, A_2, \ldots, A_n\}$ de A tal que

a) $A_i \cap A_j = \emptyset \; \forall_i \neq 1$

b) $\bigcup_{i=1}^{n} A_i = A$

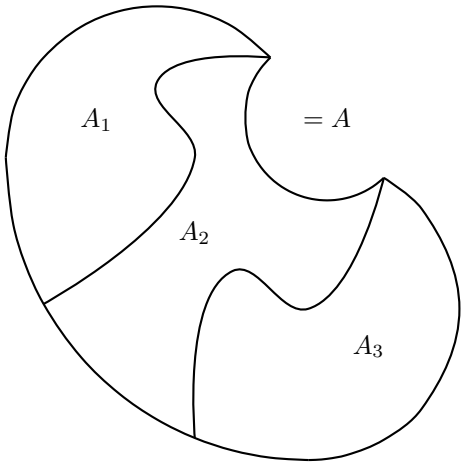


Figura 3.6: Classtering

Definición 3.13. Una relación R en A se denomina de equivalencia si y sólo si es reflexiva, simétrica y transitiva.

Definición 3.14. Sea R una relación de equivalencia en A. Si $a \in A$ entonces se define la clase de equivalencia de a como el conjunto

$$[a] = \{x \in A / aRx\}$$

Observación 3.5.1. Si R es una relación de equivalencia en A:

a) Diremos que a y b son equivalentes si y sólo si aRb, entonces en la clase de equivalencia $[a]$, están todos los elementos de A que son equivalentes a a.

b) Como en particular R es reflexiva, $a \in [a]$, entonces $[a] \neq \emptyset$.

c) $[a] = [b]$ si y sólo si aRb.

d) Si $a \neq [b]$ entonces $[a] \cap [b] = \emptyset$.

e) Llamaremos conjunto cociente de A por R a aquel conjunto cuyos elementos son las clases de equivalencia de A y lo denotaremos con $A|R$.

f) Por las observaciones anteriores el conjunto cociente $A|R$ es una partición de A y recíprocamente toda partición de un conjunto A define una relación de equivalencia R en A.

Ejemplo 3.5.1. Sea R una relación en $\mathbb{Z}$ definida por:

$$\begin{aligned} aRb &\iff a \equiv b \,, \forall a \in \mathbb{Z} \\ &\iff 2|(a-b) \end{aligned}$$

a) Demuestre que R es una relación de equivalencia.

b) Determine el conjunto cociente $A|B$.

Solución:

a) Una relación R es de equivalencia si R es reflexiva, simétrica y transitiva. demostremos entonces esas tres propiedades.

- R es reflexiva

$$\begin{aligned} 2|0 &\iff 2|(a-a) && ,\forall a \in \mathbb{Z} \\ &\iff a \equiv a(mod\ 2) && ,\forall a \in \mathbb{Z} \\ &\iff aRa && ,\forall a \in \mathbb{Z} \end{aligned}$$

- R es simétrica

$$\begin{aligned} aRb &\Rightarrow a \equiv b(mod\ 2) \\ &\Rightarrow 2|(a-b) \\ &\Rightarrow 2|-(a-b) \\ &\Rightarrow 2|(b-a) \\ &\Rightarrow b \equiv a(mod\ 2) \\ &\Rightarrow bRa \end{aligned}$$

- R es transitiva

$$\begin{aligned} aRb \text{ y } bRc &\Rightarrow a \equiv b(mod\ 2) \wedge b \equiv c(mod\ 2) \\ &\Rightarrow 2|(a-b) \wedge 2|(b-c) \\ &\Rightarrow \exists k_1, k_2 \in \mathbb{Z} \text{ tal que} \\ &\quad 2k_1 = a-b \wedge 2k_2 = b-c \\ &\Rightarrow \exists k_3 = k_1 + k_2 \in \mathbb{Z} \text{ tal que} \\ &\quad 2(k_1+k_2) = a-c \\ &\Rightarrow 2|(a-c) \\ &\Rightarrow a \equiv c(mod\ 2) \\ &\Rightarrow aRc \end{aligned}$$

$\therefore$ R es una relación de equivalencia.

b)

$[0] = \{\ldots, -4, -2, 0, 2, 4, \ldots\}$
$[1] = \{\ldots, -5, -3, -1, 1, 3, 5 \ldots\}$

Observe que:

$[0] = [2] = [4] = \ldots$
$[1] = [3] = [5] = \ldots$

entonces, $\mathbb{Z}|R = \{[0], [1]\}$ es decir, la relación $R = mod\ 2$ ha particionado a $\mathbb{Z}$ en enteros pares e impares.

Observación 3.5.2. Obsérvese que:

a) En forma análoga al ejemplo anterior se puede demostrar que:

$aRb \Longleftrightarrow a \equiv b(mod\ n)$

es una relación de equivalencia y denotaremos

$\mathbb{Z}|R = \mathbb{Z}|mod\ n = \mathbb{Z}_n$

b) De acuerdo con la observación 3.5.2-a se tiene que:

$\mathbb{Z}_2 = \{[0], [1]\}$
$\mathbb{Z}_3 = \{[0], [1], [2]\}$
$\mathbb{Z}_4 = \{[0], [1], [2], [3]\}$
$\vdots$

¿Determine [2] en $\mathbb{Z}_4$?

Teorema 3.3. *Sean R y S relaciones en A, entonces:*

a) Si R es reflexiva entonces R^{-1} es reflexiva.

b) R es reflexiva si y sólo si $\overline{R}$ es irreflexiva.

c) R es simétrica si y sólo si $R^{-1} = R$.

d) R es antisimétrica si y sólo si $R \cap R^{-1} \subseteq I$, donde $I = \{(a,a)/a \in A\}$ es la relación identidad.

e) R es asimétrica si y sólo si $R \cap R^{-1} = \emptyset$

f) Si R es simétrica entonces R^{-1} y $\overline{R}$ son simétricas.

g) Si R y S son reflexivas entonces $(R \cap S)$ y $(R \cup S)$ son reflexivas.

h) Si R y S son simétricas entonces $(R \cap S)$ y $(R \cup S)$ son simétricas.

i) $(R \cap S)^2 \subseteq R^2 \cap S^2$

j) Si R y S son transitivas entonces $R \cap S$ es transitiva.

k) Si R y S son relaciones de equivalencia entonces $(R \cap S)$ es una relación de equivalencia.

Demostración. Se hará las demostraciones de c, f e i

- Para c:

$$\begin{aligned} R \text{ es simétrica} &\iff M_R = (M_R)^T \\ &\iff M_R = M_{R^{-1}} \\ &\iff R = R^{-1} \end{aligned}$$

- Para f:

$$\begin{aligned} (R^{-1})^{-1} &= R \\ &= R^{-1} \qquad \text{por hipótesis y (c)} \end{aligned}$$

$\therefore R^{-1}$ es simétrica por (c).

- Para i:

$$\begin{aligned} (a,b) \in (R \cap S)^2 &\Rightarrow \exists \text{ una trayectoria de longitud 2 de } a \text{ a } b \text{ en } R \cup S \\ &\Rightarrow \text{ dicha trayectoria está en} R \text{ y está en } S \\ &\Rightarrow -(a,b) \in R^2 \wedge (a,b) \in S^2 \\ &\Rightarrow -(a,b) \in (R^2 \cap S^2) \end{aligned}$$

□

Las demás demostraciones se dejan como ejercicio para el lector.

3.6. Cerradura de las relaciones

Sea R una relación en A, la relacion R_c que resulta de añadir el menor número posible de pares ordenados a R tal que R_c posea una propiedad particular deseada, se llama la cerradura de R con respecto a la propiedad en cuestión.

Teorema 3.4. *Sea R una relación en A, entonces:*

a) La cerradura reflexiva de R es: $R_c = R \cup I$.

b) La cerradura simétrica de R es: $R_c = R \cup R^{-1}$.

c) La cerradura transitiva de R es: $R_c = R \cup R^{\infty}$

Demostraremos ahora el teorema 3.4:

Demostración. R^{∞} es una relación transitiva.

$$\begin{aligned} aR^{\infty}b \wedge bR^{\infty}c &\Rightarrow \exists \text{ trayectoria en } R\ \pi_1 \text{ de } a \text{ a } b \text{ y } \pi_2 \text{ de } b \text{ a } c \\ &\Rightarrow \exists \text{ una trayectoria } \pi = \pi_1 \circ \pi_2 \text{ de } a \text{ a } c \text{ en } R \\ &\Rightarrow aR^{\infty}c \end{aligned}$$

$R \subseteq R^{\infty}$

$R^{\infty} = R \cup R^2 \cup R^3 \cup \ldots$

R^{∞} es la relación transitiva más pequeña que contiene a R

Suponga que existe una relación transitiva S tal que $R \subseteq S$, entonces $S^n \subseteq S, \forall_n \geqq 1$ entonces $R^{\infty} \subseteq S^{\infty} \subseteq S$, por lo tanto $R^{\infty} \subseteq S$.

$\therefore$ la cerradura transitiva de R es $R_c = R^{\infty}$ □

Ejemplo 3.6.1. Hallar la clausura transitiva de la relación:

$$R = \{(a, c), (b, a), (b, c), (c, d)\}$$

Método 1:

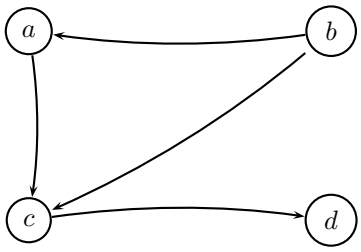


Figura 3.7: Digrafo de R

$$R_c = R^\infty$$
$$R_c = \{(a,c),(a,d),(b,a),(b,c),(b,d),(c,d)\}$$

Método 2:

$$R_c = R^\infty$$
$$R^\infty = R \cup R^2 \cup R^3 \cup R^4 \cup R^5 \cup \ldots$$

Luego:

$$M_R = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$
$$M_{R^2} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$
$$M_{R^3} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$
$$M_{R^4} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$
$$M_{R^5} = M_{R^4}$$
$$\vdots$$
$$\begin{aligned} M_{R_c} &= M_R \vee M_{R^2} \vee M_{R^3} \vee M_{R^4} \vee M_{R^5} \vee \ldots \\ &= M_R \vee M_{R^2} \vee M_{R^3} \vee (M_{R^4} \vee M_{R^4} \vee \ldots) \\ &= M_R \vee M_{R^2} \vee M_{R^3} \\ &= \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned}$$
$$R_c = \{(a,c),(a,d),(b,a),(b,c),(b,d),(c,d)\}$$

Teorema 3.5. *Sea A un conjunto finito con $|A| = n$ y sea R una relación en A. Entonces:*

$$R^\infty = R \cup R^2 \cup \ldots \cup R^n$$

Demostración.

$$
\begin{aligned}
(a,b) \in R^{\infty} &\iff \exists \text{ una trayectoria } \pi \text{ de } a \text{ a } b \text{ en } R \\
&\iff \pi = \pi_1 \circ \pi_2 \circ \ldots \pi_p \\
&\qquad \text{donde podrán algunas de ellas ser circuitos} \\
&\iff \exists \text{ una trayectoria } \pi_* \text{ de } a \text{ a } b \\
&\qquad \text{que es la de menor longitud (quitando los ciclos)} \\
&\iff \text{si } a \neq b \text{ la longitud de } \pi_* \\
&\qquad \text{es a lo más } n-1 \\
&\qquad \text{y si } a = b \text{ la longitud de } \pi_* \text{ es a lo más } n \\
&\iff (a,b) \in R^K, \text{ para algún } K = 1,2,3,\ldots,n \\
&\iff (a,b) \in (R \cup R)^2 \cup \ldots \cup R^n)
\end{aligned}
$$

$\therefore R^{\infty} = R \cup R^2 \cup \ldots \cup R^n$ □

Observación 3.6.1. El método gráfico no es práctico no sistemático para relaciones grandes. El método matricial puede usarse en general y es lo suficientemente sistemático para programarlo en computadora, pero es ineficiente y con matrices grandes su costo seria prohibitivo. A continuación presentaremos un algoritmo eficiente para calcular R^{∞}.

3.6.1. Algoritmo de Warshall

Sea R una relación en $A = \{a_1, a_2, \ldots, a_n\}$

a) En una trayectoria, cualquier vértice excepto el primero y el ultimo se llama vértice interior.

b) Para cada $K = 1, 2, \ldots, n$, se define W_K como la matriz que tiene un 1 en la posición i, j si y sólo si existe una trayectoria en R de a_i a a_j cuyos vértices interiores, provienen del conjunto $\{a_1, a_2, \ldots, a_k\}$

c) Se acepta $W_0 = M_R$

Ejemplo:

De acuerdo con la relación del ejemplo anterior.

$$
W_0 = M_R = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}
$$

$$W_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$W_3 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$W_4 = W_3$

Observación 3.6.2. Notemos:

a) En el ejemplo anterior $W_4 = M_{R^\infty}$, este resultado se ratifica en general, es decir. Si $|A| = n$ entonces $W_n = M_{R^\infty}$

b) Si $W_k = [w_{ij}^{(k)}]$ entonces

$$w_{ij}^{(k)} = w_{ij}^{(k-1)} \vee (w_{ik}^{(k-1)} \wedge w_{kj}^{(k-1)})$$

esta ecuación muestra que:

- Si $w_{ij}^{(k-1)} = 1$ entonces $w_{ij}^{(k)} = 1$, es decir que cada componente que es 1 en W_{k-1} se mantiene con 1 en la matriz W_k.
- Si $w_{ij}^{(k-1)} = 0$ entonces $w_{ij}^{(k)} = 1$, si y sólo si existen unos en las posiciones i, k y k, j de W_{k-1}.

c) La observación 3.6.2-b lleva al siguiente algoritmo para el calculo de W_k apartir de W_{k-1}.

Teorema 3.6. *Si R y S son relaciones de equivalencia en un conjunto A, entonces la relación de equivalencia más pequeña que contiene a R y a S es $(R \cup S)^\infty$.*

Demostración. Es claro que, R y S están contenidas en $(R \cup S)^\infty$.

$(R \cup S)^\infty$ es reflexiva

$$\begin{aligned} R \text{ y } S \text{ son de equivalencia} &\Rightarrow R \text{ y } S \text{ son reflexivas} \\ &\Rightarrow I \subseteq R \text{ e } I \subseteq S \\ &\Rightarrow I \subseteq (R \cup S) \subseteq (R \cup S)^\infty \\ &\Rightarrow I \subseteq (R \cup S)^\infty \\ &\Rightarrow (R \cup S)^\infty \text{ es reflexiva} \end{aligned}$$

$(R \cup S)^{\infty}$ es simétrica

$$
\begin{aligned}
R \text{ y } S \text{ son de equivalencia} &\Rightarrow R \text{ y } S \text{ son simétricas} \\
&\Rightarrow R^{-1} = R \text{ y } S^{-1} = S \\
&\Rightarrow (R \cup S)^{-1} = R^{-1} \cup S^{-1} = R \cup S \\
&\Rightarrow (R \cup S) \text{ es simétrica} \\
&\Rightarrow \text{todas las trayectorias en } (R \cup S) \\
&\quad \text{son en ambas direcciones} \\
&\Rightarrow (R \cup S)^{\infty} \text{ es simétrica}
\end{aligned}
$$

$(R \cup S)^{\infty}$ es transitiva y la mas pequeña que contiene a R y a S por ser la clausura transitiva de $(R \cup S)$. □

Observación 3.6.3. En ciencias de la computación se presenta a menudo la relación $R^* = R^{\infty} \cup I$ que la llamaremos relación de alcanzabilidad, pues aR^*b significa que b es alcanzable desde $a = b$ o si hay una trayectoria de a a b.

3.7. Lista de ejercicios

1. Sea el conjunto $A = \{1, 2, 3, 6, 9, 12, 18\}$ y sean las relaciones R y S en A definidas por: aRb si y sólo si aSb, si y sólo si a divide a b.
   Calcule:
   $a.M_{R \cup S} \quad b.M_{R \cup S^{-1}} \quad c.M_{\overline{R}} \quad d.S \cup R^2$

2. Determinar si R es una relación de equivalencia. Si su respuesta es afirmativa de una demostración y halle el conjunto cociente $A|R$ y si es negativa de un contraejemplo.

   a) $A = \mathbb{Z}^+ \times \mathbb{Z}^+$, $(a, b)R(c, d) \Leftrightarrow b = d$
   b) $A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $aRb \Leftrightarrow |a - b| \leq 2$

3. Determine si R es una relación de equivalencia. Si su respuesta es afirmativa de una demostración y halle el conjunto cociente $A|R$ y si es negativa de un contraejemplo.

   $S = \{1, 2, 3, 4, 5, 6\}, A = S \times S, (a, b)R(c, d) \Leftrightarrow a + b = c + d$

4. Determine si R es una relación de equivalencia. Si su respuesta es afirmativa de una demostración y halle el conjunto cociente $A|R$ y si es negativa de un contraejemplo.

   a) $A = \mathbb{Z}^+ \times \mathbb{Z}^+$, $(a, b)R(c, d) \Leftrightarrow a$ es múltiplo de c
   b) $S = \{1, 2, 3, 4, 5, 6\}$, $A = S \times S \Leftrightarrow ad = bc$

5. Sea $A = \{1, 2, 3, 4, 5\}$ y sean $M_R = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$ y $M_S = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}$.
   Las matrices de las relaciones R y S en A.

Calcule:

a) M_{R^2}, $M_{R \circ S}$

b) $M_{\overline{R}}$, $M_{R^{-1}}$

6. En los siguientes ejercicios, determine el dominio, rango, matriz y, cuando $A = B$, el digrafo de la relaciø'n R.

   - $A = \{1, 2, 3, 4, \}$, $B = \{1, 4, 6, 8, 9\}$; aRb si y sólo si $b = a^2$.
   - $A = \{1, 2, 3, 6, \} = B$; aRb si y sólo si a es múltiplo de b.
   - Sea $A = Z^+$, los enteros positivos, y R la relación definida por aRb si y s'lo si existe una k en Z^+ de modo que $a = A^k$ (k depende de a y b). ¿Cúales de los siguientes pares ordenados pertenecen a R?
     (a) (4, 16) (b) (1, 7) (c) (8, 2)
     (d) (3, 3) (e) (2, 8) (f) (2, 32)

7. De la relación R definida en A y su digrafo: Sea $A = \{a, b, c, d, e\}$ y

$$M_R = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

8. Sea $A = \{1, 2, 3, 4, 5, 6, 7\}$ y $R = \{(1, 2), (1, 4), (2, 3), (2, 5), (3, 6), (4, 7)\}$. Calcule las restricciones de R a B para el subconjunto de A dado.

   (a) $B = \{1, 2, 3, 4\}$

   (b) $B = \{2, 3, 4, 6\}$

9. Sea $A = \{1, 2, 3, 4\}$. Determine si la relación R cuya matriz M_R se da es reflexiva, irreflexiva, simétrica, antisimétrica o transitiva.

$$M_R = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

10. Determine si para los siguientes ejercicios la relación R en el conjunto A es reflexiva, irreflexiva, simétrica, asimétrica, antisimétrica o transitiva.

    - $A = \mathbb{Z}^+$; aRb si y sólo si $|a - b| \leq 2$.
    - $A = \mathbb{Z}^+$; aRb si y sólo si $|a - b| = 2$.
    - A = el conjunto de todos los pares ordenados de números reales; $(a, b)R(c, d)$ si y sólo si $a = c$.
    - A es el conjunto de todas las lineas que hay en el plano; $l_1 R l_2$ si y sólo si l_1 es paralela a l_2.

11. Demuestre que si una relación sobre un conjunto A es transitiva e irreflexiva, entonces es asimétrica.

12. Demuestre que si una relación R en un conjunto A es simétrica entonces la relación R^2 también es simétrica.

13. Sean $A = \{a, b, c, d\}$ y sean R y S las relaciones de equivalencia en A cuyas matrices se dan. Calcule la matriz de la relación de equivalencia más pequeña que contenga a R y a S, y determine la partición que este genera.

$$M_R = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$M_S = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

14. Sea $A = \{a, b, c, d, e\}$ y sean R y S las relaciones sobre A definidas por:

$$M_R = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$M_S = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Hallar la relación transitiva más pequeña que contenga a R y S

Capítulo 4

Estructuras de Orden

4.1. Conjuntos Parcialmente ordenados

Definición 4.1. Una relación R en A que sea reflexiva, transitiva y a su vez antisimétrica, es llamada de *orden parcial* y el par ordenado (A, R) es llamado conjunto parcialmente ordenado o poset (partially ordered set).

Ejemplo 4.1.1. Es sencillo comprobar que los siguientes son conjuntos parcialmente ordenados.

a) $(\geq, \leq)$

b) $(\mathfrak{P}(u), \subseteq)$

c) $(\mathbb{N}, |)$, donde $a|b$ significa a divide a b.

Observación 4.1.1. Podemos aseverar lo siguiente:

a) Es sencillo demostrar que si R es un orden parcial en A, entonces R^{-1} también lo es, y al conjunto parcialmente ordenado (A, R^{-1}) lo denominaremos el dual de (A, R) y recíprocamente.

b) Denotaremos un conjunto arbitrario parcialmente ordenado con $(A, \leq)$, pues un orden parcial arbitrario R tiene un significado análogo a $\leq$, y por lo tanto el orden parcial dual es $\geq$.

Definición 4.2. Sea $(A, \leq)$ un conjunto parcialmente ordenado. Los elementos a, b de A se denominan comparables si y solo si $a \leq b$ o $b \leq a$.

Ejemplo 4.1.2. Los números 2 y 8 son comparables en $(\mathbb{N}, /)$ pero 3 y 7 no lo son en $(\mathbb{N}, /)$. Así la palabra ″parcial‟en $(\mathbb{N}, /)$ significa que existen elementos que no son comparables.

Definición 4.3. Si cada par de elementos en $(A, \leq)$ son comparables, se dice que $(A, \leq)$ es un conjunto linealmente ordenado (totalmente) ordenado.

Teorema 4.1. *El orden parcial* $(A, \leq)$ *no tiene ciclos de longitud* $n \geq 2$.

Demostración. Supongamos que en $(A, \leq)$ existen ciclos en R de longitud $n \geq 2$, entonces existen:
$a_{i1}, a_{i2}, \ldots, a_{im}$ todos diferentes tales que:

$$a_{i1} \leq a_{i2} \leq \ldots \leq a_{in} \leq a_{i1}$$

Que es un ciclo de longitud m

$$\begin{array}{rl} a_{i1} \leq a_{im} & \text{pues } \leq \text{ es transitiva} \\ a_{im} \leq a_{i1} & \text{entonces } a_{i1} = a_{im} \end{array}$$

Porque $\leq$ es antisimétrica, lo cual es una contradicción al hecho de que $a_{i1}, a_{i2}, \ldots, a_{im}$ son diferentes, por lo tanto no existen ciclos de longitud $n \geq 2$. □

Observación 4.1.2. El teorema 4.1 nos permite realizar la siguiente conversión:

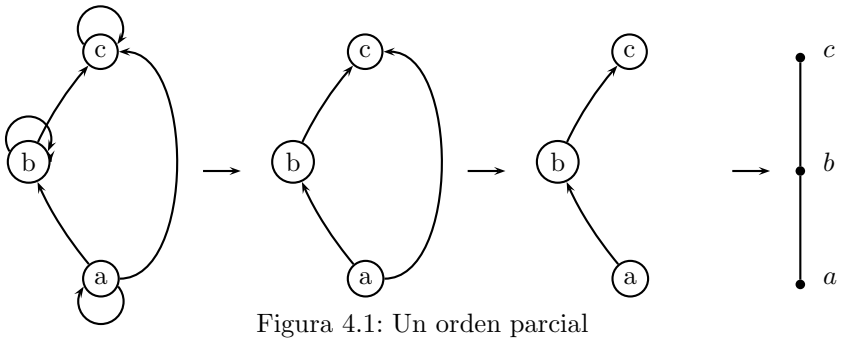


Figura 4.1: Un orden parcial

El gráfico 4.1 describe completamente al orden parcial y se denomina diagrama de *Hasse*

Ejemplo 4.1.3. El diagrama de Hasse de $(\mathfrak{P}(u), \subseteq)$ donde $u = \{a, b, c\}$ es:

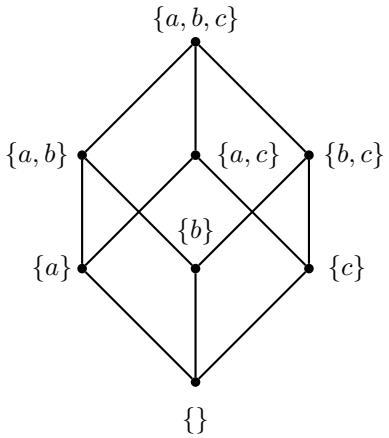


Figura 4.2: Representación mediante un diagrama de Hasse de $\mathfrak{P}(u)$

Observación 4.1.3. Se puede decir acerca de los diagramas de Hasse:

a) Es fácil verificar que los diagramas de Hasse, cumplen [2]:

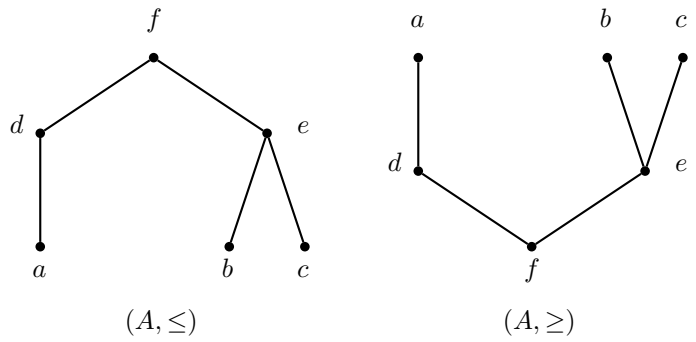


Figura 4.3: $(A, \leq)$ y su dual $(A, \geq)$

b) Los diagramas de Hasse de un conjunto linealmente ordenado tienen la forma:

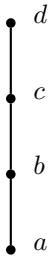


Figura 4.4: Conjunto linealmente ordenado

c) Sea un conjunto parcialmente ordenado. Algunas veces es necesario encontrar un orden lineal $\prec$ en A que sea simplemente una extensión de $\leq$ en el sentido de que si $a \leq b$ entonces $a \prec b$. El proceso de construcción del orden lineal $\prec$ se llama "clasificación topológica". Este problema surge cuando se tiene que introducir un conjunto parcialmente ordenado en una computadora. Los elementos de A deberán introducirse en un orden secuencial y la introducción será de la manera que el orden parcial se preserve.

Ejemplo 4.1.4. La figura 4.5 muestra $(A, \leq)$ y su clasificación topológica $(A, \prec)$, la cual no es única

Definición 4.4. Sea $(A, \leq)$ un conjunto parcialmente ordenado

[2]Sea ρ un conjunto parcialmente ordenado, se puede escribir el dual de ρ como ρ^d

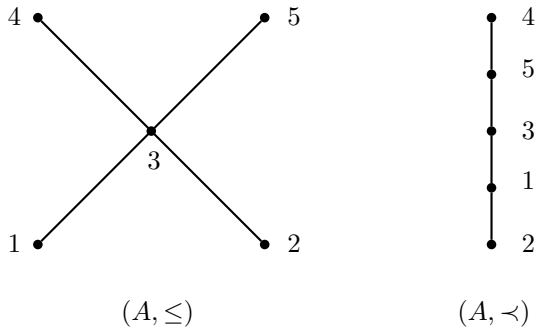


Figura 4.5: $(A, \leq)$ y su clasificación topológica $(A, \prec)$

a) Diremos que $a < b \iff a \leq b$ y $a \neq b$

b) $a \in A$ se denomina elemento maximal de $A \iff \nexists\ c \in A\ / a < c$

c) $b \in A$ se denomina elemento minimal de $A \iff \nexists\ c \in A\ / c < b$

Ejemplo 4.1.5. En el conjunto parcialmente ordenado

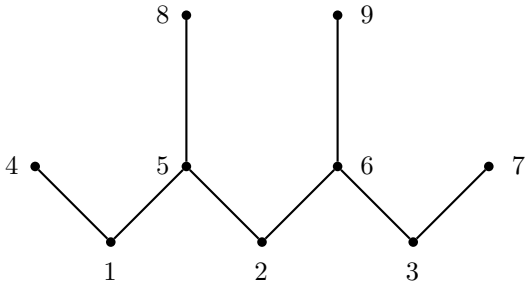


Figura 4.6: Un conjunto parcialmente ordenado

los elementos maximales son: $4, 8, 9, 7$ y los elementos minimales $1, 2, 3$

Observación 4.1.4. En todo comjunto parcialmente ordenado se cumple:

a) $(\geq, \leq)$ no tiene elementos maximales minimales

b) $a \in A$ es un elemento maximal (minimal) de $(A, \leq)$ si y soló si a es un elemento minimal (maximal) de $(A, \leq)$.

c) Si A es un conjunto finito entonces $(A, \leq)$ tiene al menos un elemento maximal y minimal.

d) Si $(A, \leq)$ es un conjunto parcialmente ordenado, $a \in A$, $B = A - \{a\}$ entonces $(B, \leq)$ es tambien un conjunto parcialmente ordenado.

e) El siguiente algoritmo, determina en un arreglo la clasificación topológica de un conjunto parcialmente ordenado.
Entrada:$(A, \leq)$ conjunto parcialmente ordenado
Salida:C "arreglo de clasificación topológica"

```
Procedure Clasificar (A)
1  i := 1
2  B := A
3  while B ≠ ∅
4        Escoja un elemento minimal a de B
5        C[i] := a
6        B := B − {a}
7  return (C)
8  end Clasificar
```

Definición 4.5. Sea $(A, \leq)$ un conjunto parcialmente ordenado

a) Elemento máximo o unidad de A, es un elemento a de A tal que $x \leq a$, $\forall x \in A$.

b) Elemento mínimo o cero de A es un elemento a de A tal que $a \leq x$, $\forall x \in A$.

Observación 4.1.5. No debemos de olvidar:

a) Un conjunto $(A, \leq)$ finito tiene a lo sumo un elemento máximo y uno mínimo.

b) El elemento máximo o mínimo de $(A, \leq)$ si existen, los denotaremos con 1 y 0.

Definición 4.6. Sea $(A, \leq)$ un conjunto parcialmente ordenado y sea $B \subseteq A$

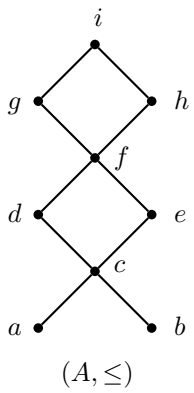
a) $a \in A$ es una cota superior de B si y solo si $b \leq a$, $\forall b \in B$

b) $a \in A$ es una cota inferior de B si y solo si $a \leq b$, $\forall b \in B$

c) La menor de las cotas superiores de B se denomina el supremo de B y lo denotaremos SupB.

d) La mayor de las cotas inferiores de B se denomina el ínfimo de B y lo denotaremos con InfB

Ejemplo 4.1.6. Para la figura 4.7 tenemos:



$(A, \leq)$

Figura 4.7: Conjunto parcialmente ordenado con 9 elementos

Elemento Máximo : i
Elemento Mínimo : no existe

Si $B = \{c, d, e\}$ entonces las cotas superiores de B son f, g, h, i y sus cotas inferiores son a, b, c $SupB = f$, $InfB = c$.

Observación

a) El supremo o ínfimo de B si existe es único.

b) Si $B = \{a, b\}$ entonces denotaremos:

$$SupB = a \vee b \quad , \quad InfB = a \wedge b$$

4.2. Látices

Definición 4.7. Un conjunto parcialmente ordenado $(L; \leq)$ tal que todo par de elementos de L tenga supremo e ínfimo se denomina retícula o látice. [1]

Ejemplo 4.2.1. Los siguientes son ejemplos de látices comunes y muy utilizados en computación

a) $(\mathfrak{P}(u), \subseteq)$ es una látice, pues: si $A, B \in \mathfrak{P}(u)$ entonces $A \vee B = A \cup B$, $A \wedge B = A \cap B$

b) Si W es un conjunto de todas las relaciones de equivalencia en A_1 $(W, \subseteq$) es una látice pues: si $R_1, R_2 \in W$ entonces $R_1 \vee R_2 = (R_1 \cup R_2)^{\infty}$, $R_1 \wedge R_2 = R_1 \cap R_2$

c) Sea n un entero positivo y sea D_n el conjunto de todos los divisores positivos de n $(D_n, /)$ es una látice.

[1] En este libro denominaremos "látice"a las retículas, usando el nombre original del ingles evitaremos diferentes confusiones.

Observación 4.2.1. Sean $(L_1, \leq_1)$ y $(L_2, \leq_2)$ látices. Si para $(a, b), (c, d) \in L = L_1 \times L_2$ definimos el orden parcial producto $\leq$ con:

$$(a, b) \leq (c, d) \Longleftrightarrow a \leq_1 c \ \wedge \ b \leq_2 d$$

tendremos que $(L, \leq)$ es tambien una látice pues:

$$(a, b) \vee (c, d) = (a \vee_1 c, b \vee_2 d) \qquad \text{y}$$
$$(a, b) \wedge (c, d) = (a \wedge_1 c, b \wedge_2 d)$$

Ejemplo 4.2.2. Dibujar el diagrama de Hasse de $L = D_2 \times D_8$

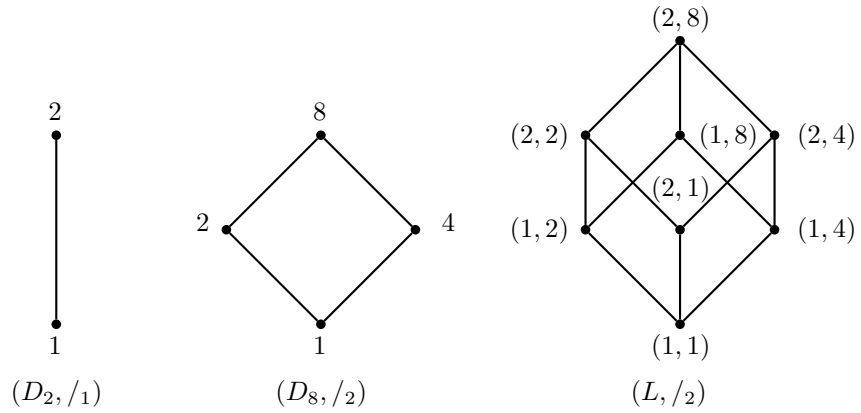


Figura 4.8: D_2, D_8 y $L = D_2 \times D_8$

Definición 4.8. Sea $(L, \leq)$ una látice. Un subconjunto no vacío S de L se denomina una sublátice si y sólo si $a \vee b \in S$, $a \wedge b \in S$, $\forall a, b \in S$

Ejemplo 4.2.3. Para la figura 4.9

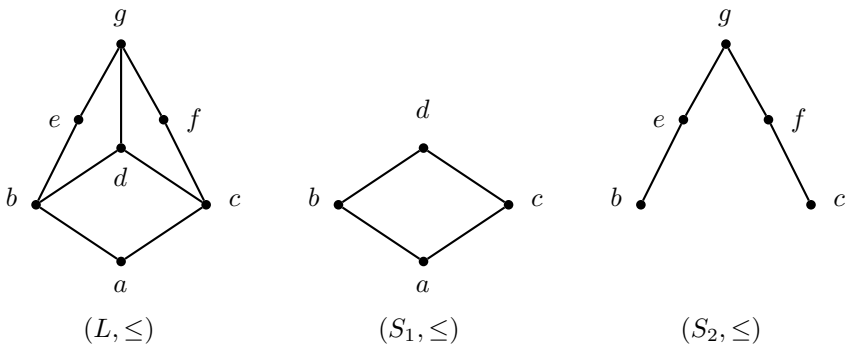


Figura 4.9: $(L, \leq)$ junto a S_1 y S_2

tenemos que $(S_1, \leq)$ es sublátice y $(S_2, \leq)$ no lo es.

Definición 4.9. Sean $(L, \leq)$ y $(L', \leq')$ látices.

a) Una función biyectiva $f : L \to L'$ si $\forall a, b \in L$ se verifica que:

1. $f(a \vee b) = f(a) \vee' f(b)$
2. $f(a \wedge b) = f(a) \wedge' f(b)$

b) Las látices $(L, \leq)$ y $(L', \leq)$ se dice que son isomorfas si y sólo si existe un isomorfismo $f : L \to L'$, lo que se denota con $L \cong L'$.

Observación 4.2.2. Podemos notar:

a) En una látice $(L, \leq)$, para $a, b \in L$, se tiene que:

$$a \leq b \iff a \vee b = b \quad \text{y} \quad a \wedge b = a$$

b) Sea $f : L \to L'$ un isomorfismo. Para $a, b \in L$, con $a \leq b$ se tiene que:$f(a) = f(a \wedge b) = f(a) \wedge f(b)$

$$\begin{aligned} f(a \vee b) &= f(a) \vee' f(b) \\ f(b) &= f(a) \vee' f(b) \qquad \text{por (a)} \end{aligned}$$

$\therefore \qquad f(a) \leq' f(b)$

Esto significa que dos látices son isomorfas si y sólo si el diagrama de Hasse de una de ellas se puede obtener a partir de la otra reetiquetando los vértices.

Ejemplo 4.2.4. ¿Serán isomorfas las látices $(D_{30}, /)$ y $(\mathfrak{P}(u), \subseteq)$?, donde $u = \{a, b, c\}$

Solución:
Analizando $(D_{30}, /)$ y $(\mathfrak{P}(u), \subseteq)$,

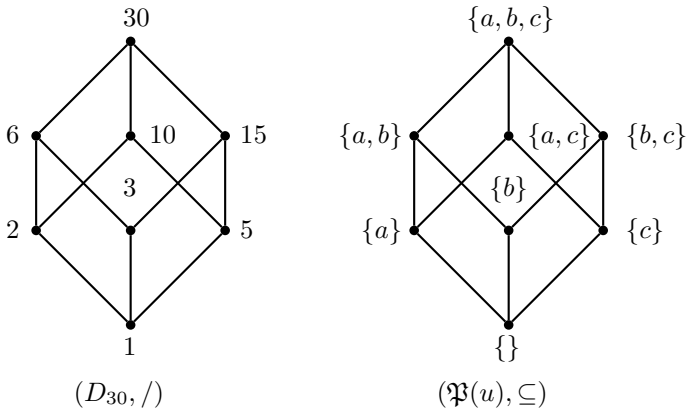


Figura 4.10: Digramas de *Hasse* para $(D_{30}, /)$ y $(\mathfrak{P}(u), \subseteq)$

es bastante claro que D_{30} es isomorfa con $\mathfrak{P}(u)$

Teorema 4.2. *Sea a, b, c elementos de una látice $(L, \leq)$*

a) $a \leq a \vee b \quad , \quad b \leq a \vee b$

b) $a \wedge b \leq a \quad , \quad a \wedge b \leq b$

c) Si $a \leq c$ y $b \leq c$ entonces $a \vee b \leq c$

d) Si $c \leq a$ y $c \leq b$ entonces $c \leq a \wedge b$

e) $a \vee a = a$
$a \wedge a = a$ *Idempotencia*

f) $a \vee b = b \vee a$
$a \wedge b = b \wedge a$ *Conmutativa*

g) $a \vee (b \vee c) = (a \vee b) \vee c$
$a \wedge (b \wedge c) = (a \wedge b) \wedge c$ *Asociativa*

h) $a \vee (a \wedge b) = a$
$a \wedge (a \vee b) = a$ *Absorción*

A continuación veremos la demostración de algunas de las afirmaciones: La siguiente demostración, pertenece al teorema 3.2 (h). Las demás pruebas se deja para el lector.

Demostración. Como $a \wedge b \leq a$ y $a \leq a$ entonces $a \vee (a \wedge b) \leq a$, además es claro que $a \leq a \vee (a \wedge b)$, luego como $\leq$ es antisimétrica $a = (a \wedge b)$. □

Definición 4.10. Una látice $(L, \leq)$ es acotada si existen sus elementos máximo y mínimo

Teorema 4.3. *Si $(L, \leq)$ es una látice finita entonces L es acotada*

Demostración. Sea $L = \{a_1, a_2, \ldots, a_n\}$ y como $\vee$ $\wedge$ son asociativas entonces:

$$I = a_1 \vee a_2 \vee \ldots \vee a_n$$
$$0 = a_1 \wedge a_2 \wedge \ldots \wedge a_n$$

□

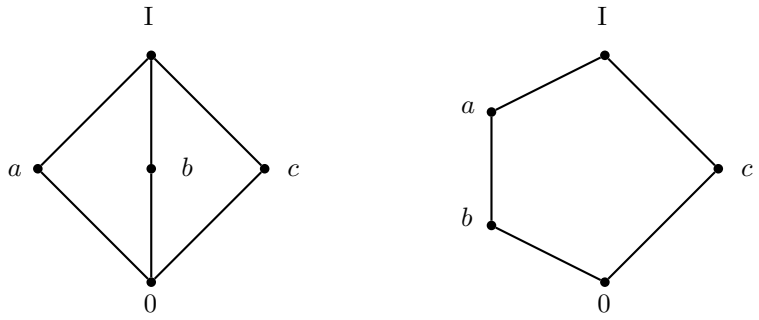
Definición 4.11. Una látice $(L, \leq)$ se dice que es distributiva, si para toda $a, b, c \in L$ se tiene que:

a) $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$

b) $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$

Ejemplo 4.2.5. La látice $(\mathfrak{P}(u), \subseteq)$ es distributiva ¿Por qué?

Ejemplo 4.2.6. Las siguientes látices en la figura 4.11 no son distributivas:



$(L_1, \leq)$ $(L_2, \leq)$

Figura 4.11: Látices no distributivas

Observación 4.2.3. Una látice no es distributiva si contiene una sublátice que sea isomorfa con una de las látices L_1 o L_2 del ejemplo anterior.

Definición 4.12. Sea $(L, \leq)$ una látice acotada. Un elemento $a' \in L$ se llamará el complemento de $a \in L$ si y sólo si $a \vee a' = I$ y $a \wedge a' = 0$.

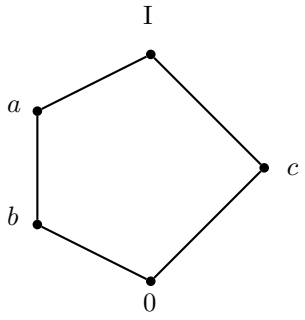
Observación 4.2.4. Se puede notar lo siguiente:

a) En una látice arbitraria los elementos máximos y mínimos cumplen:
$0 \vee 1 = 1 \quad$ y $\quad 0 \wedge 1 = 0$
entonces $0' = 1 \quad$ y $\quad 1' = 0$

b) La látice $(\mathfrak{P}(u), \subseteq)$ es acotada, $1 = u$ y $0 = \emptyset$ entonces para $A \in (\mathfrak{P}(u))$ tenemos:$A' = \bar{A}$ pues:

$$A \cup \bar{A} = U$$
$$A \cap \bar{A} = \emptyset$$

Ejemplo 4.2.7. En la figura 4.12 tenemos:



$(L_2, \leq)$

Figura 4.12: Látice no distributiva

$$\begin{array}{lcl} \text{I'} & = & 0 \\ a' & = & c \\ b' & = & c \\ c' & = & a \\ c' & = & b \\ 0' & = & \text{I} \end{array}$$

Teorema 4.4. *Si un elemento de una látice $(L, \leq)$ acotada y distributiva, tiene complemento entonces este es único.*[1]

Demostración. Supongamos que a' y a'' son los complementos de $a \in L$, entonces:

$$\begin{array}{ll} a \vee a' = 1 & a \vee a'' = 1 \\ a \wedge a' = 0 & a \wedge a'' = 0 \end{array}$$

luego:

$$\begin{array}{ll} a' = a' \vee 0 & a'' = a'' \vee 0 \\ \quad = a' \vee (a \wedge a'') & \quad = a'' \vee (a \wedge a') \\ \quad = (a' \vee a) \wedge (a' \vee a'') & \quad = (a'' \vee a) \wedge (a'' \vee a') \\ \quad = 1 \wedge (a' \vee a'') & \quad = 1 \wedge (a'' \vee a') \\ \quad = a' \vee a'' \ldots\ldots \boxed{1} & \quad = a'' \vee a' \ldots\ldots \boxed{2} \end{array}$$

Entonces de 1 y 2 $a' = a''$ lo cual es una contradicción por consiguiente el complemento de a es único. □

Definición 4.13. Una látice se dice complementada si es acotada y cada elemento tiene al menos un complemento.

Ejemplo 4.2.8. ¿D_{20} es complementada?

[1] Para poder establecer la veracidad del teorema que acabamos de afirmar se debe de realizar una demostración la cual deberá constatar la unicidad

4.3. Álgebras Booleanas

Definición 4.14. Un álgebra booleana es una látice complementada, distributiva con al menos dos elementos

Ejemplo 4.3.1. $(\mathfrak{P}(u), \subseteq)$ es un álgebra booleana.

Definición 4.15. Dos álgebras booleanas son isomorfas, si lo son como látices (Definición 3.9) y si además se cumple que:

$$f(a') = (f(a))'$$

Si u es un conjunto finito, con $|u| = n$, se puede demostrar que un álgebra booleana arbitraria es isomorfa con el álgebra booleana $(\mathfrak{P}(u), \subseteq)$, para algún u especialmente escogido, es decir que las álgebras booleanas tienen 2^n elementos.

Teorema 4.5. *Si $L, \leq$ es un álgebra booleana, y si $a, b \in L$ entonces se cumple:*

a) $(a')' = a$ *Involución*

b) $\left.\begin{array}{l}(a \vee b)' = a' \wedge b' \\ (a \wedge b)' = a' \vee b'\end{array}\right\}$ *Leyes de Morgan*

Primero veremos la demostración de la parte a y luego la parte b.

Demostración. Puesto que:

$$\begin{aligned} a \vee a' &= 1 \\ a \wedge a' &= 0 \end{aligned}$$

y haciendo una correcta lectura, se tiene que $a'' = a$ □

Demostración. Puesto que:

$$\begin{aligned} (a \vee b) \vee (a' \wedge b') &= 1 \\ (a \vee b) \wedge (a' \wedge b') &= 0 \end{aligned}$$

Entonces $(a \vee b)' = a' \wedge b'$
También

$$\begin{aligned} (a \wedge b) \vee (a' \vee b') &= 1 \\ (a \wedge b) \wedge (a' \vee b') &= 0 \end{aligned}$$

□

Ejemplo 4.3.2. Los diagramas de Hasse de la figura 4.13 representan álgebras booleanas de especial interés.

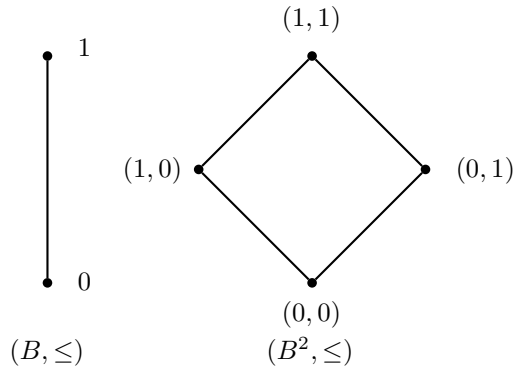


Figura 4.13: Álgebras Booleanas

Observación 4.3.1. Mencionaremos algunas observaciones importantes sobre álgebras booleanas:

a) En álgebras booleanas análogas a las del ejemplo anterior denotaremos el supremo $\vee$ con $+$, el ínfimo $\wedge$ con $\cdot$ y el complemento con $^-$, esto debido a sus aplicaciones en la electrónica.

b) Si $a = (a_1, a_2, \ldots, a_n)$ y $b = (b_1, b_2, \ldots, b_n)$ son elementos del álgebra booleana $(B^n, \leq)$, es claro que:

$$\begin{aligned} a + b &= (a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n) \\ a \cdot b &= (a_1 \cdot b_1, a_2 \cdot b_2, \ldots, a_n \cdot b_n) \\ 0_{B^n} &= (0, 0, \ldots, 0) \\ 1_{B^n} &= (1, 1, \ldots, 1) \end{aligned}$$

c) Una variable x se denomina variable booleana si x sólo toma valores en B^n

4.4. Redes Lógicas

En 1983, el matemático americano Claude Shanon percibió, las semejanzas entre la lógica proposicional y la lógica de circuitos, y pensó que el álgebra booleana B^n podría ayudar a sistematizar este nuevo dominio de la electrónica. Imaginemos que los voltajes eléctricos transmitidos por los hilos puedan asumir dos valores, alto o bajo, que representaremos por 1 y 0 respectivamente. Las fluctuaciones de voltaje entre dos valores serán ignorados, de forma que forjamos un fenómeno discreto binario a partir de un fenómeno originalmente análogo.

Supongamos ahora que llaves mecánicas pueden ser usadas a fin de que se obtiene la señal 1 cuando la llave esta cerrada y se obtiene la señal 0 con la llave abierta.

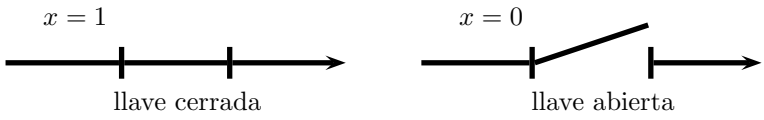


Figura 4.14: Sistema de llaves mecánicas

Entonces, combinando en paralelo los hilos x_1 y x_2 controlados con dos llaves determinan una red eléctrica con interruptores que llamaremos circuitos de conmutación.

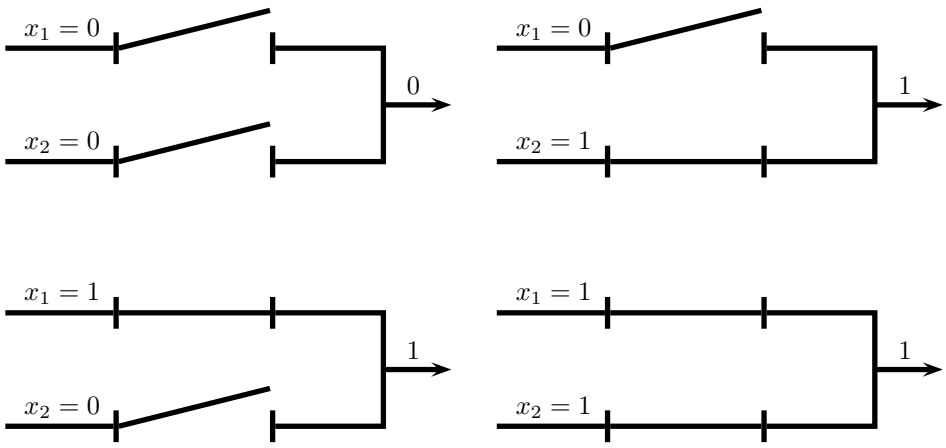


Figura 4.15: Circuitos de conmutación

El resumen del comportamiento de este circuito de conmutación se puede abstraer en la siguiente tabla:

x_1	x_2	$x_1 + x_2$
0	0	0
0	1	1
1	0	1
1	1	1

Por tanto, podemos pensar de este circuito de conmutación, como un dispositivo electrónico que realiza la operación booleana $+$. Otros circuitos de conmutación realizan las operaciones booleanas $\bullet$ y $^-$. Por ahora, vamos a ignorar

detalles de la implementación de estos dispositivos; es suficiente decir que la tecnología avanzo de llaves mecánicas a tubos de vacío y de estos a transistores para circuitos integrados. Simplemente, aceptamos que los circuitos e conmutación se pueden costruír mediante dispositivos de estado solido, llamados compuertas, que puedan intercambiar los niveles de voltaje (bits)[2], las compuertas básicas son:

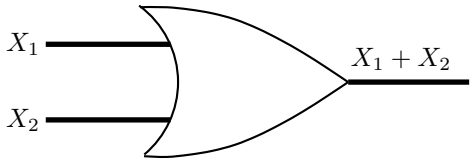


Figura 4.16: Compuerta Or

x_1	x_2	$x_1 + x_2$
0	0	0
0	1	1
1	0	1
1	1	1

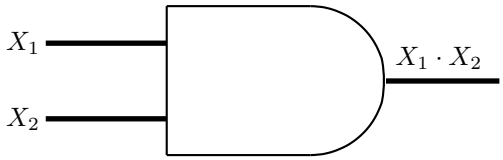


Figura 4.17: Compuerta And

x_1	x_2	$x_1 \cdot x_2$
0	0	0
0	1	0
1	0	0
1	1	1

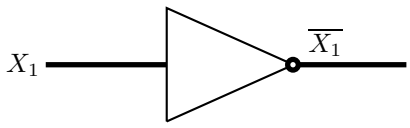


Figura 4.18: Compuerta Not

x_1	$\overline{x_1}$
0	1
1	0

[2]bit:Es la abreviatura del ingles binary digit o dígito binario en español.

Definición 4.16. Una espresión o polinomio booleano en $n-$variables $x_1, x_2, \ldots, x_n$ se define como sigue:

a) $x_1, x_2, \ldots, x_n$ son todos polinomios booleanos.

b) Los símbolos 0 y 1 son polinomios booleanos.

c) Si $p(x_1, x_2, \ldots, x_n)$ y $q(x_1, x_2, \ldots, x_n)$ son polinomios booleanos, entonces tambien lo son: $p + q$, $p \cdot q$ y $\bar{p}$

Ejemplo 4.4.1. Los siguientes son polinomios booleanos:

$$p(x, y, z) = (x + y) \cdot z$$
$$q(x, y, z) = x + \bar{y} \cdot z + x \cdot (y + 1)$$

Observación 4.4.1. En la lógica proposicional, los conectivos lógicos: conjunción, disyunción y complemento son ejemplos de polinomios booleanos, entonces las proposiciones compuestas también son ejemplos de polinomios booleanos

Definición 4.17. Una función booleana es una función $f : B^n \longrightarrow B$ para algún entero positivo n.

Ejemplo 4.4.2. La tabla de verdad para la operación booleana + describe la función booleana:
$f : B^n \longrightarrow B$ con $f(0,0) = 0$, $f(0,1) = 1$, $f(1,0) = 1$, $f(1,1) = 1$.

Observación 4.4.2. Cualquier polinomio booleano define una única función booleana de la misma forma que los polinomios booleanos:

$p(x, y) = x + y$
$q(x, y) = x \cdot y$
$r(x) = \bar{x}$

Ejemplo 4.4.3. El polinomio booleano $p(x, y, z) = x\bar{y}+z$ define una función booleana $f : B^3 \longrightarrow B$ que se acostumbra describir por medio de una tabla que llamaremos tabla de verdad.

x	y	z	$f(x, y, z)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Observación 4.4.3. Combinando las compuertas AND, OR y NOT, podemos construir una red lógica que representa determinado polinomio booleano.

Ejemplo 4.4.4. El polinomio booleano $p(x, y, z) = \overline{(xy + z)}z$ se puede representar por la red lógica:

Observación 4.4.4. Las redes lógicas son llamadas también redes combinatorias. Ellas tienen diversos aspectos que debemos percibir. Primero, las lineas de entrada o de salida no se cruzan, excepto al pasar por las compuertas. Por lo tanto las lineas pueden ser divididas a fin de establecer entradas para más de una compuerta. No hay lazos donde la salida de un componente sirva como entrada para este mismo componente. Finalmente, la salida de una red es una función instantánea de la entrada, es decir que cada entrada produce una única salida. Esta situación hasta ahora establece que:

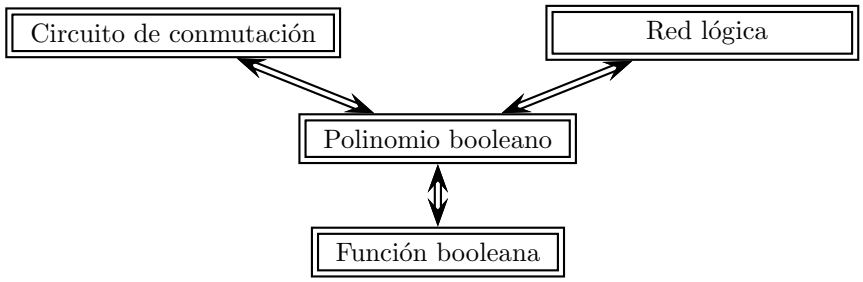


Figura 4.19: Diagrama para producir una función booleana

¿Podemos ir de una función booleana a un polinomio booleano?. Un algoritmo para resolver este problema es explicado en el siguiente ejemplo.

Ejemplo 4.4.5. Sea la función booleana $f : B^3 \longrightarrow B$ descrita por la tabla de verdad.

x	y	z	$f(x,y,z)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

1. Como existen cuatro filas en la tabla $(2, 5, 6, 8)$ para las cuales f es 1, la forma básica de nuestro polinomio booleano tendrá la forma de una suma de 4 términos.

   $$(\ ) \quad + \quad (\ ) \quad + \quad (\ ) \quad + \quad (\ )$$

   de modo que el primer término asume el valor de 1 para los valores de entrada de la fila 2, el segundo termino asume el valor de 1 para los valores de entrada de la fila 5, y así en adelante. Por lo tanto, el polinomio booleano tiene valor 1 para esas entradas y ninguna otra, precisamente lo que deseamos. (Otras entradas hacen que los términos y por lo tanto toda la suma, tenga el valor 0).

2. Cada término en la suma será un producto de la forma $l_1 \cdot l_2 \cdot l_3$, donde l_1, l_2, l_3 son llamados literales y l_1 es x o $\bar{x}$, l_2 es y o $\bar{y}$ y l_3 es z o $\bar{z}$.

3. Si el valor de la entrada x en la fila que estuviéramos analizando fuera 1, entonces $l_1 = x$ y si el valor de la entrada fuera 0, usamos $l_1 = \bar{x}$, haciendo lo mismo con las entradas y y z tendremos que $l_1 \cdot l_2 \cdot l_3$ sea 1 para las filas $(2, 5, 6, 8)$ y 0 para las demás filas.
   Por lo tanto, se tiene que:

$$f(x, y, z) = \bar{x}\bar{y}z + x\bar{y}\bar{z} + x\bar{y}z + xyz$$

Definición 4.18. Sea $f : B^n \longrightarrow B$ una función booleana.

a) Llamaremos conjunción fundamental o mintérmino a la conjunción de literales $l_1, l_2 \ldots, l_n$

b) Llamaremos Forma Normal disyuntiva (fnd) a una representación de f como una suma de mintérminos.

Observación 4.4.5. Es claro que el procedimiento descrito en el ejemplo anterior siempre nos llevará a la Forma Normal Disyuntiva de función booleana arbitraria.

Definición 4.19. Dos polinomios booleanos son equivalentes si representan la misma función booleana.

Definición 4.20. Si $f, g : B^n \longrightarrow B$ funciones booleanas, entonces se define las funciones booleanas $\bar{f}, f+g, f \cdot g$: $B^n \longrightarrow B$, complemento suma y producto respectivamente con:

a) $\bar{f}(b) = \overline{f(b)}$, $b \in B^n$

b) $(f + g)(b) = f(b) + g(b)$, $b \in B^n$

c) $(f \cdot g)(b) = f(b) \cdot g(b)$, $b \in B^n$

Observación 4.4.6. A continuación presentamos diez leyes que son consecuencias importantes de las definiciones de las propiedades de las álgebras boleanas.

1. $\bar{\bar{f}} = f$ "Ley del doble complemento "

2. $\begin{array}{rcl} \overline{f+g} & = & \bar{f} \cdot \bar{g} \\ \overline{f \cdot g} & = & \bar{f} + \bar{g} \end{array}$ "Leyes de Morgan"

3. $\begin{array}{rcl} f+g & = & g+f \\ f \cdot g & = & g \cdot f \end{array}$ "Leyes conmutativas"

4. $\begin{array}{rcl} f+(g+h) & = & (f+g)+h \\ f \cdot (g \cdot h) & = & (f \cdot g) \cdot h \end{array}$ "Leyes asociativas"

5. $\begin{array}{rcl} f+g \cdot h & = & (f+g) \cdot (f+h) \\ f \cdot (g+h) & = & f \cdot g + f \cdot h \end{array}$ "Leyes distributivas"

6. $\begin{array}{rcl} f+f & = & f \\ f \cdot f & = & f \end{array}$ "Leyes de indempotencia "

7. $\begin{array}{rcl} f+0 & = & f \\ f \cdot 1 & = & f \end{array}$ "Leyes de identidad"

8. $\begin{array}{rcl} f+\bar{f} & = & 1 \\ f\cdot\bar{f} & = & 0 \end{array}$ "Leyes de inversos"

9. $\begin{array}{rcl} f+1 & = & 1 \\ f\cdot 0 & = & 0 \end{array}$ "Leyes de dominación"

10. $\begin{array}{rcl} f+f\cdot g & = & f \\ f\cdot(f+g) & = & f \end{array}$ "Leyes de absorción"

Ejemplo 4.4.6. Hallar la fnd de la función $f : B^4 \longrightarrow B$ definida por $f(w,x,y,z) = wx\bar{y} + wy\bar{z} + xy$

Solución: Examinando cada término:

$$\begin{aligned} wx\bar{y} &= wx\bar{y}\cdot 1 && \text{"Ley de identidad "} \\ &= wx\bar{y}\cdot(z+\bar{z}) && \text{"Ley de inversos"} \\ &= wx\bar{y}z + wx\bar{y}\bar{z} && \text{"Ley distributiva"} \end{aligned}$$

$$\begin{aligned} wy\bar{z} &= w\cdot 1\cdot y\bar{z} && \text{"Ley de identidad "} \\ &= w\cdot(x+\bar{x})\cdot y\cdot\bar{z} && \text{"Ley de inversos"} \\ &= wxy\bar{z} + w\bar{x}y\bar{z} && \text{"Ley distributiva"} \end{aligned}$$

$$\begin{aligned} xy &= 1\cdot x\cdot y\cdot 1 && \text{"Ley de identidad "} \\ &= (w+\bar{w})\cdot x\cdot y\cdot(z+\bar{z}) && \text{"Ley de inversos"} \\ &= wxyz + wxy\bar{z} + \bar{w}xyz + \bar{w}xy\bar{z} && \text{"Ley distributiva"} \end{aligned}$$

entonces

$$f(w,x,y,z) = wx\bar{y}z + wx\bar{y}\bar{z} + \underline{wxy\bar{z}} + w\bar{x}y\bar{z} + wxyz + \underline{wxy\bar{z}} + \bar{w}xyz + \bar{w}xy\bar{z} \quad (5)$$

por la ley de idempotencia (6) se tiene la fnd,

$$f(w,x,y,z) = wx\bar{y}z + wx\bar{y}\bar{z} + wxy\bar{z} + w\bar{z}y\bar{z} + wxyz + \bar{w}xyz + \bar{w}xy\bar{z}$$

Observación 4.4.7. En las álgebras booleanas:

a) Si acordamos enumerar los elementos del álgebra boolena B^n veremos que los valores $x_1, x_2, \ldots, x_n$ en cualquier fila determina un número binario, como puede apreciarse en la siguiente tabla para $f : B^3 \longrightarrow B$.

x	y	z	fila	binario	$f(x,y,z)$
0	0	0	1	000 (=0)	0
0	0	1	2	001 (=1)	1
0	1	0	3	010 (=2)	0
0	1	1	4	011 (=3)	0
1	0	0	5	100 (=4)	1
1	0	1	6	101 (=5)	1
1	1	0	7	110 (=6)	0
1	1	1	8	111 (=7)	1

b) Debido a la representación en binario de B^n, la fnd de una función booleana se puede representar en forma compacta, por ejemplo, de acuerdo con f dado en la tabla anterior; tenemos:

$$f = \sum m(1, 4, 5, 7)$$

Ejemplo 4.4.7. Consideremos el ejemplo anterior

$f(w, x, y, z) = wx\bar{y} + wy\bar{z} + xy$

$f(w, x, y, z) = wx\bar{y}z + wx\overline{yz} + wxy\bar{z} + w\bar{x}y\bar{z} + wxyz + \bar{w}xyz + \bar{w}xy\bar{z}$ (fnd)

A falta de la tabla de verdad observe:

$wx\bar{y}z \iff 1101 \iff 13$
$wx\overline{yz} \iff 1100 \iff 12$
$wxy\bar{z} \iff 1110 \iff 14$
$w\bar{x}y\bar{z} \iff 1010 \iff 10$
$wxyz \iff 1111 \iff 15$
$\bar{w}xyz \iff 0111 \iff 7$
$\bar{w}xy\bar{z} \iff 0110 \iff 6$

de donde, $f = \sum m(6, 7, 10, 12, 13, 14, 15)$

Observación 4.4.8. En las álgebras booleanas se cumple:

a) El dual de una afirmación relacionada con expresiones booleanas, se obtiene reemplazando 0 por 1, 1 por 0, + por · y · por +.

b) El dual de un teorema relativo a álgebras booleanas también es un teorema.

c) El dual de una conjunción fundamental $l_1 \cdot l_2 \ldots \cdot l_n$ es una disyunción fundamental $l_1 + l_2 + \ldots + l_n$ de literales que llamaremos maxtérminos.

d) El dual de fnd es la Forma Normal Conjuntiva (fnc), que es un producto de maxtérminos.

Ejemplo 4.4.8. Hallar la fnc para $f : B^3 \longrightarrow B$ definida por la tabla

x	y	z	$f(x, y, z)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

solución:
Aplicando el concepto de dualidad, se tiene que:

$$f(x, y, z) = (x + y + z)(x + \bar{y} + z)(\bar{x} + \bar{y} + z) \qquad (fnc)$$

en forma compacta se escribe:

$$f = \prod M(0, 2, 6)$$

donde M significa maxtérminos.

Ejemplo 4.4.9. Hallar la fnc para $f : B^4 \longrightarrow B$ definida por:

$$f(w, x, y, z) = (w + x + y)(x + \bar{y} + z)(w + \bar{y})$$

Solución:

$$\begin{aligned} w + x + y &= w + x + y + 0 && \text{“Ley de identidad ”} \\ &= w + x + y + z \cdot \bar{z} && \text{“Ley de inversos”} \\ &= (w + x + y + z)(w + x + y + \bar{z}) && \text{“Ley distributiva”} \end{aligned}$$

$$\begin{aligned} x + \bar{y} + z &= 0 + x + \bar{y} + z && \text{“Ley de identidad ”} \\ &= w\bar{w} + x + \bar{y} + z && \text{“Ley de inversos”} \\ &= (w + x + \bar{y} + z)(\bar{w} + x + \bar{y} + z) && \text{“Ley distributiva”} \end{aligned}$$

$$\begin{aligned} w + \bar{y} &= w + 0 + \bar{y} + 0 && \text{por (7)} \\ &= w + x\bar{x} + \bar{y} + z\bar{z} && \text{por (8)} \\ &= (w + x + \bar{y})(w + \bar{x} + \bar{y}) + z\bar{z} && \text{por (5)} \\ &= (w + x + \bar{y} + z\bar{z})(w + \bar{x} + \bar{y} + z\bar{z}) && \text{por (5)} \\ &= (w + x + \bar{y} + z)(w + x + \bar{y} + \bar{z})(w + \bar{x} + \bar{y} + z)(w + \bar{x} + \bar{y} + \bar{z}) && \text{por (5)} \end{aligned}$$

de donde

$$\begin{aligned} f(w, x, y, z) = (w + x + y + z)(w + z + y + \bar{z})\underline{(w + x + \bar{y} + z)} \\ (\bar{w} + x + \bar{y} + z)\underline{(w + x + \bar{y} + z)}(w + x + \bar{y} + \bar{z}) \\ (w + \bar{x} + \bar{y} + z)(w + \bar{x} + \bar{y} + \bar{z}) \end{aligned}$$

por la ley de idempotencia (6) se tiene la fnc,

$$\begin{aligned} f(w, x, y, z) = (w + x + y + z)(w + z + y + \bar{z}) \\ (\bar{w} + x + \bar{y} + z)(w + x + \bar{y} + z) \\ (w + x + \bar{y} + \bar{z})(w + \bar{x} + \bar{y} + z)(w + \bar{x} + \bar{y} + \bar{z}) \end{aligned}$$

que en forma compacta se escribe como:

$$f = \prod M(0, 1, 2, 3, 6, 7, 10)$$

4.5. Aplicaciones

4.5.1. Arreglos lógicos programables

Un circuito integrado, es una red lógica que representa una o más funciones booleanas, como si algunas compuertas lógicas estuviesen debidamente interligadas en un paquete. Estos circuitos integrados son entonces combinados a fin

de obtener el resultado deseado. Los chips cuadrados de silicio en los cuales los circuitos integrados son constrídos, no pueden tener más de 6 milímetros de lado, sin embargo pueden contener el equivalente a medio millón de transistores que implementan funciones booleanas.

En vez de proyectar un chip dedicado a implementar algunas funciones booleanas en particular, podemos usar un PLA *(Programable Logic Array)*. El PLA es un chip que contiene un arreglo de compuertas AND, OR, NOT. Una vez que tengamos determinada la forma de suma de productos de las funciones booleanas deseadas, los elementos correspondientes al PLA son activados. A pesar de que este chip no es muy eficiente, y ser práctico apenas para circuitos lógicos de pequeña escala, el PLA puede ser producido en masa y apenas es una pequeña cantidad de tiempo y dinero.

Ejemplo 4.5.1. La siguiente figura muestra un PLA para tres entradas x, y, z. Existen cuatro lineas de salida de forma que podemos programar cuatro funciones en este PLA. Cuando el PLA es programado, la linea horizontal que sirve de entrada para una compuerta AND, recibe ciertas entradas y la compuerta AND realiza el producto de tales entradas. La linea vertical que entra en una compuerta OR permitirá, cuando esté programada, que la compuerta OR realice la suma de ciertos productos.

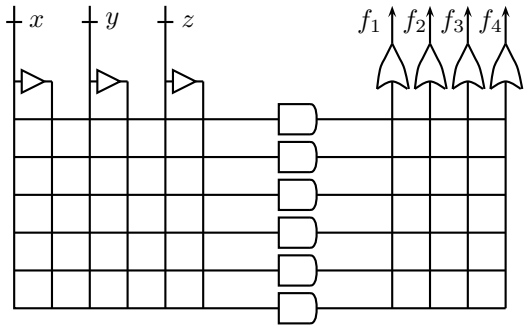


Figura 4.20: PLA

La figura 4.21 muestra el mismo PLA programada a fin de producir las funciones booleanas.

$f_1(x,y,z) = xyz + x\bar{y}z + x\bar{y}\bar{z} + \bar{x}\bar{y}z$
$f_2(x,y,z) = xyz + x\bar{y}z + x\bar{y}\bar{z} + \bar{x}\bar{y}z + \bar{x}\bar{y}\bar{z}$
$f_3(x,y,z) = x\bar{y}z + \bar{x}\bar{y}z + \bar{x}yz$

Los pequeños círculos sombreados representan puntos activados.

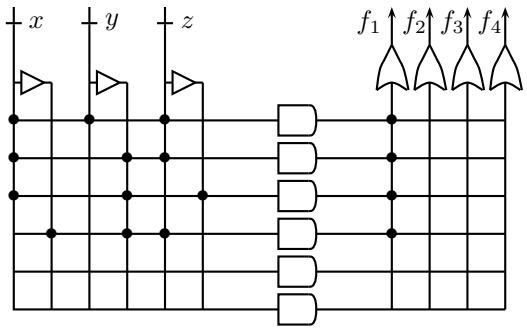


Figura 4.21: PLA activado

4.5.2. Un circuito para sumar números binarios

Para diseñar una red lógica que sume números binarios, operación básica que un computador precisa saber realizar, primero diseñemos la suma de dos bits (dígitos binarios), que dan como resultado las funciones booleanas resumidas en las siguientes tablas.

x	y	Suma en binario
0	0	0
0	1	1
1	0	1
1	1	10

x	y	Suma S
0	0	0
0	1	1
1	0	1
1	1	0

x	y	Acarreo C
0	0	0
0	1	0
1	0	0
1	1	1

La fnd de la suma y el acarreo C es,

$$\begin{aligned} S &= \bar{x}y + x\bar{y} \\ &= x \oplus y \quad , \qquad \text{recuerde que } \oplus \text{ denota la o exclusiva} \\ &= (x+y)\overline{xy} \\ C &= xy \end{aligned}$$

La figura 4.22 es una red de compuertas con dos salidas. Este dispositivo, por motivos que se aclaran más adelante, es llamado semisumador (SS).

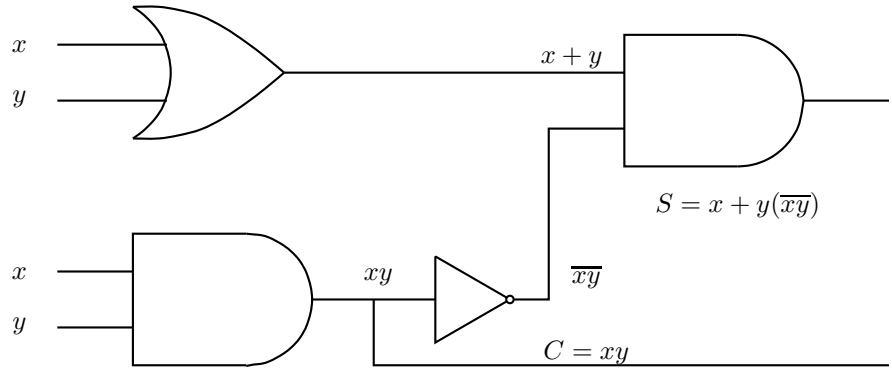


Figura 4.22: Semisumador (SS)

Si usamos dos semisumadores y una compuerta OR, construímos el sumador completo (SC) que se muestra en la figura 4.23

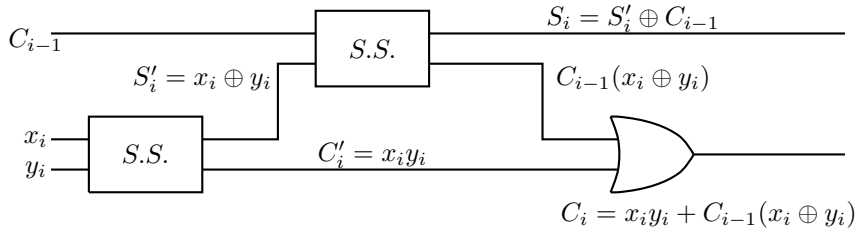


Figura 4.23: Sumador completo *SC*

Si $x = x_n \cdot x_{n-1} \cdot \ldots \cdot x_2 \cdot x_1 \cdot x_0$ y $y = y_n \cdot y_{n-1} \cdot \ldots \cdot y_2 \cdot y_1 \cdot y_0$, consideramos el proceso de sumar los bits x_i y y_i para determinar $x + y$. En este caso, c_{i-1} es el acarreo de la suma de x_{i-1} y $yi - 1$ y un posible acarreo c_{i-2}. La entrada c_{i-1}, junto con las entradas x_i y y_i, produce la suma S_i y el acarreo C_i como se muestra en la figura.

Por último, en la figura 4.24 , se combinan dos sumadores completos y un semisumador para obtener la suma de dos números binarios $x_2\ x_1\ x_0$ y $y_2\ y_1\ y_0$ cuya suma es $c_2\ s_2\ s_1\ s_0$

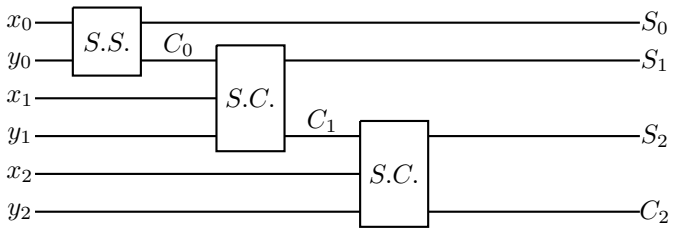


Figura 4.24: Sumador de números binarios

Si $c_2 = 1$, ha ocurrido un desbordamiento; si $c_2 = 0$, no hay desbordamiento

Para sumar dos números binarios arbitrarios, precisamente de una memoria interna primitiva, dispositivos que se llaman circuitos secuenciales.

4.5.3. Otros componentes lógicos

Antes de considerar algunas alternativas a las compuertas AND, OR, NOT, debemos dar una definición precisa de compuerta

Definición 4.21. Una compuerta es una función booleana de B^n en B.

Es claro que estamos interesados en aquellas compuertas que permitan construir circuitos lógicos arbitrarios.

Definición 4.22. Un conjunto de compuertas $\{g_1, g_2, \ldots, g_k\}$ es funcionalmente completo si, dado cualquier entero positivo n y una función $f : B^n \longrightarrow B$, es posible construir un circuito lógico, que calcule f utilizando sólo las compuertas $g_1, g_2, \ldots, g_k$.

Ejemplo 4.5.2. Los conjuntos de compuertas {AND, OR, NOT}, {AND, NOT},y {OR, NOT} son funcionalmente completos.

Definición 4.23. Una compuerta NAND que recibe las entradas x y y donde x y y son bits, y produce la salida denotada con $x \uparrow y$ y definida por:

x	y	$x \uparrow y$
0	0	1
0	1	1
1	0	1
1	1	0

Una compuerta NAND se representa con la figura 4.25:

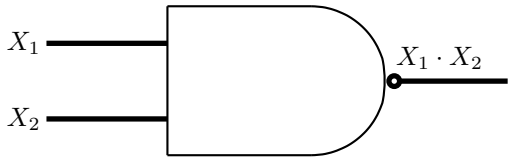


Figura 4.25: Compuerta NAND

A su vez tenemos la compuerta NOR que produce:

x	y	$x \downarrow y$
0	0	1
0	1	0
1	0	0
1	1	0

y se representa con la Figura 4.26:

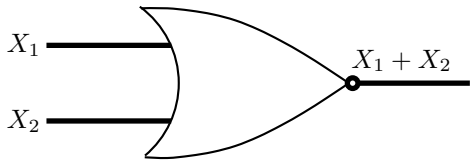


Figura 4.26: Compuerta NOR

Teorema 4.6. *El conjunto $\{NAND\}$ es un conjunto funcionalmente completo.*

Demostración. Primero observamos que, $x \uparrow y = \overline{xy}$, entonces

$$\begin{aligned} \bar{x} &= \overline{xx} \\ &= x \uparrow x \\ x + y &= \bar{\bar{x}} + \bar{\bar{y}} \\ &= \overline{\bar{x}\bar{y}} \\ &= \bar{x} \uparrow \bar{y} \\ &= (x \uparrow x) \uparrow (y \uparrow y) \end{aligned}$$

Estas ecuaciones muestran que OR y NOT se pueden escribir en términos de NAND, entonces por el ejemplo anterior {NAND} tambien lo es funcionalmente completo. □

Observación 4.5.1. Muchos de los circuitos básicos utilizados actualmente en las computadoras digitales se construyen mediante compuertas NAND

4.5.4. Construyendo funciones booleanas

Sabemos como escribir una expresión booleana y construir una red a partir de una función booleana. En general, la función booleana propiamente dicha precisa, antes ser deducida a partir de la descripción de algún problema en cuestión

Ejemplo 4.5.3. En una empresa de cosméticos de entrega por correo, un dispositivo de control automático es usado para supervisar el empaquetamiento de las remesas.

La empresa vende: loción, perfume, maquillaje y esmalte de uñas. Como bonificación, es adicionado un lápiz labial en todas las encomiendas que incluyan loción y perfume o que incluyan maquillaje y esmalte de uñas con loción o perfume.

¿Cómo podemos diseñar una red lógica que controle cuando el lápiz labial debe ser incluido?

Solución:

Las entradas de la red representan los cuatro productos que pueden ser ordenados. Vamos a llamarlos:

$w =$ Loción
$x =$ Perfume
$y =$ Maquillaje
$z =$ Esmalte de Uñas

Estas variables serán 1 cuando este producto es parte de la encomienda, y es 0 sino lo es. La salida de la red de ser 1 si el lapiz labial debe ser incluido en la encomienda , ó 0, en caso contrario. La tabla de verdad del circuito es:

w	x	y	z	$f(w,x,y,z)$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Ejemplo 4.5.4. Una ley del corredor es controlada por dos interruptores, uno en cada extremo. La tabla de verdad que permite que la luz sea prendida y apagada en ambos interruptores es:

x	y	$f(x,y)$
0	0	0
0	1	1
1	0	1
1	1	1

4.6. Minimización

Naturalmente, la equivalencia de polinomios booleanos es una relación de equivalencia en el conjunto de todos los polinomios booleanos de $n-$variables. Cada clase de equivalencia está asociada a una función booleana distinta.

Si deseamos diseñar una red lógica para una función booleana, lo ideal es que tengamos un procedimiento que obtenga el polinomio booleano más simple de la clase de equivalencia. Lo que llamamos simple dependerá de la tecnología que estamos empleando en la construcción de la red, que tipos de componentes lógicos serán usados, etc. De cualquier forma, probablemente deseemos minimizar el número total de conexiones que precisan ser hechas y el número total de componentes usados. (Al discutir procedimientos de minimización, recordemos que otros facotres pueden influenciar el costo del circuito. Si la red fuera construida apenas una vez, el tiempo utilizado para la minimización posiblemente sea mayor el de la cosntrucción de la red. Sin embargo si la red fuera construida en gran escala el costo de tiempo de la minimización se justifica.)

Ejemplo 4.6.1. En el conjunto funcionalmente completo {AND,OR,NOT} implementar una red lógica para la función booleana $f : B^4 \to B$ por:

$$f = \sum m(4, 5, 7, 8, 9, 11)$$

que está asociada a los polinomios booleanos,

$$p_1(w, x, y, z) = \bar{w}x\bar{y}\bar{z} + \bar{w}x\bar{y}z + \bar{w}xyz + w\bar{x}\bar{y}\bar{z} + w\bar{x}\bar{y}z + w\bar{x}yz$$

al aplicar propiedades del álgebra booleana se tiene que:

$$\begin{aligned}
p_1(w, x, y, z) &= \bar{w}x \cdot (\bar{y}\bar{z} + \bar{y}z + yz) + w\bar{x}(\bar{y}\bar{z} + \bar{y}z + yz) \\
&= \bar{w}x(\bar{y}(\bar{z} + z) + yz) + w\bar{x}(\bar{y}(\bar{z} + z) + yz) \\
&= \bar{w}x(\bar{y} \cdot 1 + yz) + w\bar{x}(\bar{y} \cdot 1 + yz) \\
&= \bar{w}x(\bar{y} + yz) + w\bar{x}(\bar{y} + yz) \\
&= \bar{w}x((\bar{y} + y)(\bar{y} + z)) + w\bar{x}((\bar{y} + y)(\bar{y} + z)) \\
&= \bar{w}x(1 \cdot (\bar{y} + z)) + w\bar{x}(1 \cdot (\bar{y} + z)) \\
p_2(w, x, y, z) &= \bar{w}x \cdot (\bar{y} + z) + w\bar{x} \cdot (\bar{y} + z) \\
p_3(w, x, y, z) &= \bar{w}x\bar{y} + \bar{w}xz + w\bar{x}\bar{y} + w\bar{x}z
\end{aligned}$$

Es claro que los polinomios booleanos:

$p_1(w, x, y, z)$	"fnd de f"
$p_2(w, x, y, z)$	
$p_3(w, x, y, z)$	"suma minimal de productos"

son equivalentes.

Las siguientes figuras muestran la red lógica para cada uno de estos polinomios booleanos respectivamente. A partir de ahora consideramos una entrada de la forma $\bar{w}$ como una entrada exacta, que no ha pasado por compuerta alguna, en vez de considerarla como el resultado de introducir w y pasarla por un inversor.

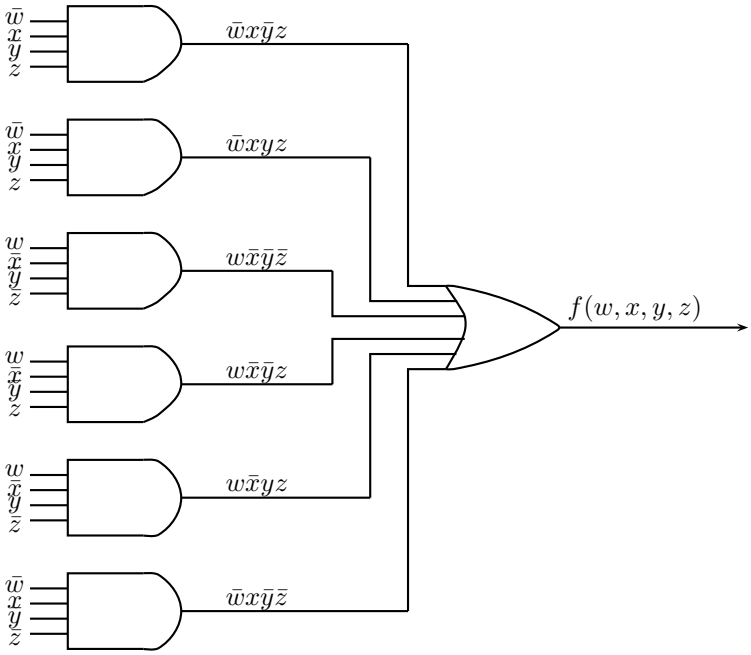


Figura 4.27: Red lógica de 2 niveles

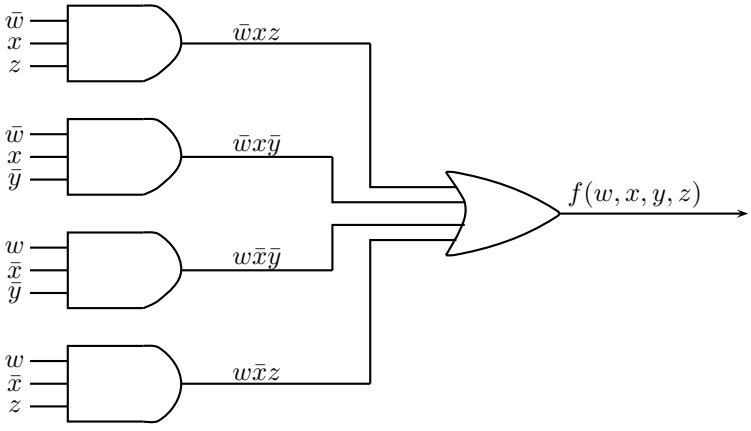


Figura 4.28: Red lógica de 2 niveles

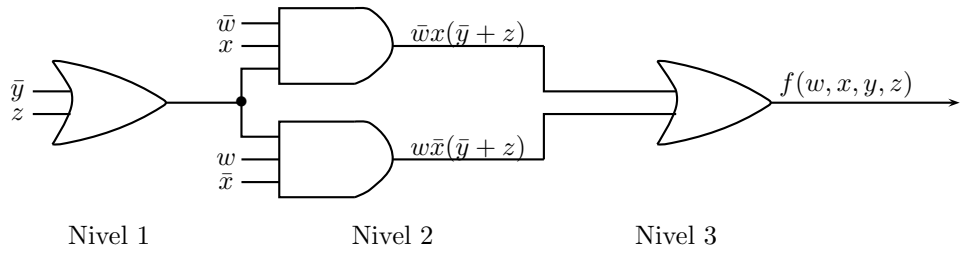


Figura 4.29: Red lógica de 3 niveles

Observación 4.6.1. La red lógica de la figura 4.28 sólo tiene cuatro compuertas lógicas, mientras que la primera 4.27 tiene 7 y 4.29 presenta 5 en consecuencia, podríamos pensar que la segunda red 4.28 es mejor respecto a la minimización de costos, puesto que cada compuerta adicional incrementa el costo de producción.

En el estudio de las redes de compuertas, las salidas se consideran funciones instantáneas de la entrada. Esto significa que cada nivel de compuertas añade un retraso en el desarrollo de la función. En el equipo digital de alta velocidad queremos minimizar el retraso, por lo que optamos por más velocidad. Esta necesidad de maximizar la velocidad nos hace desear la representación de una función booleana como una suma minimal de productos, en nuestro ejemplo la última red lógica .

Ya vimos en el ejemplo la simplificación de $f(w,x,y,z)$ a través del uso de las propiedades del álgebra booleana, en tanto, no tenemos un procedimiento que aplicar, apenas resolver el problema en forma individual. Lo que deseamos ahora es un procedimiento sistemático que podamos usar sin la necesidad de ser tan ingeniosos.

En esta sección discutiremos dos procedimientos para la reducción de una fnd a una suma minimal de productos. Por tanto, podemos minimizar dentro del contexto de forma de suma minimal de productos.

Observación 4.6.2. En el ejemplo anterior

a) La equivalencia,

$$\begin{aligned} xy + \bar{x}y &= (x+\bar{x})y \\ &= 1 \cdot y \\ &= y \end{aligned}$$

significa, que

$$\bar{w}x\bar{y}\bar{z} + \bar{w}x\bar{y}z = \bar{w}x\bar{y}$$

Por lo tanto, cuando tenemos una suma de dos productos que difieren apenas en una literal, podemos eliminar esta literal. Sin embargo, la emphfnd

de una función booleana, puede tener muchos términos, lo que dificulta la localización de dos conjunciones que difieren apenas en una literal. Para ayudar en esta búsqueda, podemos usar el mapa de *Karnaugh*, que es una representación visual de la función booleana que permite la identificación rápida de las localidades de una fnd que sólo difieran en una sola literal y que llamaremos adyacentes.

b) Podremos usar el mapa de *Karnaugh* con funciones booleanas de modo que $f : B^n \to B$, si n no es demasiado grande. Se ilustrará con ejemplos el caso que $n = 2, 3, 4$. Si n es grande o si se desea utilizar un algoritmo programable, son preferibles otras técnicas.

Ejemplo 4.6.2. En la figura siguiente, la función booleana $f : B^2 \to B$ es representada por su tabla de verdad y mapa de *Karnaugh*.

X \ Y	0	1
0		1
1		1

x	y	$f(x,y)$
0	0	0
0	1	1
1	0	0
1	1	1

En una rápida inspección, podemos identificar los 1s en las localidades adyacentes ($\bar{x}y + xy = y$) que marcaremos en el mapa de *Karnaugh*. Entonces la suma minimal de productos es:

$$f(x, y) = y$$

Ejemplo 4.6.3. En el mapa de *Karnaugh*,

X \ Y	0	1
0		1
1	1	1

se encuentras marcadas las adyacencias ($x\bar{y} + xy = x, \bar{x}y + xy = y$) que determinan que:

$$f(x, y) = x + y$$

Ejemplo 4.6.4. Hallar la suma minimal de productos para las funciones booleanas:

a) $f = \sum m(0, 2, 4, 7)$

b) $f = \sum m(0, 2, 4, 6)$

c) $f = \sum m(0, 1, 2, 3)$

d) $f = \sum m(1, 2, 3, 5, 6, 7)$

Solución:

a) "El 1 de xyz no es adyacente a ningún otro 1 en el mapa, por lo que diremos que está aislado"

X \ YZ	00	01	11	10
0	1			1
1	1		1	

$$f(x, y, z) = \bar{x}\bar{z} + \bar{y}\bar{z} + xyz$$

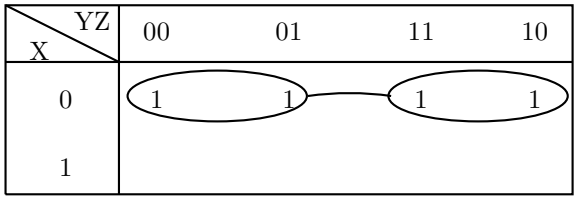
b) $f = \sum m(0, 2, 4, 6)$

$$\bar{x}\bar{y}\bar{z} + x\bar{y}\bar{z} = \bar{y}\bar{z}$$
$$\bar{x}y\bar{z} + xy\bar{z} = \bar{z}$$
$$\bar{y}\bar{z} + y\bar{z} = \bar{z}$$

X \ YZ	00	01	11	10
0	1			1
1	1			1

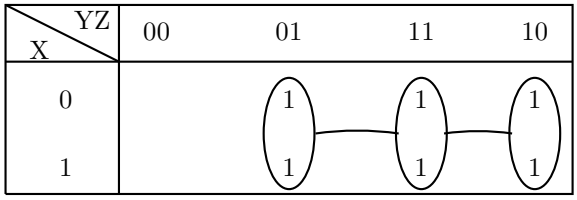
$$f(x, y, z) = \bar{z}$$

c) $f = \sum m(0,1,2,3)$



$$f(x,y,z) = \bar{x}$$

d) $f = \sum m(1,2,3,5,6,7)$ "Hay que buscar todas las adyacencias"



$$f(x,y,z) = y + z$$

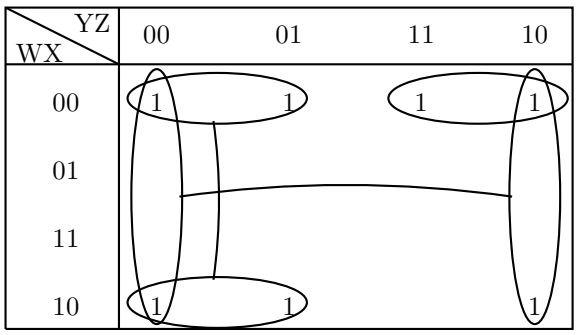
Ejemplo 4.6.5. Hallar la suma minimal de productos para las funciones booleanas:

a) $f = \sum m(0,1,2,3,8,9,10)$

b) $f = \sum m(3,4,5,7,9,13,14,15)$

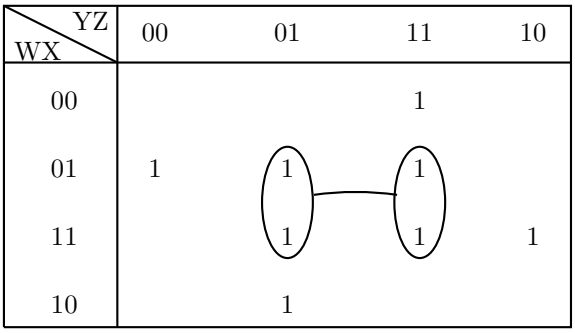
Solución

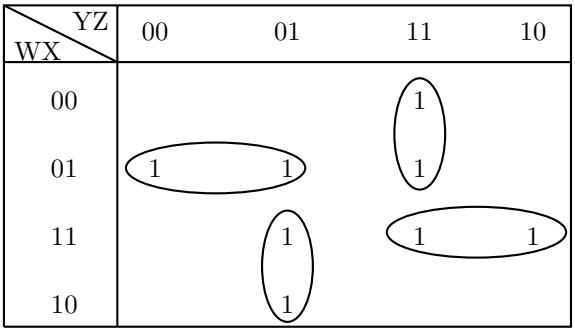
a) $f = \sum m(0,1,2,3,8,9,10)$



$$f(w,x,y,z) = \bar{w}\bar{x} + \overline{xy} + w\bar{x}$$

b) Hay una forma correcta y otra incorrecta de usar un mapa de *Karnaugh.*





En el mapa 1 combinamos un bloque de cuatro unos para formar el término xz. Pero si nos fijamos en los otros unos , hacemos lo que aparece en el mapa 2. Así el resultado del mapa 2 produce f como un suma de cuatro términos (cada uno con tres literales), mientras que el mapa 1 añade un termino adicional (innecesario) xz. Por tanto la suma minimal es:

$$f(w,x,y,z) = \bar{w}xy + \bar{w}yz + w\bar{y}z + wxy$$

Observación 4.6.3. Las siguientes sugerencias de uso de los *Mapas de Karnaugh* se basan en los hechos hasta ahora.

a) Comience a combinar los términos de la tabla donde haya como máximo una posibilidad para la minimización.

b) Verifique las cuatro esquinas del mapa, pues pueden adyacentes aunque parezca aislados.

c) En todas las simplificaciones, intente obtener el bloque máximo posible de unos adyacentes para obtener un término producto minimal (recuerde que los unos pueden usarse más de una vez , en caso necesario, debido a la ley de idempotencia) .

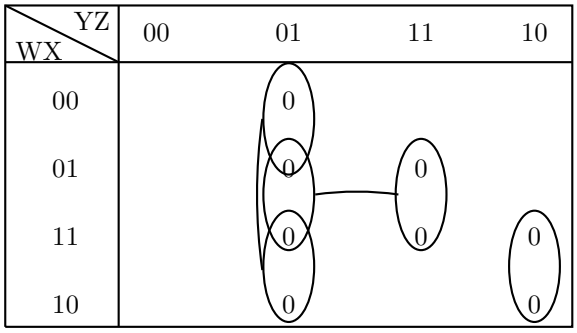
d) Si existe una opción para simplificar una entrada en el mapa, intentamos usar unos adyacentes que no hayan sido utilizados en una simplificación anterior.

Por último, presentaremos un ejemplo relacionado con el concepto dual, fnc.

Ejemplo 4.6.6. Hallar el producto minimal de sumas, para :

$$f(w,x,y,z) = \prod M(1,5,7,9,10,13,14,15)$$

Solución:
Aplicando el Concepto de dualidad, se tiene



$$g(w,x,y,z) = (\bar{x}+\bar{z})(y+\bar{z})(\bar{w}+\bar{y}+z)$$

Observación 4.6.4. Las siguientes sugerencias de uso de los *Mapas de Karnaugh* se basan en los hechos hasta ahora

a) Recordemos que el punto básico para la reducción de una fnd a una suma minimal es el reconocimiento de las adyacencias. En el mapa de *Karnaugh* determinamos donde existen esas adyacencias. Un segundo método para minimizar, es el procedimiento de *Quine McCluskey*[3], que organiza la fnd en una tabla a fin de simplificar la búsqueda de adyacencias.

b) El procedimiento es un proceso de dos pasos que se asemejan al mapa de *Karnaugh*. Primero encontramos localidades agrupables (como las adyacencias en el mapa de *Karnaugh*); en seguida, eliminamos los grupos redundantes y hacemos marcas para los términos que pueden pertenecer a diversos grupos. Ilustramos este procedimiento en el siguiente ejemplo.

Ejemplo 4.6.7. La función booleana a minimizar es representada en la siguiente tabla.

[3]Quine McCluskey

w	x	y	z	$f(w,x,y,z)$
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Las ocho 4-uplas de ceros y unos producen el valor 1 son listadas en una tabla, que es separada en cuatro grupos, de acuerdo con el número de unos.

Número de 1s	w	x	y	z
Tres	1	0	1	1
Dos	0	0	1	1
	0	1	0	1
	0	1	1	0
Uno	0	0	1	1
	0	0	1	0
	1	0	0	0
Cero	0	0	0	0

Comparamos el primer término 1011, con cada uno de los tres términos del segundo grupo a fin de encontrar términos adyacentes. Un término de este tipo es 0011, que determina que variable w es eliminada. Escribimos este término reducido con un trago en la posición w en dos términos 1011 y 0011 en la tabla original con un índice 1; este índice es un puntero que indica la linea del término reducido que es formada por esos dos elementos (numerar estas localidades equivale a trazar los grupos en un mapa de *Karnaugh*).

Continuamos este proceso con todos los términos. Un término numerado puede ser usado otras combinaciones, de la misma forma que podemos agrupar un término ya agrupado en un mapa de *Karnaugh*. Cuando hayamos terminado se obtendrá las siguientes tablas.

Número de 1s	w	x	y	z	
Tres	1	0	1	1	1
Dos	0	0	1	1	1,2,3
	0	1	0	1	4
	0	1	1	0	5
Uno	0	0	0	1	2,4,6
	0	0	1	0	3,5,7
	1	0	0	0	8
Cero	0	0	0	0	6,7,8

Tabla (a)

Número de 1s	w	x	y	z
Dos	-	0	1	1
Uno	0	0	-	1
	0	0	1	-
	0	-	0	1
	0	-	1	0
Cero	0	0	0	-
	0	0	-	0
	-	0	0	0

Tabla (b)

Repetimos este proceso en la tabla (b)

Número de 1s	w	x	y	z	
Dos	-	0	1	1	
Uno	0	0	-	1	
	0	0	1	-	-
	0	-	0	1	
	0	-	1	0	
Cero	0	0	0	-	1
	0	0	-	0	1
	-	0	0	0	

Tabla (c)

Número de 1s	w	x	y	z
cero	0	0	-	-

Tabla (d)

Observe que el proceso de reducción no puede ser continuado. Los términos que no fueron numerados son irreductibles, de forma que representan los mintérminos.

El segundo paso del proceso, consiste en comparar los términos irreductibles. Una marca en la tabla de comparación (Tabla (a)) indica que el término original en una columna acaba por llevar al término irreductible a esta linea, o que puede ser determinado siguiendo los punteros.

	1011	0011	0101	0110	0001	0010	1000	0000
-011	3	3						
0-01			3		3			
0-10				3		3		
-000							3	3
00–		3		3	3			3

Tabla(e)

Si una columna en la Tabla (e) tiene una marca en apenas una fila, el término irreductible de esta fila es el único que cubre al término original, de forma que el es un término esencial y precisa aparecer en la forma final de suma minimal de productos. Por tanto, vemos en la Tabla (e) que los términos -011, 0-01, 0-10, -000, son esenciales y precisan ser parte de la expresión final. También percibimos que todas las columnas con marcas en la fila 5 también son marcas

en otra fila y por tanto, es cubierta por un término esencia irreductible de la expresión, por tanto 00– es redundante. Por consiguiente la suma minimal de productos de f es:

$$f(w,x,y,z) = \bar{x}yz + \bar{w}\bar{y}z + \bar{w}y\bar{z} + \bar{x}\bar{y}\bar{z}$$

El procedimiento de *Quine-Mccluskey* se aplica a funciones booleanas con cualquier número de variables de entrada, pero para un número grande de variables, el procedimiento es demasiado monótono pra ser efectuado manualmente. Sin embargo, el es un buen procedimiento sistemático que lleva a una solución computarizada. Al contrario del mapa de *Karnaugh*, que se vale de la habilidad humana de reconocer patrones visuales.

Si la función booleana tuviera pocos valores 0 y un gran número de valores 1, es más facil implementar el procedimiento de *Quine-Mccluskey* para el complemento de la función, $\bar{f}$, que tendrá valores 1 donde f tenga valores 0 y viceversa. Una vez que se ha obtenido la suma minimal de productos para $\bar{f}$. Ella puede se complementada para obtener una expresión para f, pero la nueva expresión no será una suma minimal de productos. Por lo que, podemos obtener la red para f a partir de la red de suma minimal de productos para $\bar{f}$, incluyendo un inversor al final.

4.7. Lista de ejercicios

1. Sea n un entero positivo y sea D_n el conjunto de todos los divisores positivos de n. Así, $n = 20$ se tiene $D_{20} = \{1, 2, 4, 5, 10, 20\}$

   a) Hacer los diagramas de *Hasse* de D_{20} y D_{30} bajo el orden parcial de divisibilidad.

   b) Determinar si D_{20}, D_{30} y D_{60} son lattices.

   c) Determinar si D_{20}, D_{30} y D_{60} son álgebras booleanas.

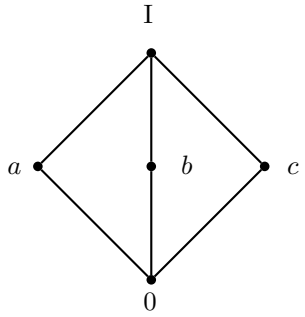
2. Determinar si las relaciones siguientes son de orden parcial en $A = N$

   a) $aRb \iff a \leq b + 1$

3. Una lattice es modular si, para $a, b, c, a <= b$ implica $a \vee (a \wedge c) = (a \vee b) \wedge c$

   a) Muestre que una lattice distributiva es modular.

   b) Muestre que la lattice de la figura es una lattice no distributiva que es modular.



4. Proporcione los diagramas de *Hasse* de todas las lattices no isomorfas con uno, dos, tres o cuatro elementos.

5. Escriba un algoritmo en pseudocódigo que construya el ordenamiento topológico de un conjunto parcialmente ordenado. Verifique su programa con un ejemplo.

6. Para cada una de las funciones booleanas f, encuentre una representación mediante una suma minimal de productos y luego diseñe una red de compuertas de dos niveles. Utilice diagramas de *Karnaugh*

   a) $f(x, y, z) = \prod M(0, 4, 5, 6)$
   b) $f(w, x, y, z) = \sum m(7, 9, 10, 11, 14, 15)$

7. Diseñe una red de compuertas lógicas de dos niveles para la suma minimal de productos de la siguiente función boolena.

$$f(w, x, y, z) = \sum m(5, 6, 8, 11, 12, 13, 14, 15)$$

8. Para cada una de las funciones booleanas, encuentre una representación mediante una suma minimal de productos y luego diseñe una red de compuertas de dos niveles. Utilice diagramas de *Karnaugh* .

$$f(w, x, y, z) = \sum m(7, 9, 10, 11, 14, 15)$$

9. Determine si el conjunto parcialmente ordenado es un álgebra booleana. Justifique su respuesta.

   a) D_{385}
   b) D_{60}

10. Sea $f : B^4 \to B$. encuentre la *fnd* y la *fnc* de f si.

$$f^{-1}(0) = \{0000, 0001, 0010, 0100, 1000, 1001, 0110\}$$

11. Simplifique la siguiente expresión booleana.

$$x + y + (\bar{x} + y + z)$$

12. Para cada una de las funciones booleanas f, encuentre una representación mediante una suma minimal de productos y luego diseñe una red de compuertas de dos niveles. Utilice diagramas de *Karnaugh*.

   a) $f(x, y, z) = \prod M(0, 4, 5, 6)$
   b) $f(w, x, y, z) = \sum m(7, 9, 10, 11, 14, 15)$

13. Sea n un entero positivo y se D_n el conjunto de todos los divisores de positivos de n. Así, si $n = 20$, se tiene $D_n = \{1, 2, 4, 5, 10, 20\}$

   a) Hacer los diagramas de *Hasse* de D_{20} y D_{30} bajo el orden parcial de divisibilidad.
   b) Determinar si D_{20} y D_{30} y D_{60} son lattices.

c) Determinar si D_{20} y D_{30} y D_{60} son álgebras booleanas.

14. Diseñe una red de compuertas lógicas de dos niveles para como suma minimal de productos. Utilice diagramas de *Karnaugh*

a) $f(x, y, z) = 1$ si y sólo si al menos dos variables tienen el valor 1.

b) $f(w, x, y, z) = \sum m(0, 2, 5, 7, 8, 10, 13, 15)$

Bibliografía

[Kolman, Busgy, Ross, 1997] *Estructuras de Matemáticas Discretas para la Computación* Prentice Hall, Mexico

[R Grimaldi, 1997] *Matemáticas Discreta y Combinatoria* Addison Wesley Iberoamericana, USA

[W. Grassmann, J. Tremblay, 1997] *Matemática Discreta y Lógica* Prentice Hall, España

[Richrad Johnsonbaugh, 1999] *Matemáticas Discretas* Prentice Hall, México

[J. Gersting, 1987] *Fundamentos Matemáticos para a Ciência da Computacão* ABDR, Rio de janeiro

[Ricardo Pea Mar, 1998] *Diseño de Programas: Formalismo y Abstracción* Prentice Hall, España

[Kenneth A. Ross, Charles R.B. Wright, 1990] *Matemáticas Discretas* Prentice Hall, México

[Edward R. Scheinerman] *Matemáticas Discretas* Thomson, México

[G. Brassard, P. Bratley, 1997] *Fundamentos de Algoritmia* Prentice Hall, España

[J. Glenn Brookshear, 1995] *Introducción a las Ciencias de la Computación* Addison Wesley Iberoamericana, USA