UNIVERSIDAD DE GRANADA

Lógica y Métodos Discretos

Temario de la asignatura

Ismael Sallami Moreno

Recursos Ingeniería Informática y Ade

Licencia

Lógica y Métodos Discretos

Ismael Sallami Moreno

https://elblogdeismael.github.io

Índice general

Capítulo 1

Álgebras de Boole y funciones booleanas

Bloque 1 del programa. La axiomática, las álgebras finitas y su representación atómica, las formas normales, los conjuntos funcionalmente completos y los circuitos combinacionales con su simplificación.

1.1 Axiomática

Definición 1.1 (Álgebra de Boole). Un conjunto B con dos operaciones binarias $\vee$ y $\wedge$, una unaria $'$ y dos elementos distinguidos 0 y 1, que cumple para todos $a, b, c \in B$:

- conmutativa: $a \vee b = b \vee a$ y $a \wedge b = b \wedge a$;
- distributiva de cada una respecto de la otra;
- elementos neutros: $a \vee 0 = a$ y $a \wedge 1 = a$;
- complemento: $a \vee a' = 1$ y $a \wedge a' = 0$.

Lo llamativo de la axiomática es la **doble distributiva**: en un anillo, el producto distribuye sobre la suma y no al revés. Aquí las dos operaciones están al mismo nivel, y de ahí sale el principio que ahorra la mitad del trabajo.

Proposición 1.1 (Principio de dualidad). Toda identidad válida en un álgebra de Boole sigue siéndolo al intercambiar $\vee$ con $\wedge$ y 0 con 1.

Los teoremas que se deducen, cada uno con su dual:

Nombre	Enunciado	Dual
Idempotencia	$a \vee a = a$	$a \wedge a = a$
Acotación	$a \vee 1 = 1$	$a \wedge 0 = 0$
Absorción	$a \vee (a \wedge b) = a$	$a \wedge (a \vee b) = a$
Involución	$(a')' = a$	—
De Morgan	$(a \vee b)' = a' \wedge b'$	$(a \wedge b)' = a' \vee b'$
Unicidad del complemento	si $a \vee x = 1$ y $a \wedge x = 0$, entonces $x = a'$	

Ejemplos que son álgebras de Boole:

Conjunto	$\vee$	$\wedge$	$'$
$\{0,1\}$	OR	AND	NOT
$\mathcal{P}(U)$	unión	intersección	complementario
Divisores de n libre de cuadrados	mcm	mcd	n/d
Funciones de $\{0,1\}^n$ en $\{0,1\}$	punto a punto	punto a punto	punto a punto

El segundo es el que explica que la teoría de conjuntos y la lógica se parezcan tanto: no se parecen, son la misma estructura.

1.2 Orden y representación atómica

Toda álgebra de Boole tiene un orden natural:

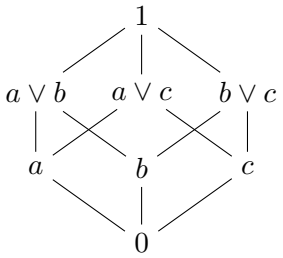
$$a \leq b \iff a \wedge b = a \iff a \vee b = b$$

Con él, B es un retículo con máximo 1 y mínimo 0, complementado y distributivo.

Definición 1.2 (Átomo). $a \neq 0$ es un átomo si no hay ningún x con $0 < x < a$.

Teorema 1.1 (Representación de Stone, caso finito). Toda álgebra de Boole finita es isomorfa al álgebra de partes del conjunto de sus átomos. En consecuencia, su cardinal es una potencia de 2, y dos álgebras de Boole finitas con el mismo número de elementos son isomorfas.

El teorema tiene una consecuencia tajante: **no existe ningún álgebra de Boole con 6 elementos**, ni con 12, ni con ningún cardinal que no sea potencia de dos. Y todas las de 16 elementos son «la misma».



álgebra de 8 elementos, con tres átomos

1.3 Funciones booleanas

Definición 1.3 . Una función booleana de n variables es $f : \{0,1\}^n \to \{0,1\}$.

Hay 2^{2^n} funciones distintas de n variables: la tabla de verdad tiene 2^n filas y cada una se rellena de dos formas. Con $n = 2$ son 16 y con $n = 5$ son más de cuatro mil millones, lo que explica que no se puedan enumerar y haga falta teoría.

1.3.1 Formas normales

Forma	Construcción
Disyuntiva (FND)	disyunción de los mintérminos donde f vale 1
Conjuntiva (FNC)	conjunción de los maxtérminos donde f vale 0

Las dos formas canónicas **existen siempre y son únicas**, y en eso se apoya todo lo demás: para probar que dos expresiones son equivalentes basta llevarlas a forma normal y comparar.

Ejemplo 1.1 . Sea $f(x, y, z)$ que vale 1 en las filas 1, 2, 4 y 7 —numerando desde 0—. Su forma normal disyuntiva es

$$f = x'y'z \vee x'yz' \vee xy'z' \vee xyz$$

que es la función «paridad impar»: vale 1 cuando el número de unos es impar. Es la XOR de tres variables, y su forma normal necesita los cuatro términos: **no se simplifica**.

La XOR es el ejemplo estándar de función cuya forma normal es la mínima, y por eso los circuitos de paridad no se pueden abaratar con mapas de Karnaugh.

1.4 Conjuntos funcionalmente completos

Definición 1.4 . Un conjunto de conectivas es funcionalmente completo si toda función booleana se puede expresar usando solo esas.

Conjunto	¿Completo?	Por qué
$\{\wedge, \vee, '\}$	sí	la forma normal disyuntiva las usa
$\{\wedge, '\}$	sí	De Morgan da $\vee$
$\{\vee, '\}$	sí	De Morgan da $\wedge$
$\{\uparrow\}$ (NAND)	sí	genera $'$, $\wedge$ y $\vee$
$\{\downarrow\}$ (NOR)	sí	ídem
$\{\wedge, \vee\}$	**no**	sin negación no se puede obtener el complemento
$\{\rightarrow\}$	no	falta el 0

La demostración de que $\{\wedge, \vee\}$ no es completo se hace por **monotonía**: toda función escrita solo con esas conectivas cumple que aumentar una entrada de 0 a 1 no baja la salida, y la negación no lo cumple. Es un argumento de invariante, y la técnica general para probar que algo no se puede hacer.

Que **NAND baste por sí sola** es lo que permite fabricar cualquier circuito con una única celda:

$$a' = a \uparrow a, \qquad a \wedge b = (a \uparrow b) \uparrow (a \uparrow b), \qquad a \vee b = (a \uparrow a) \uparrow (b \uparrow b)$$

1.5 Circuitos combinacionales

Una función booleana se realiza con puertas lógicas, y la forma normal disyuntiva da un circuito de dos niveles: una capa de AND y una OR final.

Coste	Qué mide
Número de puertas	área del circuito
Número de entradas totales	complejidad de la conexión
Número de niveles	retardo de propagación

Los dos objetivos —pocas puertas y pocos niveles— **compiten entre sí**, y por eso la minimización tiene siempre un criterio explícito detrás.

1.5.1 Simplificación

Método	Alcance
Manipulación algebraica	cualquier tamaño, sin garantía de llegar al mínimo
Mapas de Karnaugh	hasta 4 o 5 variables, visual
Quine-McCluskey	sistemático y automatizable, coste exponencial en el peor caso
Heurísticas	lo que usan las herramientas reales

Quine-McCluskey en dos fases:

1. **Obtener los implicantes primos.** Se agrupan los mintérminos que difieren en un solo bit, marcando con un guion la variable eliminada, y se repite hasta que no haya más combinaciones. Lo que queda sin combinar son los implicantes primos.
2. **Elegir una cobertura mínima.** Se construye la tabla de implicantes primos contra mintérminos; los primos que cubren en solitario algún mintérmino son **esenciales** y entran obligatoriamente; el resto se completa con el menor número posible.

Ejemplo 1.2 . Para $f = \sum m(0,1,2,5,6,7)$ sobre tres variables, la primera fase combina $(0,1)$, $(0,2)$, $(1,5)$, $(2,6)$, $(5,7)$ y $(6,7)$, y en la segunda pasada no queda ninguna combinación posible. Los seis implicantes primos son $x'y'$, $x'z'$, $y'z$, yz', xz y xy.

Ninguno es esencial: cada mintérmino está cubierto por dos primos. La cobertura mínima tiene tres, por ejemplo $f = x'y' \vee yz' \vee xz$.

La segunda fase es el problema de **cobertura de conjuntos**, que es NP-completo. Por eso las herramientas de síntesis reales no buscan el óptimo: aplican heurísticas y aceptan un resultado bueno.

Nota 1.1 . Que un problema tenga solución algorítmica no significa que sea abordable. Quine-McCluskey termina siempre y su número de implicantes primos puede crecer como $3^n/n$. Es la diferencia entre decidible y tratable, que reaparece en Algorítmica.

1.6 Ejercicios

Ejercicio 1.6.1. Demostrar la ley de absorción $a \vee (a \wedge b) = a$ a partir de los axiomas.

Solución 1.6.1. $a \vee (a \wedge b) = (a \wedge 1) \vee (a \wedge b) = a \wedge (1 \vee b) = a \wedge 1 = a$, usando el neutro, la distributiva y la acotación. La dual, $a \wedge (a \vee b) = a$, sale por el principio de dualidad sin repetir el cálculo.

Ejercicio 1.6.2. ¿Cuántos elementos puede tener un álgebra de Boole finita? ¿Existe alguna con 12?

Solución 1.6.2. Solo potencias de 2. Por el teorema de representación, toda álgebra de Boole finita es isomorfa a $\mathcal{P}(A)$ con A el conjunto de sus átomos, y $|\mathcal{P}(A)| = 2^{|A|}$. Como 12 no es potencia de 2, no existe ninguna con 12 elementos.

Ejercicio 1.6.3. Expresar $a \to b$ usando solo NAND.

Solución 1.6.3. $a \to b \equiv a' \vee b$. Con $a' = a \uparrow a$ y $x \vee y = (x \uparrow x) \uparrow (y \uparrow y)$ queda

$$a \to b \equiv ((a \uparrow a) \uparrow (a \uparrow a)) \uparrow (b \uparrow b)$$

que se simplifica a $a \uparrow (b \uparrow b)$, porque $(a \uparrow a) \uparrow (a \uparrow a) = a$.

El desarrollo del álgebra de Boole está en García Miranda, 2017 y Permingeat y Glaude, 1992, y su aplicación a circuitos en Rosen, 2003 y Grimaldi, 1997.

Capítulo 2

Lógica proposicional

Bloque 2 del programa. El lenguaje proposicional, la implicación semántica y sus propiedades, la forma clausulada de una fórmula, y los algoritmos de Davis-Putnam y de resolución.

2.1 El lenguaje

Elemento	Qué es
Variables proposicionales	$p, q, r, \ldots$
Conectivas	$\neg,\ \wedge,\ \vee,\ \rightarrow,\ \leftrightarrow$
Fórmulas	lo que generan las reglas de formación

Las reglas de formación son recursivas: toda variable es fórmula; si α es fórmula, $\neg\alpha$ lo es; y si α y β lo son, también $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \rightarrow \beta)$ y $(\alpha \leftrightarrow \beta)$.

La precedencia habitual, de mayor a menor: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$. La implicación **asocia por la derecha**, así que $p \rightarrow q \rightarrow r$ significa $p \rightarrow (q \rightarrow r)$, que no es lo mismo que $(p \rightarrow q) \rightarrow r$.

2.1.1 Semántica

Una **valoración** asigna un valor de verdad a cada variable, y se extiende a las fórmulas por las tablas de las conectivas.

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

La columna de la implicación es la que choca: **con antecedente falso la implicación es verdadera**. La justificación operativa es que $p \rightarrow q$ solo afirma que no ocurre p sin q, y eso es exactamente $\neg p \vee q$.

Clasificación	Definición
Tautología	verdadera con toda valoración
Contradicción	falsa con toda valoración
Satisfacible	verdadera con alguna
Contingente	ni tautología ni contradicción

Clasificación	Definición

2.2 Implicación semántica

Definición 2.1 (Consecuencia lógica). $\Gamma \models \alpha$ si toda valoración que hace verdaderas todas las fórmulas de Γ hace verdadera α.

Teorema 2.1 (De la deducción). $\Gamma \cup \{\alpha\} \models \beta$ si y solo si $\Gamma \models \alpha \to \beta$.

Teorema 2.2 (Refutación). $\Gamma \models \alpha$ si y solo si $\Gamma \cup \{\neg\alpha\}$ es insatisfacible.

El teorema de refutación es el que hace mecanizable la lógica: convierte «probar que algo se sigue» en «probar que algo es contradictorio», y esto último tiene algoritmos. Es lo que hacen Davis-Putnam y la resolución.

Propiedad	Enunciado
Reflexividad	$\Gamma \models \alpha$ si $\alpha \in \Gamma$
Monotonía	si $\Gamma \models \alpha$ y $\Gamma \subseteq \Delta$, entonces $\Delta \models \alpha$
Transitividad	si $\Gamma \models \alpha$ y $\Delta \cup \{\alpha\} \models \beta$, entonces $\Gamma \cup \Delta \models \beta$

La **monotonía** merece un comentario: añadir premisas nunca invalida una conclusión. Es lo que distingue la lógica clásica del razonamiento cotidiano, donde información nueva sí puede hacer retirar una conclusión, y de ahí que existan lógicas no monótonas para representar el sentido común.

2.2.1 Equivalencias útiles

Equivalencia	Expresión
Implicación	$p \to q \equiv \neg p \vee q$
Contrapositiva	$p \to q \equiv \neg q \to \neg p$
Bicondicional	$p \leftrightarrow q \equiv (p \to q) \wedge (q \to p)$
De Morgan	$\neg(p \wedge q) \equiv \neg p \vee \neg q$
Distributiva	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
Absorción	$p \vee (p \wedge q) \equiv p$

La contrapositiva es correcta y **el recíproco no lo es**: de $p \to q$ no se sigue $q \to p$. Confundirlos es la falacia más común, y en las demostraciones se traduce en probar lo contrario de lo pedido.

2.3 Forma normal conjuntiva y clausulada

Definición 2.2 . Un *literal* es una variable o su negación. Una *cláusula* es una disyunción de literales. Una fórmula está en forma normal conjuntiva si es una conjunción de cláusulas.

Teorema 2.3 . Toda fórmula proposicional es lógicamente equivalente a una en forma normal conjuntiva.

El procedimiento para obtenerla:

1. Eliminar $\leftrightarrow$ y $\rightarrow$ con las equivalencias.
2. Empujar las negaciones hacia dentro con De Morgan hasta que solo afecten a variables.
3. Distribuir $\vee$ sobre $\wedge$.
4. Simplificar: quitar literales repetidos y cláusulas con un literal y su negación.

Ejemplo 2.1 .

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

Paso 1: $\neg(\neg p \vee q) \vee (r \wedge \neg p)$. Paso 2: $(p \wedge \neg q) \vee (r \wedge \neg p)$. Paso 3, distribuyendo:

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

Paso 4: la segunda cláusula es una tautología y se elimina. El conjunto clausulado es

$$\{\, \{p, r\},\ \{\neg q, r\},\ \{\neg q, \neg p\} \,\}$$

Nota 2.1 . El paso 3 puede hacer crecer la fórmula **exponencialmente**. Cuando solo importa la satisfacibilidad y no la equivalencia, se usa la transformación de Tseitin, que introduce variables auxiliares para nombrar subfórmulas y produce una forma clausulada de tamaño lineal, equisatisfacible aunque no equivalente. Es lo que hace todo resolutor SAT real.

2.4 El algoritmo de Davis-Putnam

Decide si un conjunto de cláusulas es satisfacible. Sobre un conjunto S:

Regla	Cuándo se aplica	Qué hace
Cláusula unitaria	hay una cláusula con un solo literal L	L debe ser cierto: se propaga
Literal puro	un literal aparece siempre con el mismo signo	se pone a cierto y se borran sus cláusulas
División	ninguna de las anteriores	se prueba p y $\neg p$ por separado

Y dos condiciones de parada: si S queda **vacío**, es satisfacible; si contiene la **cláusula vacía**, es insatisfacible, porque una disyunción sin literales no puede ser cierta.

Ejemplo 2.2 . $S = \{\{p, q\},\ \{\neg p, r\},\ \{\neg q, r\},\ \{\neg r\}\}$.

La cláusula $\{\neg r\}$ es unitaria: r es falso. Propagando, $\{\neg p, r\}$ pasa a $\{\neg p\}$ y $\{\neg q, r\}$ a $\{\neg q\}$. Ahora $\{\neg p\}$ es unitaria: p es falso, y $\{p, q\}$ pasa a $\{q\}$. Pero $\{\neg q\}$ obliga a q falso y $\{q\}$ a q cierto: se produce la cláusula vacía. **El conjunto es insatisfacible.**

La **propagación unitaria** es la regla que más trabajo ahorra, y sigue siendo el corazón de los resolutores modernos, que son descendientes directos de este algoritmo.

2.5 Resolución

Una única regla de inferencia:

$$\frac{\{L\} \cup C_1 \qquad \{\neg L\} \cup C_2}{C_1 \cup C_2}$$

Es decir: de dos cláusulas con un literal complementario se deduce la unión del resto.

Teorema 2.4 (Corrección y completitud refutacional). La resolución es correcta: toda resolvente es consecuencia de sus premisas. Y es refutacionalmente completa: un conjunto de cláusulas es insatisfacible si y solo si la cláusula vacía es derivable por resolución.

Completa para refutar y no para deducir: la resolución no genera todas las consecuencias de un conjunto, solo detecta la contradicción. Con el teorema de refutación eso basta, y por eso para probar $\Gamma \models \alpha$ se resuelve sobre $\Gamma \cup \{\neg\alpha\}$.

Ejemplo 2.3 . Probar que de $p \to q$ y $q \to r$ se sigue $p \to r$.

Se niega la conclusión y se pasa todo a cláusulas:

$$\{\neg p, q\}, \quad \{\neg q, r\}, \quad \{p\}, \quad \{\neg r\}$$

Resolviendo $\{p\}$ con $\{\neg p, q\}$ sale $\{q\}$; con $\{\neg q, r\}$ sale $\{r\}$; y con $\{\neg r\}$ sale la cláusula vacía. Queda probado.

2.5.1 Estrategias

La resolución sin control genera muchísimas cláusulas inútiles. Las restricciones habituales:

Estrategia	Restricción
Lineal	cada resolvente usa la anterior como una de sus premisas
Lineal-input	además, la otra premisa es una cláusula de entrada
Unitaria	una de las dos premisas es unitaria
Por conjunto soporte	al menos una premisa desciende de la conclusión negada

Lineal-input es completa solo para cláusulas de Horn, es decir, con como mucho un literal positivo. Esa restricción es exactamente la que define Prolog, y es lo que hace que su motor de inferencia sea eficiente. El precio es que no todo se puede expresar en Horn.

2.6 Ejercicios

Ejercicio 2.6.1. ¿Es $((p \to q) \to p) \to p$ una tautología?

Solución 2.6.1. Sí, es la ley de Peirce. Si p es verdadera, la implicación exterior tiene consecuente

verdadero y es cierta. Si p es falsa, entonces $p \to q$ es verdadera, luego $(p \to q) \to p$ es falsa, y una implicación con antecedente falso es cierta. En los dos casos la fórmula vale 1.

Ejercicio 2.6.2. Llevar $(p \leftrightarrow q) \to r$ a forma clausulada.

Solución 2.6.2. El bicondicional es $(p \to q) \wedge (q \to p) \equiv (\neg p \vee q) \wedge (\neg q \vee p)$. La implicación exterior da $\neg[(\neg p \vee q) \wedge (\neg q \vee p)] \vee r$, y por De Morgan $(p \wedge \neg q) \vee (q \wedge \neg p) \vee r$. Distribuyendo:

$$\{p, q, r\}, \quad \{p, \neg p, r\}, \quad \{\neg q, q, r\}, \quad \{\neg q, \neg p, r\}$$

Las dos centrales son tautologías y se eliminan, así que quedan $\{p, q, r\}$ y $\{\neg p, \neg q, r\}$.

Ejercicio 2.6.3. Comprobar por resolución si $\{\, \{p, q\}, \{\neg p, q\}, \{p, \neg q\}, \{\neg p, \neg q\} \,\}$ es satisfacible.

Solución 2.6.3. Resolviendo la primera con la segunda sobre p sale $\{q\}$; la tercera con la cuarta sale $\{\neg q\}$; y esas dos entre sí dan la cláusula vacía. Es insatisfacible, como era de esperar: las cuatro cláusulas descartan las cuatro valoraciones posibles de dos variables.

El desarrollo de la lógica proposicional está en García Miranda, 2017 y Paniagua et al., 2003, la resolución en Chang y Lee, 1973, y los ejercicios resueltos en Hortalá et al., 2008.

Capítulo 3

Lenguajes de primer orden

Bloque 3 del programa. El lenguaje de primer orden, estructuras y valoraciones, implicación semántica, equivalencia lógica y formas normales.

3.1 Por qué hace falta

La lógica proposicional no distingue la estructura interna de los enunciados. El razonamiento «todos los primos mayores que 2 son impares; 7 es primo y mayor que 2; luego 7 es impar» es válido, y en proposicional serían tres variables sin relación alguna.

Lo que falta son **cuantificadores y predicados**: hablar de objetos, de sus propiedades y de las relaciones entre ellos.

3.2 El lenguaje

Elemento	Papel
Variables	x, y, z: designan objetos
Constantes	a, b, c: objetos concretos
Símbolos de función	f, g: construyen objetos a partir de otros
Símbolos de predicado	P, Q: propiedades y relaciones
Cuantificadores	$\forall$, $\exists$
Conectivas	las de proposicional

Y dos categorías sintácticas que no hay que mezclar:

Categoría	Qué es	Ejemplo
Término	designa un objeto	x, a, $f(x, a)$
Fórmula	afirma algo	$P(x)$, $\forall x\,(P(x) \to Q(x))$

Un término no es verdadero ni falso, y una fórmula no designa un objeto. Confundirlos produce expresiones sin sentido, del estilo $f(P(x))$.

3.2.1 Variables libres y ligadas

Una aparición de una variable está **ligada** si cae bajo el alcance de un cuantificador sobre ella, y **libre** en otro caso. Una fórmula sin variables libres es una **sentencia**, y solo las sentencias son verdaderas o falsas por sí solas.

Ejemplo 3.1 . En $\forall x\,(P(x) \to Q(x,y))$, la variable x está ligada y la y libre. No es una sentencia: su verdad depende de qué objeto designe y.

Nota 3.1 . Al sustituir una variable libre por un término hay que evitar la **captura**: si el término contiene una variable que quedaría ligada por un cuantificador de la fórmula, primero se renombra el cuantificador. Sustituir x por y en $\exists y\,(x \neq y)$ daría $\exists y\,(y \neq y)$, que es falsa siempre, mientras que la original decía algo razonable.

3.3 Estructuras y valoraciones

Definición 3.1 (Estructura). Una estructura para un lenguaje consta de un conjunto no vacío D, el dominio, y una interpretación que asigna a cada constante un elemento de D, a cada símbolo de función de aridad n una función $D^n \to D$, y a cada predicado de aridad n una relación sobre D^n.

La verdad se define por recursión sobre la fórmula, y los dos casos que importan son:

$$\mathcal{A} \models \forall x\,\alpha \iff \mathcal{A} \models \alpha[x/d] \text{ para todo } d \in D$$

$$\mathcal{A} \models \exists x\,\alpha \iff \mathcal{A} \models \alpha[x/d] \text{ para algún } d \in D$$

Clasificación	Definición
Válida	verdadera en toda estructura
Satisfacible	verdadera en alguna
Insatisfacible	falsa en todas
$\Gamma \models \alpha$	toda estructura que satisface Γ satisface α

Ejemplo 3.2 . $\forall x\,\exists y\,(x < y)$ es verdadera en $\mathbb{N}$ con el orden usual y falsa en $\{1,2,3\}$ con el mismo orden. La misma fórmula cambia de valor según la estructura, que es toda la diferencia con la lógica proposicional.

Nota 3.2 . **El orden de los cuantificadores no se puede cambiar.** En $\mathbb{N}$, $\forall x\,\exists y\,(x < y)$ es verdadera —para cada número hay uno mayor— y $\exists y\,\forall x\,(x < y)$ es falsa —no hay un número mayor que todos—. Es el error más frecuente al formalizar, y en informática aparece cada vez que se confunde «para todo dato existe un algoritmo» con «existe un algoritmo para todo dato».

3.4 Equivalencia lógica

Equivalencia	Expresión
Negación del universal	$\neg\forall x\,\alpha \equiv \exists x\,\neg\alpha$
Negación del existencial	$\neg\exists x\,\alpha \equiv \forall x\,\neg\alpha$
Distribución del universal	$\forall x(\alpha \wedge \beta) \equiv \forall x\,\alpha \wedge \forall x\,\beta$
Distribución del existencial	$\exists x(\alpha \vee \beta) \equiv \exists x\,\alpha \vee \exists x\,\beta$
Cuantificador vacío	si x no es libre en β: $\forall x(\alpha \vee \beta) \equiv \forall x\,\alpha \vee \beta$
Conmutación de iguales	$\forall x\forall y \equiv \forall y\forall x$; ídem con $\exists$

Las dos primeras son De Morgan generalizado: negar «todos» da «alguno no», y negar «alguno» da «ninguno».

Y hay dos que **no** son equivalencias, solo implicaciones en un sentido:

$$\forall x\, \alpha \vee \forall x\, \beta \;\Rightarrow\; \forall x(\alpha \vee \beta)$$

$$\exists x(\alpha \wedge \beta) \;\Rightarrow\; \exists x\, \alpha \wedge \exists x\, \beta$$

Los recíprocos fallan. Para el primero, con α «es par» y β «es impar» sobre $\mathbb{N}$: todo número es par o impar, y no es cierto que todos sean pares ni que todos sean impares.

3.5 Formas normales

3.5.1 Forma normal prenexa

Todos los cuantificadores delante, seguidos de una matriz sin cuantificadores:

$$Q_1 x_1\, Q_2 x_2 \ldots Q_n x_n\; \varphi$$

El procedimiento:

1. Eliminar $\rightarrow$ y $\leftrightarrow$.
2. Empujar las negaciones hasta los átomos, negando los cuantificadores al pasar.
3. **Renombrar** las variables ligadas para que no se repitan.
4. Sacar los cuantificadores hacia fuera.

El paso 3 es obligatorio y se olvida. Sin renombrar, sacar cuantificadores captura variables y cambia el significado de la fórmula.

3.5.2 Forma de Skolem

Para decidir satisfacibilidad se eliminan los cuantificadores existenciales sustituyendo su variable por un término nuevo:

Situación del $\exists y$	Se sustituye y por
Sin universales delante	una **constante** nueva
Con $\forall x_1 \ldots \forall x_k$ delante	una **función** nueva $f(x_1, \ldots, x_k)$

Teorema 3.1 . La forma de Skolem no es lógicamente equivalente a la original, pero es **equisatisfacible**: una es satisfacible si y solo si lo es la otra.

Que la función de Skolem dependa de las variables universales que la preceden es justamente la asimetría del orden de los cuantificadores. En $\forall x \exists y\, (x < y)$, el y depende de x, y por eso se sustituye por $f(x)$ y no por una constante.

Después se lleva la matriz a forma clausulada, y el resultado es un conjunto de cláusulas con variables implícitamente cuantificadas universalmente. **Ese es el formato de entrada del bloque siguiente.**

Ejemplo 3.3 .

$$\forall x \left(P(x) \to \exists y \left(Q(y) \land R(x,y)\right)\right)$$

Paso 1: $\forall x \left(\neg P(x) \lor \exists y \left(Q(y) \land R(x,y)\right)\right)$. Prenexa: $\forall x \exists y \left(\neg P(x) \lor \left(Q(y) \land R(x,y)\right)\right)$. Skolem, con $y = f(x)$: $\forall x \left(\neg P(x) \lor \left(Q(f(x)) \land R(x, f(x))\right)\right)$. Clausulada:

$$\{\neg P(x), Q(f(x))\}, \qquad \{\neg P(x), R(x, f(x))\}$$

3.6 Límites

Dos resultados que conviene conocer aunque no se demuestren aquí, porque marcan lo que la lógica de primer orden puede y no puede hacer:

Resultado	Qué dice
Completitud de Gödel	hay un cálculo deductivo que deriva exactamente las fórmulas válidas
Indecidibilidad de Church-Turing	no hay algoritmo que decida en general si una fórmula es válida

La combinación es característica: la validez es **semidecidible**. Un procedimiento que enumera las demostraciones acaba encontrando la de cualquier fórmula válida, y con una que no lo sea puede no terminar nunca. Es la razón de que los demostradores automáticos lleven siempre un límite de tiempo.

3.7 Ejercicios

Ejercicio 3.7.1. Formalizar: «todo estudiante que aprueba todas las asignaturas obtiene el título».

Solución 3.7.1. Con $E(x)$ «es estudiante», $A(y)$ «es asignatura», $P(x,y)$ «x aprueba y» y $T(x)$ «x obtiene el título»:

$$\forall x \left(E(x) \land \forall y \left(A(y) \to P(x,y)\right) \;\to\; T(x)\right)$$

El $\forall y$ va *dentro* del antecedente. Sacarlo fuera cambiaría el sentido y diría que para toda asignatura ocurre algo, en vez de exigir que se aprueben todas.

Ejercicio 3.7.2. Obtener la forma de Skolem de $\exists x \, \forall y \, \exists z \left(P(x,y) \to Q(y,z)\right)$.

Solución 3.7.2. El $\exists x$ no tiene universales delante, así que x se sustituye por una constante a. El $\exists z$ va detrás de $\forall y$, así que z pasa a $g(y)$. Queda

$$\forall y \left(P(a,y) \to Q(y, g(y))\right)$$

y en forma clausulada, $\{\neg P(a,y),\ Q(y,g(y))\}$.

Ejercicio 3.7.3. Dar una estructura donde $\forall x \exists y \, R(x,y)$ sea verdadera y $\exists y \forall x \, R(x,y)$ falsa.

Solución 3.7.3. Con dominio $\mathbb{N}$ y $R(x,y)$ interpretado como $x < y$: para cada número hay uno mayor, así que la primera es verdadera; y no existe un número mayor que todos, luego la segunda

es falsa. Cualquier orden sin máximo sirve.

El desarrollo de los lenguajes de primer orden está en García Miranda, 2017 y Chang y Lee, 1973, y sus ejercicios resueltos en Hortalá et al., 2008.

Capítulo 4

Unificación y resolución

Bloque 4 del programa. El algoritmo de unificación, el principio de resolución en primer orden, y las estrategias lineal, lineal-input y lineal-input ordenada.

4.1 Del proposicional al primer orden

La resolución del bloque 2 se aplicaba a cláusulas sin variables. Con variables aparece un problema nuevo: dos literales pueden ser complementarios **después de sustituir**, sin serlo antes.

$$P(x) \quad \text{y} \quad \neg P(a)$$

No son complementarios literalmente, y sí lo son sustituyendo x por a. Encontrar esa sustitución es lo que hace la unificación.

4.2 Sustituciones

Definición 4.1 (Sustitución). Aplicación finita $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$ que reemplaza cada variable x_i por el término t_i. Se aplica *simultáneamente*, no en cadena.

Que sea simultánea importa: aplicar $\{x/y,\ y/a\}$ a $P(x, y)$ da $P(y, a)$, y no $P(a, a)$.

La **composición** $\sigma\theta$ es aplicar primero σ y después θ. Es asociativa y no conmutativa.

Definición 4.2 (Unificador). σ unifica un conjunto de expresiones si todas se vuelven iguales al aplicarla. Es un *unificador de máxima generalidad* (umg) si todo otro unificador θ se factoriza como $\theta = \sigma\lambda$ para alguna λ.

Teorema 4.1 (De unificación). Si un conjunto de expresiones es unificable, el algoritmo de unificación calcula un unificador de máxima generalidad, único salvo renombramiento de variables. Si no lo es, el algoritmo lo detecta y termina.

La máxima generalidad es lo que hace correcta a la resolución. Un unificador cualquiera podría concretar de más y perder soluciones; el umg sustituye lo mínimo imprescindible.

4.3 El algoritmo de unificación

Sobre un conjunto W de expresiones, partiendo de $\sigma = \varepsilon$:

1. Si todas las expresiones de $W\sigma$ son iguales, devolver σ.
2. Localizar el **conjunto de discrepancia**: la primera posición, de izquierda a derecha, donde

las expresiones difieren.
3. Si en esa posición hay una variable x y un término t con x **no** en t, componer σ con $\{x/t\}$ y volver al paso 1.
4. En cualquier otro caso, el conjunto no es unificable.

4.3.1 El control de ocurrencia

El paso 3 exige que x no aparezca en t. Sin esa comprobación, unificar x con $f(x)$ produciría la sustitución $\{x/f(x)\}$ y, al aplicarla repetidamente, un término infinito.

Nota 4.1 . Prolog **omite el control de ocurrencia** por rendimiento: comprobarlo cuesta tiempo lineal en el tamaño del término y casi nunca hace falta. El precio es que `X = f(X)` construye una estructura cíclica y un intento de imprimirla puede no terminar. Es un caso de corrección sacrificada a propósito, documentado en el estándar.

Ejemplo 4.1 . Unificar $P(x,\ f(y),\ b)$ con $P(g(z),\ f(a),\ b)$.

Primera discrepancia: x frente a $g(z)$. Como x no está en $g(z)$, se toma $\{x/g(z)\}$.

Las expresiones son ahora $P(g(z), f(y), b)$ y $P(g(z), f(a), b)$. Discrepancia: y frente a a, y se añade $\{y/a\}$.

Ya coinciden. El umg es $\sigma = \{x/g(z),\ y/a\}$. Nótese que z queda libre: un umg no concreta lo que no hace falta.

Ejemplo 4.2 . $P(x, x)$ y $P(a, b)$ con $a \neq b$ constantes: la primera discrepancia da $\{x/a\}$, y entonces las expresiones son $P(a, a)$ y $P(a, b)$, cuya discrepancia enfrenta dos constantes distintas. **No son unificables.**

4.4 Resolución en primer orden

Definición 4.3 (Regla de resolución binaria). Dadas dos cláusulas $C_1 = \{L\} \cup D_1$ y $C_2 = \{\neg M\} \cup D_2$ sin variables en común, si $\sigma = \mathrm{umg}(L, M)$ existe, la resolvente es

$$(D_1 \cup D_2)\sigma$$

«Sin variables en común» no es un detalle. Las variables de una cláusula están cuantificadas universalmente y son locales a ella, así que hay que renombrar antes de resolver. Sin renombrar, $P(x)$ y $\neg P(f(x))$ no unificarían por el control de ocurrencia, cuando en realidad la deducción es correcta.

Hace falta además la **factorización**: si dos literales de la misma cláusula unifican, se fusionan aplicando su umg. Sin ella la resolución **no es completa**.

Teorema 4.2 (Completitud refutacional). Un conjunto de cláusulas de primer orden es insatisfacible si y solo si la cláusula vacía es derivable por resolución con factorización.

El procedimiento completo para probar $\Gamma \models \alpha$:

1. Negar α y añadirla a Γ.

2. Llevar todo a forma prenexa, skolemizar y pasar a cláusulas.
3. Renombrar variables para que ninguna cláusula comparta ninguna.
4. Resolver hasta obtener la cláusula vacía.

Ejemplo 4.3 . De «todo hombre es mortal» y «Sócrates es hombre», deducir «Sócrates es mortal».

Cláusulas: $\{\neg H(x), M(x)\}$, $\{H(s)\}$, y la conclusión negada $\{\neg M(s)\}$.

Resolviendo la primera con la segunda, con $\sigma = \{x/s\}$, sale $\{M(s)\}$; y esa con la tercera da la cláusula vacía.

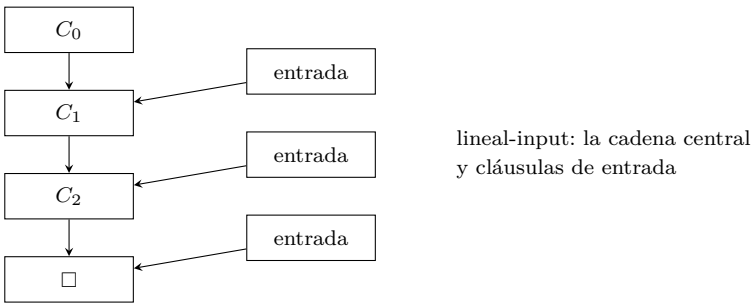
4.5 Estrategias

La resolución sin control genera un número enorme de cláusulas irrelevantes. Las restricciones que se estudian:

Estrategia	Restricción	¿Completa?
Por saturación	ninguna	sí, e inviable
Por conjunto soporte	una premisa desciende de la conclusión negada	sí
Lineal	cada resolvente usa la anterior	sí
Lineal-input	además, la otra premisa es de entrada	solo para Horn
Lineal-input ordenada	además, se resuelve sobre el primer literal	solo para Horn

4.5.1 Resolución lineal

Se parte de una cláusula inicial y cada paso resuelve la última resolvente con alguna cláusula del conjunto o con una resolvente anterior. La derivación es una cadena, no un árbol, y eso permite implementarla con una pila.



4.5.2 Cláusulas de Horn y Prolog

Definición 4.4 (Cláusula de Horn). Cláusula con como mucho un literal positivo.

Tipo	Forma	En Prolog
Hecho	$\{P\}$	`p.`

Tipo	Forma	En Prolog
Regla	$\{P, \neg Q_1, \ldots, \neg Q_n\}$	`p :- q1, ..., qn.`
Objetivo	$\{\neg Q_1, \ldots, \neg Q_n\}$	`?- q1, ..., qn.`

La resolución lineal-input ordenada es completa exactamente sobre cláusulas de Horn, y eso es lo que hace viable a Prolog: su motor recorre las cláusulas del programa en orden, resuelve siempre sobre el primer literal del objetivo, y retrocede al fallar.

Las dos consecuencias visibles del diseño:

- **El orden de las cláusulas del programa importa**, aunque lógicamente no debería. Una regla recursiva antes que su caso base produce un bucle infinito.
- **La negación es por fallo**, no negación lógica: `\+ p` significa «no se ha podido demostrar p», que solo coincide con «p es falso» bajo la hipótesis de mundo cerrado.

Nota 4.2 . Restringirse a Horn tiene precio: no se puede expresar «p o q» sin decir cuál. La disyunción en la cabeza es justo lo que Horn prohíbe, y por eso hay problemas naturales que Prolog no representa directamente.

4.6 Ejercicios

Ejercicio 4.6.1. Unificar, si es posible, $Q(f(x), g(y))$ con $Q(f(a), g(h(z)))$.

Solución 4.6.1. Primera discrepancia dentro de f: x frente a a, que da $\{x/a\}$. Segunda, dentro de g: y frente a $h(z)$, y como y no aparece en $h(z)$ se añade $\{y/h(z)\}$. El umg es $\{x/a,\ y/h(z)\}$, y z queda libre.

Ejercicio 4.6.2. ¿Por qué no se pueden unificar $P(x)$ y $P(f(x))$?

Solución 4.6.2. Por el control de ocurrencia: x aparece dentro de $f(x)$, así que la sustitución $\{x/f(x)\}$ no iguala las dos expresiones sino que las vuelve a separar, generando $f(f(x))$, $f(f(f(x)))$ y así indefinidamente. No hay ningún término finito que sea igual a f aplicado a sí mismo.

Ejercicio 4.6.3. Probar por resolución que de «todo el que estudia aprueba» y «alguien estudia» se deduce «alguien aprueba».

Solución 4.6.3. Cláusulas: $\{\neg E(x), A(x)\}$ de la primera premisa; $\{E(c)\}$ de la segunda, tras skolemizar el existencial con la constante c; y la conclusión negada, $\neg\exists y\, A(y) \equiv \forall y\, \neg A(y)$, que da $\{\neg A(y)\}$.

Resolviendo la primera con la segunda, con $\{x/c\}$, sale $\{A(c)\}$; y con la tercera, con $\{y/c\}$, la cláusula vacía.

El algoritmo de unificación y la resolución en primer orden están desarrollados en Chang y Lee, 1973 y García Miranda, 2017, con ejercicios resueltos en Hortalá et al., 2008 y Paniagua et al., 2003.

Capítulo 5

Inducción y recurrencia

Bloque 5 del programa. Los números naturales, el principio de inducción y el segundo principio, las definiciones recursivas, las recurrencias lineales con coeficientes constantes y los usos de la recursividad.

5.1 Los números naturales

$\mathbb{N}$ queda caracterizado por los axiomas de Peano, de los que el quinto es el que interesa aquí: **todo subconjunto de $\mathbb{N}$ que contenga al 0 y sea cerrado para el sucesor es todo $\mathbb{N}$**. Ese axioma es el principio de inducción.

Una formulación equivalente:

Teorema 5.1 (Buena ordenación). Todo subconjunto no vacío de $\mathbb{N}$ tiene mínimo.

Inducción y buena ordenación son equivalentes, y la segunda es la que se usa para demostrar por reducción al absurdo: si una propiedad falla en algún natural, hay un **mínimo** contraejemplo, y llegar a una contradicción con él suele ser fácil.

5.2 Principio de inducción

Teorema 5.2 (Inducción simple). Sea $P(n)$ una propiedad. Si $P(n_0)$ es cierta y para todo $n \geq n_0$ se cumple $P(n) \Rightarrow P(n+1)$, entonces $P(n)$ es cierta para todo $n \geq n_0$.

Teorema 5.3 (Inducción fuerte, o segundo principio). Si $P(n_0)$ es cierta y para todo $n > n_0$ se cumple

$$\big(P(n_0) \wedge \cdots \wedge P(n-1)\big) \Rightarrow P(n)$$

entonces $P(n)$ es cierta para todo $n \geq n_0$.

Los dos principios son equivalentes en potencia, y **la fuerte es más cómoda** cuando el paso necesita más de un caso anterior. Es lo que ocurre con la factorización en primos —$n = ab$ con $a, b < n$ cualesquiera— o con Fibonacci, que necesita los dos términos previos.

Ejemplo 5.1 . Demostrar que $\sum_{k=1}^{n} k = \dfrac{n(n+1)}{2}$.

Base: con $n = 1$, la suma es 1 y la fórmula da $1 \cdot 2/2 = 1$.

Paso: suponiendo la fórmula para n,

$$\sum_{k=1}^{n+1} k = \frac{n(n+1)}{2} + (n+1) = \frac{n(n+1)+2(n+1)}{2} = \frac{(n+1)(n+2)}{2}$$

que es la fórmula para $n+1$.

Nota 5.1 . Los dos errores clásicos. El primero, **olvidar la base**: sin ella se «demuestra» que todo natural es par, porque si n lo es también lo es $n+2$. El segundo, usar la hipótesis de inducción **sobre un caso que no está cubierto**, que es lo que hace la falsa demostración de que todos los caballos son del mismo color: el paso de $n=1$ a $n=2$ no funciona porque no hay ningún caballo compartido entre los dos grupos.

5.2.1 Variantes

Variante	Cuándo
Con varios casos base	el paso necesita k anteriores: hacen falta k bases
Estructural	sobre estructuras definidas recursivamente: listas, árboles, fórmulas
Sobre un orden bien fundado	cualquier orden sin cadenas descendentes infinitas

La **inducción estructural** es la que se usa constantemente en informática. Probar algo sobre todas las fórmulas de la lógica proposicional es probarlo sobre las variables y ver que las conectivas lo conservan; probar algo sobre un árbol binario es probarlo sobre la hoja y ver que se conserva al unir dos subárboles.

5.3 Definiciones recursivas

Una definición recursiva da los casos base y una regla que construye los demás a partir de anteriores. Para que esté bien definida hacen falta dos cosas:

- Que exista al menos un caso base.
- Que toda llamada recursiva **reduzca** el argumento respecto de un orden bien fundado.

Es exactamente lo que garantiza que una función recursiva termina, y por eso demostrar la terminación de un programa es exhibir esa medida decreciente.

Objeto	Base	Regla
Factorial	$0! = 1$	$n! = n\,(n-1)!$
Fibonacci	$F_0 = 0,\ F_1 = 1$	$F_n = F_{n-1} + F_{n-2}$
Listas	la lista vacía	una cabeza y una lista
Árboles binarios	el árbol vacío	raíz con dos subárboles
Fórmulas	las variables	conectivas aplicadas a fórmulas

5.4 Recurrencias lineales con coeficientes constantes

$$a_n + c_1 a_{n-1} + \cdots + c_k a_{n-k} = f(n)$$

Es **homogénea** si $f(n) = 0$, y de orden k.

5.4.1 Caso homogéneo

Se prueba $a_n = r^n$, y sustituyendo sale el **polinomio característico**:

$$r^k + c_1 r^{k-1} + \cdots + c_k = 0$$

Raíces	Solución general
k raíces simples $r_1, \ldots, r_k$	$a_n = \sum A_i r_i^n$
Raíz r de multiplicidad m	aporta $(A_1 + A_2 n + \cdots + A_m n^{m-1}) r^n$
Raíces complejas $\rho e^{\pm i\theta}$	$\rho^n (A \cos n\theta + B \operatorname{sen} n\theta)$

Las constantes se fijan con las condiciones iniciales, que son tantas como el orden.

Ejemplo 5.2 . $a_n = a_{n-1} + 2a_{n-2}$ con $a_0 = 2$ y $a_1 = 1$.

Polinomio característico: $r^2 - r - 2 = 0$, con raíces 2 y -1. La solución general es $a_n = A \cdot 2^n + B(-1)^n$. Imponiendo las condiciones: $A + B = 2$ y $2A - B = 1$, de donde $A = 1$ y $B = 1$.

$$a_n = 2^n + (-1)^n$$

Comprobación: $a_2 = 4 + 1 = 5$, y por la recurrencia $a_2 = 1 + 2 \cdot 2 = 5$.

Ejemplo 5.3 (Fibonacci). $F_n = F_{n-1} + F_{n-2}$ da $r^2 - r - 1 = 0$, con raíces $\varphi = (1 + \sqrt{5})/2$ y $\psi = (1 - \sqrt{5})/2$. Con $F_0 = 0$ y $F_1 = 1$ sale la fórmula de Binet

$$F_n = \frac{\varphi^n - \psi^n}{\sqrt{5}}$$

Como $|\psi| < 1$, el segundo término tiende a cero: F_n es el entero más próximo a $\varphi^n/\sqrt{5}$, y el cociente F_{n+1}/F_n tiende a la razón áurea.

5.4.2 Caso no homogéneo

$$a_n = a_n^{(h)} + a_n^{(p)}$$

La solución general de la homogénea más una particular. Para buscar la particular se prueba una del mismo tipo que $f(n)$:

$f(n)$	Se prueba
Constante	una constante
Polinomio de grado d	polinomio de grado d
b^n	Cb^n
Producto de los anteriores	el producto

Con una salvedad importante: si la forma propuesta ya es solución de la homogénea, hay que multiplicarla por n tantas veces como haga falta. Con $a_n = 2a_{n-1} + 2^n$, probar $C2^n$ falla porque 2^n resuelve la homogénea; lo que funciona es $Cn2^n$.

5.4.3 Recurrencias de divide y vencerás

Las que salen al analizar algoritmos recursivos:

$$T(n) = a\,T(n/b) + f(n)$$

Caso	Condición	Solución
1	$f(n) = O(n^{\log_b a - \varepsilon})$	$\Theta(n^{\log_b a})$
2	$f(n) = \Theta(n^{\log_b a})$	$\Theta(n^{\log_b a} \log n)$
3	$f(n) = \Omega(n^{\log_b a + \varepsilon})$ y regularidad	$\Theta(f(n))$

Algoritmo	Recurrencia	Coste
Búsqueda binaria	$T(n) = T(n/2) + \Theta(1)$	$\Theta(\log n)$
Mergesort	$T(n) = 2T(n/2) + \Theta(n)$	$\Theta(n \log n)$
Recorrido de un árbol	$T(n) = 2T(n/2) + \Theta(1)$	$\Theta(n)$
Karatsuba	$T(n) = 3T(n/2) + \Theta(n)$	$\Theta(n^{\log_2 3})$

La última fila explica por qué multiplicar con tres productos en lugar de cuatro compensa: $\log_2 3 \approx 1{,}585$ frente al 2 del método clásico.

5.5 Usos de la recursividad

Uso	Ejemplo
Definir estructuras	listas, árboles, gramáticas
Recorrerlas	los recorridos de un árbol
Dividir y vencer	mergesort, quicksort, búsqueda binaria
Vuelta atrás	n reinas, sudoku, laberintos
Programación dinámica	cuando los subproblemas se repiten

Ejemplo 5.4 (Torres de Hanói). Mover n discos exige mover $n-1$ a la varilla auxiliar, mover el mayor y volver a mover los $n-1$:

$$T(n) = 2T(n-1) + 1, \qquad T(1) = 1$$

La homogénea da $A2^n$ y la particular constante, -1. Con $T(1) = 1$ sale

$$T(n) = 2^n - 1$$

Con 64 discos y un movimiento por segundo, son unos 585 000 millones de años. Y esa es la cota inferior: no hay solución mejor.

Nota 5.2 . Un algoritmo recursivo puede ser desastroso sin cambiar su definición. Calcular Fibonacci con la recurrencia directa cuesta $\Theta(\varphi^n)$ porque recalcula los mismos valores una y otra vez; guardándolos, $\Theta(n)$. La recursión describe *qué* se calcula, no *cuántas veces*.

5.6 Ejercicios

Ejercicio 5.6.1. Demostrar por inducción que $n! > 2^n$ para todo $n \geq 4$.

Solución 5.6.1. *Base*: $4! = 24 > 16 = 2^4$.

Paso: si $n! > 2^n$ con $n \geq 4$, entonces $(n+1)! = (n+1)\,n! > (n+1)\,2^n \geq 5 \cdot 2^n > 2 \cdot 2^n = 2^{n+1}$. La desigualdad falla para $n \leq 3$, y por eso la base va en 4 y no en 0.

Ejercicio 5.6.2. Resolver $a_n = 5a_{n-1} - 6a_{n-2}$ con $a_0 = 1$ y $a_1 = 0$.

Solución 5.6.2. $r^2 - 5r + 6 = 0$ tiene raíces 2 y 3, así que $a_n = A2^n + B3^n$. De $A + B = 1$ y $2A + 3B = 0$ sale $B = -2$ y $A = 3$:

$$a_n = 3 \cdot 2^n - 2 \cdot 3^n$$

Comprobación: $a_2 = 12 - 18 = -6$, y por la recurrencia $5 \cdot 0 - 6 \cdot 1 = -6$.

Ejercicio 5.6.3. Resolver $a_n = 3a_{n-1} + 2^n$ con $a_0 = 1$.

Solución 5.6.3. La homogénea da $A3^n$. Como 2^n no resuelve la homogénea, se prueba $a_n^{(p)} = C2^n$: sustituyendo, $C2^n = 3C2^{n-1} + 2^n$, es decir $2C = 3C + 2$ y $C = -2$. La general es $a_n = A3^n - 2^{n+1}$, y con $a_0 = 1$ queda $A = 3$:

$$a_n = 3^{n+1} - 2^{n+1}$$

El principio de inducción y las recurrencias están desarrollados en Grimaldi, 1997 y Rosen, 2003, con más ejemplos en Gunderson, 2016 y problemas en Lipschutz, 2004.

Capítulo 6

Grafos y árboles

Bloque 6 del programa. Vértices y lados, matriz de adyacencia, tipos especiales, el algoritmo de Havel-Hakimi, caminos, grafos bipartidos, planos, coloración y árboles.

6.1 Definiciones

Definición 6.1 (Grafo). Un par $G = (V, E)$ con V un conjunto finito no vacío de vértices y E un conjunto de pares no ordenados de vértices, los lados.

Concepto	Definición
Adyacentes	dos vértices unidos por un lado
Incidente	un lado y cada uno de sus extremos
Grado $\mathrm{gr}(v)$	número de lados incidentes en v
Grafo simple	sin bucles ni lados repetidos
Multigrafo	admite lados repetidos
Dirigido	los lados son pares ordenados
Ponderado	cada lado lleva un peso

Teorema 6.1 (Del apretón de manos).

$$\sum_{v \in V} \mathrm{gr}(v) = 2|E|$$

En consecuencia, el número de vértices de grado impar es par.

Demostración 6.1 . Cada lado aporta uno al grado de cada uno de sus dos extremos, así que la suma de grados cuenta cada lado exactamente dos veces. Al ser la suma par, los sumandos impares tienen que ser un número par de ellos.

Es el resultado más rentable del bloque: descarta de un vistazo montones de grafos imposibles. **No existe un grafo con cinco vértices todos de grado 3**, porque la suma sería 15, impar.

6.2 Representación

Representación	Espacio	¿Hay lado (u, v)?	Vecinos de u
Matriz de adyacencia	$\Theta(n^2)$	$\Theta(1)$	$\Theta(n)$
Listas de adyacencia	$\Theta(n+m)$	$\Theta(\text{gr}(u))$	$\Theta(\text{gr}(u))$
Matriz de incidencia	$\Theta(nm)$	—	—

La matriz de adyacencia de un grafo no dirigido es **simétrica** con diagonal nula, y tiene una propiedad que se usa mucho:

Proposición 6.1 . El elemento (i, j) de A^k es el número de caminos de longitud k entre v_i y v_j.

De ahí que la traza de A^3 dividida por 6 cuente los triángulos del grafo, y que las potencias de la matriz respondan preguntas de conectividad sin recorrer nada.

6.3 Tipos especiales

Grafo	Notación	Descripción	Lados
Completo	K_n	todos con todos	$\binom{n}{2}$
Ciclo	C_n	un solo ciclo de n vértices	n
Camino	P_n	una cadena	$n-1$
Bipartido completo	$K_{m,n}$	dos clases, todo cruzado	mn
Rueda	W_n	un ciclo más un centro	$2n$
k-regular	—	todos los vértices de grado k	$nk/2$
Hipercubo	Q_k	cadenas de k bits, unidas si difieren en uno	$k\,2^{k-1}$

El **hipercubo** aparece en arquitecturas de interconexión de multiprocesadores: con 2^k nodos, dos cualesquiera están a distancia como mucho k, y cada nodo solo necesita k enlaces.

6.3.1 El algoritmo de Havel-Hakimi

Decide si una lista de números puede ser la sucesión de grados de un grafo simple.

Teorema 6.2 (Havel-Hakimi). La sucesión $d_1 \geq d_2 \geq \cdots \geq d_n$ es gráfica si y solo si lo es la que resulta de borrar d_1 y restar uno a los d_1 términos siguientes, reordenando después.

El procedimiento termina cuando quedan todos ceros —la sucesión es gráfica— o aparece un número negativo o mayor que la longitud restante —no lo es—.

Ejemplo 6.1 . $(4, 3, 3, 2, 2)$: se borra el 4 y se resta uno a los cuatro siguientes, que da $(2, 2, 1, 1)$. Se borra el 2 y se resta uno a los dos primeros: $(1, 0, 1)$, que reordenado es $(1, 1, 0)$. Se borra el 1 y se resta uno al siguiente: $(0, 0)$. Es gráfica.

$(4, 4, 4, 4, 4, 1)$: la suma es 21, impar, así que ni hace falta el algoritmo. El apretón de manos ya la descarta.

6.4 Caminos y conectividad

Concepto	Definición
Recorrido	secuencia de vértices consecutivamente adyacentes
Camino	recorrido sin vértices repetidos
Ciclo	camino cerrado
Conexo	hay camino entre todo par de vértices
Componente conexa	subgrafo conexo maximal
Distancia	longitud del camino más corto
Diámetro	la mayor de las distancias

6.4.1 Recorridos eulerianos y hamiltonianos

	Euleriano	Hamiltoniano
Recorre	todos los **lados** una vez	todos los **vértices** una vez
Caracterización	conexo y 0 o 2 vértices de grado impar	**no se conoce ninguna**
Decidir si existe	$\Theta(n+m)$	NP-completo

Ese contraste es la lección del apartado. Dos problemas casi idénticos de enunciado tienen dificultades opuestas: uno se resuelve mirando los grados, y del otro no se conoce ningún algoritmo eficiente.

Teorema 6.3 (Euler). Un grafo conexo tiene un circuito euleriano si y solo si todos sus vértices tienen grado par, y un camino euleriano abierto si y solo si tiene exactamente dos de grado impar.

Es el resultado con el que Euler resolvió el problema de los puentes de Königsberg: los cuatro vértices tenían grado impar, así que ningún recorrido era posible.

6.5 Grafos bipartidos

Definición 6.2 . G es bipartido si V se parte en dos conjuntos disjuntos de forma que todo lado une un vértice de uno con uno del otro.

Teorema 6.4 . Un grafo es bipartido si y solo si no contiene ciclos de longitud impar.

La caracterización da un algoritmo directo: se recorre en anchura pintando los niveles alternativamente de dos colores, y se comprueba que ningún lado une dos del mismo. Es $\Theta(n+m)$.

Los bipartidos son el marco de los problemas de **emparejamiento**: asignar tareas a personas, alumnos a plazas, o candidatos a puestos.

Teorema 6.5 (Hall). Un grafo bipartido con clases X e Y tiene un emparejamiento que satura X si y solo si $|N(S)| \geq |S|$ para todo $S \subseteq X$, donde $N(S)$ es el conjunto de vecinos de S.

La condición dice que ningún grupo de k elementos puede tener menos de k opciones entre todos. Es el mismo principio del palomar, aplicado a subconjuntos.

6.6 Grafos planos

Definición 6.3 . G es plano si admite un dibujo en el plano sin que se corten sus lados.

Teorema 6.6 (Fórmula de Euler). En un grafo plano conexo con n vértices, m lados y c caras,

$$n - m + c = 2$$

Corolario 6.1 . Un grafo plano simple con $n \geq 3$ cumple $m \leq 3n-6$. Si además no tiene triángulos, $m \leq 2n - 4$.

El corolario es la herramienta práctica: **si un grafo tiene demasiados lados, no es plano**, y basta contar.

Ejemplo 6.2 . K_5 tiene $n = 5$ y $m = 10$, y $3n - 6 = 9 < 10$: no es plano.

$K_{3,3}$ tiene $n = 6$ y $m = 9$, y $3n - 6 = 12 \geq 9$, así que el criterio no decide. Pero es bipartido y por tanto no tiene triángulos, luego se aplica $m \leq 2n - 4 = 8 < 9$: tampoco es plano.

Teorema 6.7 (Kuratowski). Un grafo es plano si y solo si no contiene ningún subgrafo homeomorfo a K_5 ni a $K_{3,3}$.

Que **los dos únicos obstáculos sean esos dos grafos** es un resultado notable: toda la no planaridad del mundo se reduce a dos configuraciones.

6.7 Coloración

Definición 6.4 . Una coloración propia asigna colores a los vértices de modo que dos adyacentes nunca compartan color. El número cromático $\chi(G)$ es el mínimo de colores necesario.

Grafo	χ
K_n	n
Bipartido con al menos un lado	2
Ciclo C_n con n par	2
Ciclo C_n con n impar	3
Árbol con al menos un lado	2
Plano	≤ 4

Teorema 6.8 (De los cuatro colores). Todo grafo plano se puede colorear con cuatro colores.

Enunciado en 1852 y demostrado en 1976 por Appel y Haken **con ayuda del ordenador**, comprobando 1936 configuraciones inevitables. Fue la primera demostración importante que nadie podía verificar a mano, y abrió el debate sobre qué cuenta como demostración.

Calcular $\chi(G)$ en general es NP-difícil, así que en la práctica se usan heurísticas voraces: se ordenan los vértices y se asigna a cada uno el menor color libre. El resultado depende del orden y nunca supera $\Delta + 1$ colores, con Δ el grado máximo.

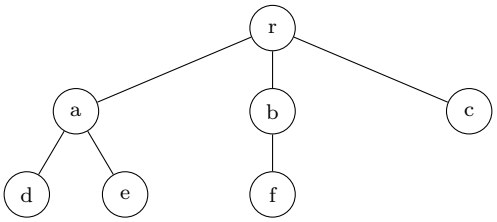
Dónde aparece la coloración: asignación de frecuencias, planificación de exámenes sin solapes, y asignación de registros en un compilador, donde los vértices son variables vivas y los lados los conflictos.

6.8 Árboles

Definición 6.5 . Un árbol es un grafo conexo sin ciclos.

Teorema 6.9 (Caracterizaciones). Para un grafo G con n vértices son equivalentes:

- G es un árbol.
- G es conexo y tiene $n - 1$ lados.
- G no tiene ciclos y tiene $n - 1$ lados.
- Entre todo par de vértices hay un único camino.
- G es conexo y quitar cualquier lado lo desconecta.
- G no tiene ciclos y añadir cualquier lado crea uno.



Concepto	Definición
Árbol con raíz	se designa un vértice como raíz
Hoja	vértice de grado 1 distinto de la raíz
Bosque	grafo sin ciclos, no necesariamente conexo
Árbol de recubrimiento	subgrafo que es árbol y contiene todos los vértices

Teorema 6.10 (Cayley). El número de árboles etiquetados distintos sobre n vértices es n^{n-2}.

Con 10 vértices son cien millones, lo que descarta cualquier enfoque por enumeración y justifica los algoritmos de Prim y Kruskal para el árbol de recubrimiento mínimo.

Problema	Algoritmo	Coste
Árbol de recubrimiento mínimo	Prim, Kruskal	$\Theta(m \log n)$
Camino mínimo con pesos no negativos	Dijkstra	$\Theta((n+m) \log n)$

Problema	Algoritmo	Coste
Recorrido en anchura y profundidad	BFS, DFS	$\Theta(n+m)$

Los árboles con raíz son además la estructura de los códigos de prefijo. En un **código de Huffman**, los símbolos van en las hojas y el camino desde la raíz da la codificación, y que estén en las hojas es lo que garantiza que ningún código sea prefijo de otro.

6.9 Ejercicios

Ejercicio 6.9.1. ¿Existe un grafo simple con 6 vértices y grados $(5, 5, 4, 3, 2, 1)$?

Solución 6.9.1. La suma es 20, par, así que el apretón de manos no lo descarta. Havel-Hakimi: se borra el primer 5 y se resta uno a los cinco siguientes, dando $(4, 3, 2, 1, 0)$. Se borra el 4 y se resta uno a los cuatro siguientes: $(2, 1, 0, -1)$. Aparece un negativo, así que la sucesión no es gráfica: no existe tal grafo.

Ejercicio 6.9.2. Un grafo plano conexo tiene 10 vértices y todas sus caras son triángulos. ¿Cuántos lados y caras tiene?

Solución 6.9.2. Cada cara está rodeada por 3 lados y cada lado separa 2 caras, luego $3c = 2m$. Con Euler, $10 - m + c = 2$ y $c = 2m/3$, de donde $10 - m + 2m/3 = 2$, es decir $m/3 = 8$ y $m = 24$. Entonces $c = 16$. Se comprueba la cota: $3n - 6 = 24 = m$, así que es un grafo plano maximal, como debe ser si todas las caras son triángulos.

Ejercicio 6.9.3. Demostrar que todo árbol con al menos dos vértices tiene al menos dos hojas.

Solución 6.9.3. Se toma el camino más largo del árbol, $v_0, \ldots, v_k$ con $k \geq 1$. Si v_0 tuviera otro vecino además de v_1, ese vecino o alarga el camino —contradicción con la maximalidad— o ya está en él, y entonces se forma un ciclo, imposible en un árbol. Luego v_0 es hoja, y por el mismo argumento v_k también.

El desarrollo de la teoría de grafos está en Biggs, 1998, Grimaldi, 1997 y Rosen, 2003, con problemas en Lipschutz, 2004 y Jiménez Murillo, 2015.

Capítulo 7

Relación de problemas

El temario práctico de la guía es la resolución de problemas en los grupos reducidos. Esta relación recorre los seis bloques con la solución completa.

7.1 Álgebras de Boole

Ejercicio 7.1.1. Simplificar $f(x, y, z) = \sum m(0, 2, 4, 5, 6)$ y dar el circuito con menos puertas.

Solución 7.1.1. En el mapa de Karnaugh, los mintérminos 0, 2, 4 y 6 son todos los que tienen $z = 0$: un grupo de cuatro que da z'. Queda el 5, que se agrupa con el 4 dando xy'. Por tanto

$$f = z' \vee xy'$$

Dos puertas y dos inversores, frente a los cinco términos de tres literales de la forma canónica.

Ejercicio 7.1.2. Demostrar que $\{\wedge, \vee\}$ no es funcionalmente completo.

Solución 7.1.2. Por inducción estructural: toda función construida solo con $\wedge$ y $\vee$ es *monótona*, es decir, cambiar una entrada de 0 a 1 nunca hace bajar la salida. Es cierto para las variables, y se conserva al aplicar $\wedge$ y $\vee$, que son monótonas en cada argumento. La negación no lo es, luego no se puede expresar.

7.2 Lógica proposicional

Ejercicio 7.2.1. Decidir si $\{p \to q,\ q \to r,\ \neg r\} \models \neg p$.

Solución 7.2.1. Por refutación: se añade p y se pasa todo a cláusulas, que da $\{\neg p, q\}$, $\{\neg q, r\}$, $\{\neg r\}$ y $\{p\}$. Resolviendo $\{p\}$ con la primera sale $\{q\}$; con la segunda, $\{r\}$; y con $\{\neg r\}$, la cláusula vacía. La consecuencia es válida: es el *modus tollens* encadenado.

Ejercicio 7.2.2. Un conjunto de cláusulas contiene $\{p\}$, $\{\neg p, q\}$, $\{\neg q, r\}$ y $\{\neg r, s\}$. Aplicar Davis-Putnam.

Solución 7.2.2. $\{p\}$ es unitaria: p verdadero. Propagando desaparece $\{p\}$ y $\{\neg p, q\}$ pasa a $\{q\}$, otra unitaria. Repitiendo, se obtienen r y s verdaderos y el conjunto queda vacío: es satisfacible, con la valoración que hace verdaderas las cuatro variables. Toda la resolución ha sido propagación unitaria, sin una sola división.

7.3 Primer orden

Ejercicio 7.3.1. Formalizar y decidir: «hay alguien a quien todos admiran» frente a «todos admiran a alguien».

Solución 7.3.1. Con $A(x, y)$ «x admira a y»:

$$\exists y \, \forall x \, A(x, y) \qquad \text{y} \qquad \forall x \, \exists y \, A(x, y)$$

La primera implica la segunda y no al revés. Contraejemplo para el recíproco: dominio $\{a, b\}$ con $A(a, a)$ y $A(b, b)$ y nada más. Todos admiran a alguien —a sí mismos— y no hay nadie admirado por todos.

Ejercicio 7.3.2. Skolemizar $\forall x \, \exists y \, \forall z \, \exists w \; P(x, y, z, w)$.

Solución 7.3.2. y va tras $\forall x$, así que pasa a $f(x)$. w va tras $\forall x$ y $\forall z$, así que pasa a $g(x, z)$. Queda

$$\forall x \, \forall z \; P(x, f(x), z, g(x, z))$$

Cada función de Skolem depende exactamente de los universales que la preceden, ni uno más.

7.4 Unificación y resolución

Ejercicio 7.4.1. Unificar $P(f(x), y, g(y))$ con $P(f(a), z, g(z))$.

Solución 7.4.1. Primera discrepancia dentro de f: $\{x/a\}$. Segunda: y frente a z, que da $\{y/z\}$. Aplicándola, las terceras posiciones son $g(z)$ las dos. El umg es $\{x/a, \; y/z\}$.

Ejercicio 7.4.2. Probar por resolución que de «todos los perros son mamíferos» y «algún mamífero no es acuático» no se sigue «algún perro no es acuático».

Solución 7.4.2. No se puede probar porque no es válido: basta un contraejemplo semántico. Dominio $\{p, m\}$ con $Perro(p)$, $Mamífero(p)$, $Mamífero(m)$, $Acuático(p)$ y m no acuático. Las dos premisas son verdaderas y la conclusión falsa, ya que el único perro es acuático. La resolución no derivaría la cláusula vacía, aunque el hecho de no encontrarla no demuestre nada por sí solo: la validez en primer orden solo es semidecidible.

7.5 Inducción y recurrencia

Ejercicio 7.5.1. Demostrar que $\sum_{k=1}^{n} k^2 = \dfrac{n(n+1)(2n+1)}{6}$.

Solución 7.5.1. *Base*: con $n = 1$ los dos lados valen 1.

Paso: sumando $(n+1)^2$ a la hipótesis,

$$\frac{n(n+1)(2n+1)}{6} + (n+1)^2 = \frac{(n+1)\,[n(2n+1) + 6(n+1)]}{6} = \frac{(n+1)(2n^2+7n+6)}{6}$$

y como $2n^2+7n+6=(n+2)(2n+3)$, queda $\frac{(n+1)(n+2)(2n+3)}{6}$, que es la fórmula para $n+1$.

Ejercicio 7.5.2. Resolver $a_n = 4a_{n-1} - 4a_{n-2}$ con $a_0 = 1$ y $a_1 = 3$.

Solución 7.5.2. $r^2-4r+4=(r-2)^2$: raíz doble $r=2$. La solución general lleva el factor n: $a_n=(A+Bn)2^n$. De $a_0=A=1$ y $a_1=2(1+B)=3$ sale $B=1/2$.

$$a_n = \left(1+\frac{n}{2}\right)2^n = 2^n + n\,2^{n-1}$$

Comprobación: $a_2 = 4+4 = 8$, y por la recurrencia $4\cdot 3 - 4\cdot 1 = 8$.

Ejercicio 7.5.3. Un algoritmo divide el problema en tres subproblemas de tamaño $n/2$ y combina en tiempo lineal. ¿Cuál es su coste?

Solución 7.5.3. $T(n) = 3T(n/2) + \Theta(n)$. Aquí $\log_2 3 \approx 1{,}585 > 1$, así que $f(n)=n$ está por debajo y se aplica el primer caso: $T(n) = \Theta(n^{\log_2 3})$. Es exactamente el coste del algoritmo de Karatsuba para multiplicar enteros grandes.

7.6 Grafos y árboles

Ejercicio 7.6.1. ¿Cuántos lados tiene un grafo 4-regular con 12 vértices? ¿Puede ser plano?

Solución 7.6.1. Por el apretón de manos, $2m = 12\cdot 4 = 48$, luego $m=24$. La cota de planaridad da $3n-6 = 30 \geq 24$, así que el criterio no lo descarta y podría ser plano. La cota es condición necesaria y no suficiente: para decidirlo haría falta Kuratowski o exhibir un dibujo.

Ejercicio 7.6.2. Determinar el número cromático de C_7 y de $K_{3,4}$.

Solución 7.6.2. C_7 es un ciclo de longitud impar, así que $\chi = 3$: con dos colores alternos, al cerrar el ciclo el primero y el último vértice coincidirían. $K_{3,4}$ es bipartido y tiene lados, luego $\chi = 2$, uno por clase, sin importar cuántos lados haya.

Ejercicio 7.6.3. Un árbol tiene 4 vértices de grado 3, 2 de grado 2 y el resto son hojas. ¿Cuántos vértices tiene?

Solución 7.6.3. Sea h el número de hojas y $n = 4+2+h$ el total. Un árbol tiene $n-1$ lados, así que la suma de grados es $2(n-1)$:

$$4\cdot 3 + 2\cdot 2 + h = 2(6+h-1)$$

es decir $16+h = 10+2h$, de donde $h=6$ y $n=12$.

Ejercicio 7.6.4. Un grafo conexo tiene todos sus vértices de grado par. ¿Se puede recorrer pasando por cada lado exactamente una vez y volviendo al punto de partida?

Solución 7.6.4. Sí, por el teorema de Euler: un grafo conexo tiene circuito euleriano si y solo si todos los grados son pares. La intuición del porqué: al entrar en un vértice hay que poder salir, y cada visita consume dos lados, así que un grado impar dejaría un lado sin pareja.

Los problemas siguen el estilo de Lipschutz, 2004 y Hortalá et al., 2008, con la teoría de Grimaldi, 1997, Biggs, 1998 y García Miranda, 2017.

Bibliografía

Biggs, N. L. (1998). *Matemática Discreta*. Vicens Vives.

Chang, C.-L., & Lee, R. C.-T. (1973). *Symbolic Logic and Mechanical Theorem Proving*. Academic Press.

García Miranda, J. (2017). *Lógica para informáticos y otras herramientas matemáticas*. Flemming.

Grimaldi, R. P. (1997). *Matemática discreta y combinatoria*. Addison-Wesley.

Gunderson, D. S. (2016). *Handbook of Mathematical Induction: Theory and Applications*. CRC Press.

Hortalá, T., Martí, N., Palomino, M., Rodríguez, M., & del Vado, R. (2008). *Lógica matemática para informáticos. Ejercicios resueltos*. Prentice Hall Pearson.

Jiménez Murillo, J. A. (2015). *Matemáticas para informática*. Marcombo.

Lipschutz, S. (2004). *2000 problemas resueltos de matemática discreta*. McGraw-Hill.

Paniagua, E., Sánchez González, J. L., & Martín Rubio, F. (2003). *Lógica computacional*. Paraninfo.

Permingeat, N., & Glaude, D. (1992). *Álgebra de Boole: teoría, métodos de cálculo y aplicaciones*. Vicens Vives.

Rosen, K. H. (2003). *Matemática discreta y sus aplicaciones*. McGraw-Hill.