Apuntes: Lógica y Matemática Computacional

Contenido

UNIDAD 1: Lógica de Proposiciones

Contenido: Sintaxis y semántica de fórmulas proposicionales. Interpretaciones. Fórmulas satisfacibles. Modelos. Tautologías. Contradicciones. Implicación lógica. Consecuencia lógica. Teorema de deducción. Conectivos adecuados. Reglas de prioridad. Circuitos lógicos. Reglas de inferencia. Modus ponens. Modus tollens. Silogismo hipotético. Aplicaciones.

La palabra lógica tiene sus raíces en Grecia, derivando de *logos*, que significa *palabra, tratado, idea, principio, pensamiento o razón*; y del término *ica* que significa, *relacionado a* o *relacionado con*; entonces la lógica puede ser asociada con *lo relacionado al pensamiento* o *a la razón*.

Proposiciones

Las proposiciones son oraciones o afirmaciones que tienen un único valor de verdad: o son verdaderas o son falsas. Son proposiciones: “el triángulo es un polígono de tres lados”; “Avatar y Titanic fueron dirigidas por James Cameron”; “2010 es un año bisiesto”; “El matemático G. Boole falleció en 1964”.

Las proposiciones son oraciones que asumen alguno de estos dos valores: verdadero o false (pero no ambos simultáneamente).

Proposiciones atómicas

Proposiciones como: “1 es un número primo”, “8 es múltiplo de 2”, “En el idioma español, las vocales son a, e, i, o, u”; se conocen como *primitivas o atómicas*, ya que no pueden descomponerse o subdividirse en partes más simples.

La Lógica Proposicional Clásica no asigna los valores de verdad a cada proposición atómica. Éstas ya tienen un valor de verdad asignado cuando se las analiza lógicamente.

Proposiciones compuestas

A partir de las proposiciones primitivas podemos obtener nuevas proposiciones, llamadas *compuestas*, combinándolas por medio de operadores lógicos llamados conectivos. Una propiedad básica es que la verdad o falsedad de una proposición compuesta depende de la verdad o la falsedad de las proposiciones primitivas o atómicas que la componen.

Fórmulas proposicionales

La LPC de un modelo matemático para los razonamientos que pueden hacerse en base a las distintas formas de operar con las proposiciones. Para ello cuenta con dos aspectos:

La sintaxis o gramática, que especifica qué secuencia de símbolos se consideran fórmulas bien formadas; y

La semántica, que permite interpretar las fórmulas y asignarles su valor de verdad o falsedad.

Un lenguaje formal es definido a partir de un conjunto (finito o infinito numerable) de símbolos y de reglas que permiten operar esos símbolos.

Símbolo: son símbolos, a, b, /, 0, 3, *, i.

Alfabeto: conjunto no vacío o finito de símbolos; por ejemplo, son alfabetos $\Sigma_1 = \{0, 1, *, \wedge\}$ o $\Sigma_2 = \{a, b, c\}$ (Σ se lee sigma).

Cadenas de símbolos o palabras: son las formadas con los símbolos del alfabeto. Éstas contienen un número finito de símbolos del alfabeto.

Gramática formal (o sintaxis): se define como el conjunto de reglas que permiten formar las palabras del lenguaje.

Fórmula bien formada (palabra, fórmula bien definida o simplemente fórmula): Cadena de símbolos formada de acuerdo a una gramática dada.

Hablar de un lenguaje formal definido a partir de los símbolos de un alfabeto, es análogo a hablar del conjunto de todas sus fórmulas bien formadas. A diferencia de lo que ocurre con el alfabeto (que debe ser un conjunto finito) y con cada fórmula bien formada (que debe tener una longitud finita), un lenguaje formal puede estar compuesto por un número infinito de fórmulas bien formadas.

Ejemplo

Alfabeto: $\Sigma_2 = \{a, b, c\}$

Gramática: identifica como fbf a aquellas que utilizan cantidades iguales de símbolos a y b.

Algunas fórmulas bien formadas del lenguaje son: ab, bca, c, cacbcacbc, cababbabac, bbbaaa, ccc, etc.

Un caso especial es la cadena vacía, simbolizada como λ (lambda) y no contiene símbolos del alfabeto.

Para algunos lenguajes formales existe una semántica formal que puede interpretar y dar significado a las fórmulas bien formadas del lenguaje. Sin embargo, una semántica formal no es condición necesaria para definir un lenguaje formal, y eso es una diferencia esencial con los lenguajes naturales.

Si el alfabeto es $\Sigma = \{p, q, \wedge, (,)\}$ y las únicas reglas de la sintaxis son:

i) p, q son fórmulas.
ii) Si m y n son fórmulas entonces $(m \wedge n)$ es una fórmula.

Entonces son fórmulas: p, q, $(p \wedge p)$, $(p \wedge q)$, $((p \wedge q) \wedge (q \wedge p))$, $((q \wedge q) \wedge q)$, etc.

La semántica permite dar significado a estas fórmulas. Podemos interpretarlas, y al interpretarlas les asignaremos un valor de verdad. Por ejemplo:

P: “2 es un número par”; q: “3 es número impar”: $\wedge$: “y”, luego interpretamos a $(p \wedge q)$ como “2 es un número par y 3 es número impar”. Esta interpretación convierte a p, q, $(p \wedge q)$ en proposiciones.

Sintaxis

Cada lenguaje formal tiene *símbolos propios* y una *sintaxis*, es decir una especificación rigurosa de las secuencias de símbolos que están permitidas. La sintaxis comienza con una especificación del alfabeto del lenguaje. Veamos a continuación las definiciones sintácticas correspondientes a la LPC.

El alfabeto Σ = {p, |, ¬, ∧, ∨, →, ↔, (,)} de la LPC consiste en un conjunto no vacío y finito de símbolos, que describimos a continuación:

i) Variables proposicionales: p, p|, p||, p|||, … Para mayor simplicidad las indicaremos como p_0 = p; p_1 = p|; p_2 = p||, etc o simplemente p, q, r, s, etc. Usaremos las últimas letras de nuestro abecedario para las variables proposicionales. Indicaremos con **Var** al conjunto de variables proposicionales.
ii) Conectivos lógicos: ¬, ∧, ∨, →, ↔.
iii) Símbolos auxiliares: (,). Estos símbolos se llaman de agrupación.

A cada posible conjunto de variables proposicionales corresponde un alfabeto diferente; es decir, las variables proposicionales son símbolos propios de cada alfabeto; pero cada alfabeto de la LPC tiene los mismos cinco conectivos y los mismos símbolos auxiliares; es decir, los conectivos y los símbolos auxiliares son símbolos comunes para todos los alfabetos de la LPC. Usando los símbolos del alfabeto, podemos formar secuencias de símbolos.

Fórmula bien formada

Una *fórmula bien formada* (o simplemente *fórmula*), que indicaremos con fbf, se define recursivamente como:

i) Una variable proposicional es una fbf.
ii) Si θ (leemos theta o tita) es una fbf, ¬θ es una fbf.
iii) Si θ y ϕ (leemos phi, fi) son fórmulas bien formadas (en adelante, fbfs) entonces también son fbfs (θ ∧ ϕ), (θ ∨ ϕ), (θ → ϕ) y (θ ↔ ϕ).
iv) Una sucesión o cadena de variables proposicionales, conectivos lógicos o símbolos del alfabeto es una fbf si y sólo si puede obtenerse mediante un número finito de aplicaciones de las reglas i), ii), iii).

Indicaremos con **Form** al conjunto de todas las fórmulas.

Ejemplo: ¬, ∧, ∨, →, ↔.

Si el conjunto de variables proposicionales es: Var = {p, q, r}, entonces son fórmulas bien formadas p, ¬q, ((p ∨ q) → r), (p ∧ (q ∨ r)), entre otras. La fórmula más simple es un átomo. Las variables proposicionales son átomos y son fbfs. Las fórmulas que no son átomos, por ejemplo ¬p, (p ∨ q), se llaman fórmulas compuestas.

Lenguaje proposicional

El lenguaje proposicional **L** dado por el alfabeto Σ es el conjunto de todas las fórmulas (bien formadas) que pueden construirse partiendo de los símbolos del alfabeto. Esta definición justifica que identifiquemos al lenguaje L con Form, que es el conjunto de sus fórmulas.

Si el alfabeto Σ_1 y el alfabeto Σ_2 son diferentes, es decir, si el conjunto de átomos pertenecientes a Σ_1 es diferente del conjunto de átomos pertenecientes a Σ_2, luego el lenguaje proposicional dado por Σ_1 es diferente del lenguaje proposicional dado por Σ_2. Podemos observar que un lenguaje proposicional es siempre un conjunto infinito, aún cuando el conjunto de variables proposicionales del alfabeto pueda ser unitario.

Semántica

La semántica es el significado que adquiere una fórmula. Trataremos a las fórmulas como proposiciones, como enunciados a los que se le puede asignar uno de los valores de verdad “verdadero” o “falso”. Al referirnos a “proposiciones” haremos referencia a fórmulas con significado.

Una fórmula puede ser o verdadera o falsa, dependiendo de la verdad o falsedad de las fórmulas más simples que son sus componentes (átomos).

Podemos determinar su valor de verdad recurriendo a alguna interpretación de sus átomos. Veremos cómo las interpretaremos.

Tabla de verdad

Es una tabla que ayuda a la comprensión de lo expresado por los valores de verdad de una proposición, donde escribimos 0 para indicar que la proposición es falsa y 1 para indicar que es verdadera.

Negación

Consideramos una fórmula de la forma ¬θ, donde θ es una fórmula arbitraria. El conectivo usado "¬" se llama negación.

Decimos que la fórmula ¬θ (se lee no tita) es verdadera si la fórmula θ es falsa y que la fórmula ¬θ es falsa si la fórmula θ es verdadera.

Tabla de verdad:

θ	¬θ
0	1
1	0

Este conectivo se una para modelar la palabra "no" de nuestro lenguaje cotidiano.

Conjunción

Consideramos la fórmula de la forma (θ ∧ ϕ), siendo θ y ϕ fórmulas arbitrarias. El conectivo "∧" se llama *conjunción* y la fórmula (θ ∧ ϕ) se lee θ y ϕ (tita y fi).

Una fórmula del tipo (θ ∧ ϕ), es verdadera sólo cuando ambas, "θ" y "ϕ" son verdaderas. Esto es análogo al uso de la palabra "y" del lenguaje usual.

Tabla de verdad:

θ	ϕ	(θ ∧ ϕ)
0	0	0
0	1	0
1	0	0
1	1	1

Disyunción

Disyunción incluyente

Consideramos la fórmula de la forma (θ ∨ ϕ), siendo θ y ϕ fórmulas arbitrarias. El conectivo "∨" se llama *disyunción (incluyente)* y la fórmula (θ ∨ ϕ), se lee θ o ϕ (tita o fi).

La fórmula del tipo (θ ∨ ϕ), es falsa cuando ambas, "θ" y "ϕ" son falsas. Esto es análogo al uso de la palabra "o" del lenguaje usual.

Tabla de verdad:

θ	ϕ	(θ ∨ ϕ)
0	0	0
0	1	1
1	0	1
1	1	1

Disyunción excluyente

Consideramos la fórmula de la forma (θ $\underline{\vee}$ ϕ), siendo θ y ϕ fórmulas arbitrarias. El conectivo "$\underline{\vee}$" se llama *disyunción exclusiva* o *diferencia simétrica*, que leeremos como o θ o ϕ (o teta o fi).

La fórmula del tipo (θ $\underline{\vee}$ ϕ), es verdadera si una o si la otra, pero ambas fórmulas son verdaderas.

Tabla de verdad:

θ	ϕ	$(\theta \underline{\vee} \phi)$
0	0	0
0	1	1
1	0	1
1	1	0

Implicación

En la fórmula de la forma (θ → ϕ), siendo θ y ϕ fórmulas arbitrarias, encontramos el conectivo "→" que se llama *implicación* y la fórmula (θ → ϕ) se lee si θ entonces ϕ, si θluego ϕ o bien, ϕ si θ.

Es verdadera en todos los casos, excepto donde "θ" es verdadera y "ϕ" es falsa. (No se pretende que una fórmula verdadera implique una fórmula falsa).

Podemos observar que la fórmula de la forma (θ → ϕ) es verdadera toda vez que θ sea falsa, sin importar la interpretación de ϕ. La fórmula "θ" se denomina *hipótesis, antecedente o premisa* de la implicación, y "ϕ" se denomina *conclusión o consecuente* de la implicación.

Tabla de verdad:

θ	ϕ	(θ → ϕ)
0	0	1
0	1	1
1	0	0
1	1	1

Si el antecedente es verdadero, entonces el valor de verdad del condicional es igual al valor de verdad del consecuente y si el antecedente es falso el valor del condicional es verdadero.

Implicaciones asociadas

Sean θ y ϕ fórmulas. Si llamamos a la fórmula (θ → ϕ) como *implicación directa,* la fórmula (ϕ → θ) se llama *recíproca* de (θ → ϕ); la implicación (¬θ→¬ϕ) es su *contraria*, mientras que (¬ϕ→¬θ) es su *contra recíproca*. Se dice que las implicaciones recíprocas, contraria y contra recíproca son implicaciones asociadas a la directa.

Doble implicación

En la fórmula de la forma (θ ↔ ϕ), siendo θ y ϕ fórmulas arbitrarias, el conectivo ↔ se llama *equivalencia, bicondicional o doble implicación* y la fórmula se lee θ sí y sólo si ϕ, o bien, θ es equivalente a ϕ.

La fórmula de la forma (θ ↔ ϕ) es verdadera precisamente cuando "θ" y "ϕ" tienen el mismo valor de verdad.

Tabla de verdad:

θ	ϕ	(θ ↔ ϕ)
0	0	1
0	1	0
1	0	0
1	1	1

Equivalencias lógicas:

- Leyes de identidad: $p \wedge \boldsymbol{V} \equiv p$ $\quad p \vee \boldsymbol{F} \equiv p$
- Leyes de dominación: $p \vee \boldsymbol{V} \equiv \boldsymbol{V}$ $\quad p \wedge \boldsymbol{F} \equiv \boldsymbol{F}$
- Leyes idempotentes: $p \wedge p \equiv p$ $\quad p \vee p \equiv p$
- Ley de la doble negación: $\neg(\neg p) \equiv p$
- Leyes conmutativas: $p \wedge q \equiv q \wedge p$ $\quad p \vee q \equiv q \vee p$
- Leyes asociativas: $(p \vee q) \vee r \equiv p \vee (q \vee r)$ $\quad (p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$
- Leyes distributivas: $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ $\quad p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
- Leyes de De Morgan: $\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\quad \neg(p \vee q) \equiv \neg p \wedge \neg q$
- Leyes de absorción: $p \vee (p \wedge q) \equiv p$ $\quad p \wedge (p \vee q) \equiv p$
- Leyes de negación: $p \vee \neg p \equiv \boldsymbol{V}$ $\quad p \wedge \neg p \equiv \boldsymbol{F}$

Equivalencias lógicas relacionadas con implicaciones:

- Implicación como disyunción: $p \rightarrow q \equiv \neg p \vee q$
- Disyunción como implicación: $p \vee q \equiv \neg p \rightarrow q$
- Conjunción como implicación: $p \wedge q \equiv \neg(p \rightarrow \neg q)$
- Negación de una implicación: $\neg(p \rightarrow q) \equiv p \wedge \neg q$
- Conjunción de implicaciones: $(p \rightarrow q) \wedge (p \rightarrow r) \equiv p \rightarrow (q \wedge r)$
  $(p \rightarrow r) \wedge (q \rightarrow r) \equiv (p \vee q) \rightarrow r$
- Disyunción de implicaciones: $(p \rightarrow q) \vee (p \rightarrow r) \equiv p \rightarrow (q \vee r)$
  $(p \rightarrow r) \vee (q \rightarrow r) \equiv (p \wedge q) \rightarrow r$

Equivalencias lógicas relacionadas con doble implicación:

- $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$
- $p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$
- $p \leftrightarrow q \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$
- $\neg(p \leftrightarrow q) \equiv p \leftrightarrow \neg q$

Interpretación

Un concepto fundamental en semántica es el de *Interpretación*. Cada interpretación define qué fórmulas del lenguaje son falsas y cuáles son verdaderas. Los valores de verdad son dos: falso o verdadero, y pueden ser indicados con 0 o 1; off u on; no o si, F o T, respectivamente.

Es decir, a partir de una fórmula proposicional compuesta, la interpretación define qué fórmulas atómicas (proposición), del lenguaje de proposiciones, son **falsas** (**0**) y qué fórmulas son **verdaderas** (**1**).

Sea Var el conjunto de átomos del lenguaje L.

Una *interpretación* es una función de Var = $\{p1, p2, p3, ...\}$ en $\{0, 1\}$, que asigna a cada elemento de Var un 0 o un 1.

Una interpretación puede ser explícitamente identificada por el subconjunto de átomos de Var que tienen como imagen 1. Bajo esta representación cada interpretación se identifica con los átomos de Var a los que se ha asignado un 1.

Ejemplo. Si Var = {p, q, r}, una interpretación I_1 puede estar dada por I_1= {p}, esto significa que p tiene como imagen 1 mientras que q y r tienen como imagen 0. Otro ejemplo es I_2 = {p, q}, donde p y q tienen como imagen

1 mientras que r tiene como imagen 0, es decir $I_2(p)=1$; $I_2(q)=1$; $I_2(r)=0$; una nueva valuación, es I_3 = {p, q, r} donde todas las variables proposiciones tienen como imagen el 1.

Si el conjunto de átomos es n, entonces será posible definir 2^n interpretaciones.

Valuación

Sea I una interpretación. Sea Form el conjunto de fórmulas de L. Una *valuación* bajo una interpretación I, es cualquier función v_I de Form en {0, 1}, que satisface las siguientes reglas:

v.1) $v_I(p_n) = I(p_n)$ para cada variable proposicional. Es decir, cuando la fórmula es una variable proposicional, su valuación coincide con la asignación que le hace la interpretación.

v.2) Siendo ϕ y ρ fórmulas arbitrarias, se define la valuación de una fórmula θ como sigue:

ϕ	ρ	$\theta = \neg\phi$	$\theta = (\phi \wedge \rho)$	$\theta = (\phi \vee \rho)$	$\theta = (\phi \rightarrow \rho)$	$\theta = (\phi \leftrightarrow \rho)$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Dicho de otra manera:

Se define el valor de verdad de una fórmula θ bajo I como una función *valuación* $v_I(\theta)$ de la siguiente manera:

a) Si θ es una variable proposicional entonces el valor de verdad de θ, $v_I(\theta)$, es la imagen (0 o 1) que le corresponde a la variable proposicional bajo la interpretación I.

b) Si θ es de la forma $\neg\phi$ entonces $v_I(\theta) = 1$ si $v_I(\phi) = 0$ y $v_I(\theta) = 0$ si $v_I(\phi) = 1$.

c) Si θ es de la forma $(\phi \wedge \rho)$ entonces $v_I(\theta) = 1$ si $v_I(\phi)= 1$ y $v_I(\rho) = 1$ y $v_I(\theta) = 0$ en cualquier otro caso.

d) Si θ es de la forma $(\phi \vee \rho)$ entonces $v_I(\theta) = 0$ si $v_I(\phi)= 0$ y $v_I(\rho) = 0$ y $v_I(\theta) =1$ en cualquier otro caso.

e) Si θ es de la forma $(\phi \rightarrow \rho)$ entonces $v_I(\theta) = 0$ si $v_I(\phi) = 1$ y $v_I(\rho) = 0$ y $v_I(\theta) =1$ en

cualquier otro caso.

f) Si θ es de la forma $(\phi \leftrightarrow \rho)$ entonces $v_I(\theta) = 1$ si $v_I(\phi) = v_I(\rho)$ y $v_I(\theta) = 0$ en cualquier

otro caso.

Formula Satisfacible

Sea θ una fórmula y sea I una interpretación.

Se dice que θ es verdadera bajo I si su valor de verdad es 1. Cuando θ es verdadera bajo I, se dice que I *satisface* a θ; o que θ es *satisfacible* por I. Escribiremos que I satisface a θ como $I \models \theta$.

Se dice que θ es falsa bajo I si su valor de verdad es 0. Cuando θ es falsa bajo I, se dice que I *no satisface* a θ.

Modelo. Tautología. Contradicción

Modelo

Sea θ una fórmula y sea I una interpretación. I es un *modelo* para θ sí I *satisface* a θ; se dice también que θ tiene a I como *modelo*.

Tautología

Una fórmula θ se llama *tautología* si todas las interpretaciones son modelos para la fórmula. Se utiliza el símbolo T_0 para denotar cualquier tautología. Cuando una fórmula es tautología vamos a anotar |= θ.

- Las equivalencias lógicas: se utilizan como esquemas de sustitución en procesos de razonamiento. $\sim(p \vee q) \leftrightarrow (\sim p \wedge \sim q)$
- Las implicaciones lógicas: se utilizan como esquemas de razonamientos válidos. $(p \wedge q) \vee r \rightarrow (p \vee r) \wedge (q \vee r)$

Contradicción

Una fórmula se llama *contradicción*, *insatisfacible* o *inconsistente* si es falsa para todas las interpretaciones posibles. Se utiliza F_0 para designar toda contradicción. Una contradicción no tiene interpretaciones que sean modelos.

Contingencia

Una proposición lógica que no es una *tautología* ni es una *contradicción* se denomina *contingencia*. Una contingencia es satisfacible pero no es tautología. Esto significa que existirá alguna interpretación I para la cual $v_I(\theta) = 1$ y alguna interpretación J donde $v_J(\theta) = 0$.

Modelo para conjunto de fórmulas

Sea S un conjunto de fórmulas, S ⊆ Form y v una valuación. Diremos que una valuación v satisface un conjunto S de fórmulas, si v(ϕ)=1 para toda ϕ ∈S. Es decir, la valuación es modelo para S si es modelo para todas las fórmulas de S.

Conjunto de fórmulas satisfables

Un conjunto de fórmulas se dice satisfacible si existe al menos una valuación que satisfaga a todas las fórmulas del conjunto e insatisfacible en caso contrario.

Propiedad

Un conjunto finito y no vacío de fórmulas es satisfacible si y sólo si la conjunción de todas las fórmulas del conjunto es satisfacible.

Implicación lógica y consecuencia lógica

Sea S un conjunto de fórmulas y ϕ una fórmula.

Se dice que ϕ es una *consecuencia lógica* de S si toda interpretación que es un modelo de S es también un modelo para ϕ. Podemos escribir esto como S|= ϕ. A veces diremos que S *implica lógicamen*te a ϕ o simplemente que S *implica* a ϕ.

Dicho de otra manera, ϕ es *consecuencia lógica* de S si toda interpretación I que hace verdadera a todas las fórmulas de S, también hace verdadera a ϕ.

Es decir, si v(θ) = 1 para toda θ ∈ S, entonces v(ϕ) = 1.

Indicaremos Conj S = {ϕ ∈ Form / S|= ϕ} al conjunto de las consecuencias de S. El conjunto Conj S puede ser infinito.

En particular si S = {θ} podemos escribir θ|= ϕ o bien θ ⇒ ϕ y leemos θ implica (lógicamente) a ϕ, en cualquier caso.

Definición:

Sean S_1 y S_2 dos conjuntos de fórmulas. Luego S_2 es *consecuencia lógica* de S_1 y escribimos $S_1 \models S_2$ si $S_1 \models \phi$ para toda fórmula $\phi \in S_2$. También podemos decir que S_1 implica (lógicamente) a S_2.

Teorema de Deducción

Sea S un conjunto de fórmulas y θ y ϕ fórmulas arbitrarias.

Luego S∪{θ} |= ϕ si y sólo si S |= (θ→ϕ). También podemos enunciarlo así: ϕ ∈ Con(S∪{θ}) si y sólo si (θ→ϕ) ∈ Con S.

Observación:

Como corolario del teorema de la deducción, surge que para todo par de fórmulas θ, ϕ se tiene que {θ}| = ϕ si y sólo si |= (θ→ϕ), es decir, si y sólo si (θ→ϕ) es una tautología. O bien, ϕ ∈ Con {θ} si y sólo si (θ→ϕ) ∈ Con {} si y sólo si (θ→ϕ) es tautología.

TEOREMA

Para todo conjunto de fórmulas S, S ⊆ Form y para toda fórmula θ, se tiene que S |= θ sí y sólo si S∪{¬θ} es insatisfacible.

Equivalencia Lógica

Dos fórmulas θ y ϕ son *lógicamente equivalentes* si tienen los mismos modelos, es decir, si para toda interpretación I, los valores de verdad coinciden; es decir $v_I(\theta) = v_I(\phi)$ para toda

interpretación.

También podemos decir que dos fórmulas θ y ϕ son *lógicamente equivalentes* si y sólo se verifican θ |= ϕ y ϕ |= θ. Se expresa θ ⇔ ϕ o bien θ ≡ ϕ y se lee θ es equivalente a ϕ o bien θsí y sólo si ϕ.

Similarmente dos conjuntos de fórmulas S_1 y S_2 se dicen *lógicamente equivalentes* si y sólo si $S_1 \models S_2$ y además $S_2 \models S_1$.

TEOREMA

Si θ y ϕ son fórmulas arbitrarias, entonces θ ⇔ ϕ si y sólo si |= (θ ↔ ϕ).

Conectivos adecuados

Podemos expresar que dado un conjunto de conectivos, se dice que es *adecuado* sí a partir de sus elementos pueden definirse todos los demás conectivos.

El conjunto {¬, ∧, ∨} es adecuado.

En consecuencia, toda fórmula del LPC puede ser construida a partir de este conjunto adecuado de conectivos.

En efecto (θ → ϕ) ≡ (¬θ ∨ ϕ); (θ ↔ ϕ) ≡ (θ → ϕ) ∧ (ϕ → θ) ≡ (¬θ ∨ ϕ) ∧ (¬ϕ ∨θ)

Es posible demostrar que {¬, ∧}, {¬, ∨}, {¬, →} son conjuntos de conectivos adecuados y que no lo son {¬}, {∧, ∨}, {→, ∨}.

Reglas de prioridad

Estas reglas establecen una prioridad para los conectivos, lo que nos permiten reconstruir unívocamente a toda fbf que se le hayan suprimido los paréntesis no necesarios, de acuerdo a estas reglas.

La conexión ¬ (operador unario) tiene siempre la prioridad más alta.

Para los restantes conectivos (operadores binarios), la prioridad más alta se le da a ∧, seguida por ∨, →, ↔ en ese orden.

Es decir, el orden de prioridad de mayor a menor sería: ¬, ∧, ∨, →, ↔.

Ejemplo: $(p \rightarrow (q \wedge t)) \leftrightarrow t \Leftrightarrow p \rightarrow q \wedge t$

Al decir prioridad más alta hacemos referencia a que, si no aparecen paréntesis, inferimos que en la construcción recursiva de la fórmula primero se negaron todas las variables proposicionales a las que antecede el conectivo ¬, después se realizaron todas las conjunciones que aparecen indicadas, luego las disyunciones, luego las implicancias y por último las equivalencias.

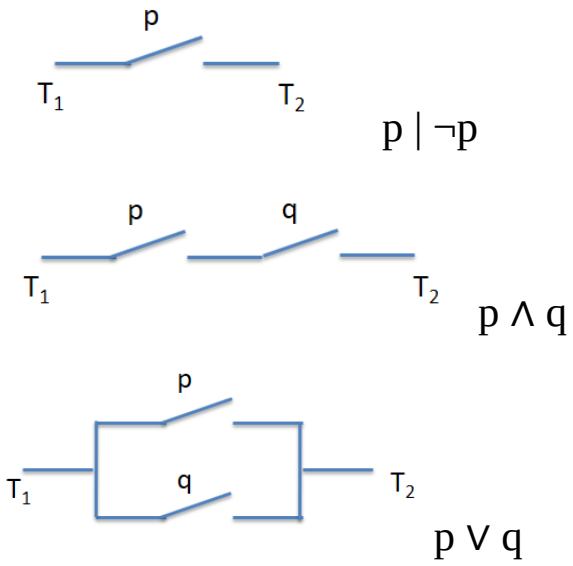
Si en una expresión los conectivos tienen el mismo orden de precedencia, la evaluación de tales expresiones se realiza de izquierda a derecha. Por ejemplo, la expresión $p \rightarrow q \rightarrow r$ hace referencia a $((p \rightarrow q) \rightarrow r)$.

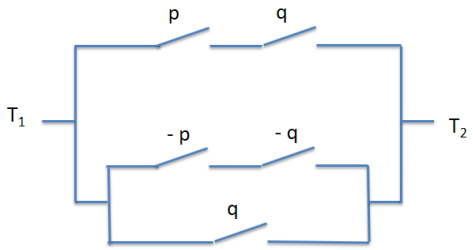
Circuitos lógicos o red de conmutación

Las operaciones proposicionales pueden representarse mediante circuitos con tantos interruptores como proposiciones, combinados en serie o paralelamente.

Una red de conmutación o circuito lógico está formada por cables e interruptores que conectan dos terminales T1 y T2. Si un interruptor está abierto, entonces no pasa la corriente por él y lo indicaremos con un "0", mientras que un "1" indicará que el interruptor está cerrado y por consiguiente pasa la corriente por él.

Los circuitos lógicos también se pueden simplificar. Esto significa analizar la posibilidad de obtener otra red equivalente a la dada con menor cantidad de interruptores y/o conexiones.





$$(p \land q) \lor ((\neg p \land \neg q) \lor q)$$

Reglas de inferencia

La función principal de la lógica de proposiciones es proporcionar reglas de inferencia o razonamientos Un razonamiento es básicamente una implicación lógica.

El antecedente de la implicación es una conjunción de fbfs que expresan un conocimiento ya obtenido; cada fbf es una *premisa*. El consecuente de la implicación es la *conclusión* y se obtiene a partir de la conjunción de las premisas, como un conocimiento nuevo.

Definición:

El *razonamiento* es una implicación cuyo antecedente es la conjunción de un número finito de proposiciones llamadas *premisas* y cuyo consecuente es la *conclusión*.

Tanto las premisas como la conclusión son fbfs y consecuentemente un razonamiento es una fbf. Simbolizamos las premisas con p1, p2, p3, ..., p_n y la conclusión con la letra c.

Entonces, en símbolos, un razonamiento es expresado utilizando la implicación o condicional, en estos términos: $p1 \land p2 \land p3 \land ... \land p_n \rightarrow c$.

Validez de un razonamiento

La validez de un razonamiento hace referencia a cuáles son las condiciones para que la conclusión se deduzca de una o más proposiciones. La corrección (validación) o no de un razonamiento se decide a partir de la suposición de que las premisas son verdaderas.

La lógica nos permite discriminar entre razonamientos correctos o válidos e incorrectos o no válidos.

Un razonamiento es válido cuando el condicional es tautológico, esto es, si las premisas son verdaderas entonces la conclusión debe ser necesariamente verdadera y si las premisas son falsas no importará lo que ocurre con la conclusión ya que el condicional será verdadero y por consiguiente el razonamiento será válido.

En términos lógicos, un razonamiento válido tiene esta forma: $p1 \land p2 \land p3 \land ... \land p_n \Rightarrow c$

Un razonamiento válido también se denomina *regla de inferencia*.

Cuando una conclusión se obtiene a partir de un conjunto de premisas, usando reglas aceptadas de razonamiento, entonces tal proceso de derivación se conoce como ***deducción o prueba formal***. En una prueba formal, cada regla de inferencia que se usa en alguna etapa de la deducción, se manifiesta.

A continuación, estudiaremos en detalle algunas de las reglas de inferencia que se aplican habitualmente:

- **Modus Ponens: $p \land (p \rightarrow q) \rightarrow q$.**
  p: Juan es correntino. q: Juan es argentino. Juan es correntino **y si** Juan es correntino **entonces** Juan es argentino; **luego** Juan es argentino.
- **Modus Tollens: $(p \rightarrow q) \land \sim q \rightarrow \sim p$.**
  p: aprobé el examen. q: te presto los apuntes. Si aprobé el examen **entonces** te presto los apuntes **y** no te presto los apuntes. **Luego**, no aprobé el examen.

- **Silogismo Hipotético: $(p \rightarrow q) \wedge (q \rightarrow r) \rightarrow (p \rightarrow r)$.**
  p: hago mi trabajo más eficiente. q: gano más. r: mejora mi nivel de vida. **Si** hago mi trabajo más eficiente **entonces** gano más **y si** gano más **entonces** mejora mi nivel de vida. **Luego: si** hago mi trabajo más eficiente, **entonces** mejora mi nivel de vida.
- Silogismo Disyuntivo:
  - **Disyunción incluyente y excluyente: $(p \vee q) \wedge \sim p \rightarrow q$**
    p: compramos un departamento, q: cambiamos el auto. Compramos un departamento **o** cambiamos el auto **y** no compramos un departamento. **Luego**: cambiamos el auto.
  - **Disyunción excluyente: $(p \Delta q) \wedge p \rightarrow \sim q$**
    p: compramos un departamento, q: cambiamos el auto. **O** compramos un departamento **o** cambiamos el auto **y** compramos un departamento, **luego, NO** cambiamos el auto.
- **Dilema Constructivo: $((p \vee q) \wedge (p \rightarrow r) \wedge (q \rightarrow r)) \rightarrow r$**
  p: viajo a Salta, q: viajo a Córdoba, r: voy a descansar. Viajo a Salta **o** viajo a Córdoba **y, si** viajo a Salta **entonces** voy a descansar, **y, si** viajo a Córdoba **entonces** voy a descansar; **luego,** voy a descansar.
- **Dilema Destructivo: $(p \rightarrow q) \wedge (r \rightarrow s) \wedge (\sim q \vee \sim s) \rightarrow (\sim p \vee \sim r)$**
  p: viajo a Salta, q: viajo a Córdoba, r: voy a descansar, s: me siento mejor. **Si** viajo a Salta **entonces** viajo a Córdoba **y si** voy a descansar **entonces** me siento mejor, **y si** no viajo a Córdoba **entonces** no me siento mejor. **Luego**, no viajo a Salta o no voy a descansar.

UNIDAD II: Relaciones de recurrencia

Contenido: Sucesiones y sumatorias. Definiciones por recurrencia. Relaciones de recurrencia. Clasificación de las relaciones de recurrencia. Resolución de los diferentes tipos de relaciones lineales – no lineales – homogéneas – no homogéneas, con coeficientes constantes – con coeficientes variables. Generación de números (pseudo) aleatorios.

Sucesiones y sumatorias

Una sucesión es una lista ordenada de objetos o una colección de objetos dispuestos en un determinado orden. Por lo tanto, existe un elemento que está en primer lugar, otro en segundo lugar y así sucesivamente. A dichos elementos se los denomina los "términos de la sucesión".

Tipos:

a) Si la lista tiene un número finito de elementos, se dice que la sucesión es finita.
b) Si la lista continúa indefinidamente, se dice que la sucesión es infinita.

Una sucesión es una función cuyo dominio va de Naturales a los Reales; si va de N a Z, entonces se llama *sucesión entera*.

Dada una sucesión denotada por *S*, S_n detona la sucesión completa a_1, a_2, a_3… etc.

S: $a_1, a_2, a_3, \ldots, a_n$ o *S:* $\{a_n : n \in N\}$ o *S:* $\{a_n\}$

Los elementos $a_1, a_2, a_3, \ldots$ son *términos* de la sucesión.

a_1 es el primer término de la sucesión, a_2 es el segundo término de la sucesión y así siguiendo hasta a_n que es el término *n-ésimo* de la sucesión.

Relaciones de recurrencia

Una relación de recurrencia es una ecuación que permiten obtener cada uno de los términos de la sucesión en función de uno o más elementos anteriores a él. Será necesario conocer al menos el primer término de la sucesión.

Definición matemática

Dada una sucesión *S: a_1, a_2, a_3, …, a_n,* el término a_n puede ser expresado en función de los términos anteriores a_{n-1}, a_{n-2}, a_{n-3}, …, a_1; luego, la expresión obtenida es una ***relación de recurrencia*** y se puede expresar como:

$$a_n = F\ (a_{n-1},\ a_{n-2},\ a_{n-3},\ \ldots,\ a_1)$$

En general, para poder calcular los términos de una sucesión, es necesario conocer al menos un término de la misma.

Condiciones iniciales o de frontera

Son los valores iniciales que se necesitan para resolver la relación de recurrencia, se denotan como a_0 o a_1.

Sea *k* el entero menor para el cual tenemos asignados valores de a, a_1, a_2, …, a_k;

$a_n = F\ (a_{n-1},\ a_{n-2},\ \ldots,\ a_1)$, permite calcular valores únicos para a_n si $n > k$.

Los valores de a_1, a_2, …, a_k se llaman ***condiciones iniciales o condiciones de frontera*** de la relación.

Las condiciones iniciales con la relación de recurrencia generan unívocamente la sucesión.

Término general

Se llama término general a la ecuación que permite hallar el valor de a_n.

Dado S: 4, 7, 10, 13,… con n > 1 y $a_1 = 4$.

$a_1 = 4$

$a_2 = 7 = 4 + 3$

$a_3 = 10 = 7 + 3$

$a_n = a_{n-1} + 3$

Solución general

Se denomina como solución general de la relación de recurrencia a la fórmula explícita para a_n que permite calcular la expresión para cada valor de *n*, sin necesidad de conocer los términos previos.

Ejemplo:

$$\begin{cases} a_1 = 4, & n > 1 \\ a_n = a_{n-1} + 3 \end{cases}$$

$a_1 = 4$

$a_2 = a_1 + 3$

$a_3 = a_2 + 3 = a_1 + 3 + 3 = a_1 + 2 * 3 = 10$

$a_n = a_{n-1} + 3 = a_{n-2} + 3 + 3 = a_1 + 3 + \ldots + 3 = a_1 + (n - 1) * 3$

$a_n = a_1 + (n - 1) * 3$

Clasificación de las relaciones de recurrencia

1) Lineal o no Lineal:

- **Lineal:** cuando cada término con subíndice de la relación de recurrencia aparece elevado a la primera potencia, por ejemplo: $a_{n+1} = 3a_n, a_0 = 3, n \geq 0$

- **No Lineal:** cuando algún término con subíndice de la relación de recurrencia aparece elevado a una potencia diferente a la primera potencia, por ejemplo: $a_{n+1}{}^2 = 3a_n{}^2, a_0 = 3, n \geq 0$

2) Con coeficientes constantes o variables:

- **Coeficientes constantes:** cuando cada término con subíndice de la relación de recurrencia está multiplicado por una constante, por ejemplo: $a_{n+1} = 3a_n, a_0 = 3, n \geq 0$
- **Coeficientes variables:** cuando algún término con subíndice de la relación de recurrencia está multiplicado por un valor variable, por ejemplo: $a_n = na_{n-1}, a_0 = 1, n \geq 1$

3) Homogéneos o no homogéneos:

- **Homogénea:** cuando *f(n)* = 0 para todo $n \in N$, por ejemplo: $a_{n+1} = 3a_n \rightarrow a_{n+1} - 3a_n = 0, a_0 = 3, n \geq 0$
  O una relación de recurrencia es **homogénea** cuando la sucesión idénticamente nula la satisface. $a_n = 0$ *para todo n que satisface la relación*. $a_n + c_1a_{n-1} + c_1a_{n-2} + \cdots + c_1a_{n-k} = 0, n \geq k$
- **No Homogénea:** cuando $f(a) \neq 0$ para todo $n \in N$, por ejemplo: $a_{n+1} = 3a_n \rightarrow a_{n+1} - 3a_n = \boldsymbol{n}, a_0 = 3, n \geq 0$
  O una relación de recurrencia es **no homogénea** cuando la sucesión idénticamente nula no la satisface. $a_n + c_1a_{n-1} + c_1a_{n-2} + \cdots + c_1a_{n-k} = b_n, n \geq k$

4) Primer o Segundo Orden:

- **Primer Orden**: cuando la relación de recurrencia sólo depende de su predecesor inmediato, por ejemplo: $a_{n+1} = 3a_n, a_0 = 3, n \geq 0$
- **Segundo Orden**: cuando la relación de recurrencia depende de sus dos predecesores inmediatos, por ejemplo: $a_n = a_{n-1} + 5a_{n+2}, a_0 = 0\,, a_1 = 1, n \geq 2$

Resolución de los diferentes tipos de relaciones

Transformación de una Relación de Recurrencia No Lineal a Lineal

Se puede transformar una relación de recurrencia no lineal a lineal para resolverla mediante una sustitución algebráica $b_n = a_n{}^2$. Ejemplo:

$$a_{n+1}{}^2 = 3a_n{}^2, a_0 = 3, n \geq 0$$

$$b_{n+1} = b_n, b_0 = 9, n \geq 0$$

Una vez hecho esto procedemos a resolverla como una relación de recurrencia lineal. Para este ejemplo obtuvimos una relación de recurrencia de primer orden, homogénea y con coeficientes constantes.

Después de resolverla sacamos raíz a cada número obtenido de la solución general para tener la solución general de la relación de recurrencia no lineal. Ejemplo:

$$b_n = 9(3)^n, b_0 = 9, n \geq 0$$

$$a_n = 3(\sqrt{3})^n, a_0 = 3, n \geq 0$$

Solución de relaciones de recurrencia lineales, homogéneas con coeficientes constantes

Definición: Una relación de recurrencia es una relación lineal, homogénea con coeficientes constantes de orden *k* ($k \in N$) si es de la forma:

$$a_n + c_1a_{n-1} + c_2a_{n-2} + \cdots + c_ka_{n-k} = 0, n \geq k$$

$c_1, c_2, \ldots, c_k$ son constantes reales y $c_k \neq 0$

A partir de esa relación de recurrencia obtenemos:

La **Solución General, que tiene la forma:**

$$\boldsymbol{a_n = a_0 r^n} \qquad \boldsymbol{a_0 \neq 0 \wedge r \neq 0}$$

$$a_n + c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k} = 0, n \geq k$$

$$a_0 r^n + c_1 a_0 r^{n-1} + c_2 a_0 r^{n-2} + \cdots + c_k a_0 r^{n-k} = 0$$

$$a_0 r^{n-k}(r^n + c_1 r^{n-1} + c_2 r^{n-2} + \cdots + c_k) = 0 \quad dado\ que\ a_0 r^{n-k} \neq 0$$

La **Ecuación característica:**

$$\boldsymbol{r^n + c_1 r^{n-1} + c_2 r^{n-2} + \cdots + c_k = 0}$$

Y también se obtiene el **Polinomio Característico** de grado k asociado:

$$\boldsymbol{P(r) = r^n + c_1 r^{n-1} + c_2 r^{n-2} + \cdots + c_k}$$

Teoremas

Relación de recurrencia lineal, homogénea, de primer orden

Hipótesis: Sea una ecuación de recurrencia, lineal, homogénea, de primer orden

$$a_n + c_1 a_{n-1} = 0, \qquad n \geq 1, a_0$$

Tesis: La solución es $a_n = a_0(-c)^n,\ n \geq 0$

Demostración: Proponemos como solución $a_n = a_0 r^n \qquad a_0 \neq 0 \wedge r \neq 0$

$$a_n + c_1 a_{n-1} = a_0 r^n + c_1 a_0 r^{n-1} = 0$$

$$a_0 r^{n-1}(r + c) = 0 \quad \rightarrow \quad r = -c$$

$$a_n = a_0(-c)^n, \qquad n \geq 0$$

Relación de recurrencia lineal, homogénea, de segundo orden

$$a_n + c_1 a_{n-1} + c_1 a_{n-2} = 0, \qquad n \geq 3, \qquad condiciones\ inciales\ a_1\ y\ a_2$$

La ecuación característica es $r^2 + c_1 r + c_2 = 0$

	Raíces reales y distintas S_1 y S_2	**Raíces reales e iguales $S_1 = S_2 = S$**
Solución	$a_n = u.s_1^n + v.s_2^n$	$a_n = u.s^n + n.v.s^n$
Condiciones iniciales a_1 y a_2	$\begin{cases} u.s_1 + v.s_2 = a_1 \\ u.s_1^2 + v.s_2^2 = a_2 \end{cases}$	$\begin{cases} u.s + v.s = a_1 \\ u.s^2 + v.s^2 = a_2 \end{cases}$
Condiciones iniciales a_0 y a_1	$\begin{cases} u + v = a_0 \\ u.s_0 + v.s_1 = a_1 \end{cases}$	$\begin{cases} u = a_0 \\ u.s + v.s = a_1 \end{cases}$

u y v dependen de las condiciones iniciales

UNIDAD III: Estructuras algebraicas finitas

Contenidos: Leyes de composición interna. Propiedades. Monoide. Semigrupo. Semigrupo con unidad. Grupo. Grupo Abeliano. Subgrupo. Anillo. Anillo con unidad. Cuerpo.

Leyes de composición interna

Una ley de composición interna, definida en un conjunto no vacío A, consiste en una operación que asigna a cada par ordenado de elementos de A un único elemento de A. Esto significa que a cada objeto de AxA le corresponde un único elemento de A.

Definición:

Ley de composición interna definida en un conjunto no vacío A, es toda función de AxA en A.

En símbolos: $* \; es\ una\ ley\ interna\ en\ A \leftrightarrow *: A^2 \to A$

Es decir $a \in A \wedge b \in A \to a * b = c \in A$

Son ejemplos de leyes de composición interna, la adición y multiplicación en N, Z, Q, R y C.

Algebra I. Rojo

Sea A un conjunto, * se llama "Ley de composición interna en A" sí y sólo si: $a * b = c \in A, \forall a, b \in A$. Es decir, al componer con * a dos elementos del conjunto A, su resultado es otro elemento de A.

También, * se llama "operación binaria interna en A". El conjunto A está cerrado para *.

Sea * ley de composición interna en A y $a, b \in A$, entonces se verifica la *uniformidad* (al componer dos elementos iguales con un tercero, el resultado es el mismo):

a) $a = b \to a * c = b * c, \forall c \in A$
b) $a = b \to c * a = c * b, \forall c \in A$

Propiedades

Sea * una ley de composición interna en A, es decir, $*: A^2 \to A$

1. Asociatividad

$$* : A^2 \to A\ es\ asociativa \leftrightarrow (a * b) * c = a * (b * c), \forall a, b, c \in A$$

Ejemplo:

$$* \ definida\ en\ R\ por\ a * b = a + b + 2ab\ es\ asociativa\ ya\ que:$$

$$\begin{aligned} a * (b * c) &= a * (b + c + 2bc) \\ &= a + (b + c + 2bc) + 2a(b + c + 2bc) \\ &= a + b + c + 2bc + 2ab + 2ac + 4abc \\ (a * b) * c &= (a + b + 2ab) * c \\ &= (a + b + 2ab) + c + 2(a + b + 2ab)c \\ &= a + b + 2ab + c + 2ac + 2bc + 4abc \end{aligned}$$

2. Conmutatividad

$$*: A^2 \to A\ es\ conmutativa \leftrightarrow a * b = b * a, \forall a, b \in A$$

Ejemplo:

$$* \, definida \; en \; R \; por \; a * b = a + b + 2ab \; es \; conmutativa \; ya \; que:$$

$$a * b = a + b + 2ab = b + a + 2ba = b * a$$

3. Existencia de elemento neutro

Cabe preguntarse si existe en el conjunto A un elemento *e*, que compuesto a izquierda y a derecha con cualquier otro no lo altere. Si un elemento tal existe se lo llama neutro o identidad respecto de la ley * de acuerdo con la siguiente definición:

$$e \in A \; es \; neutro \; respecto \; de * \leftrightarrow \exists e \in A / \forall a \in A : a * e = e * a = a$$

Ejemplo:

$$* \, definida \; en \; R \; por \; a * b = a + b + 2ab:$$

$$a * e = a$$

$$a + e + 2ae = a \rightarrow e + 2ae = 0$$

$$e(1 + 2a) = 0 \rightarrow e = 0 \wedge a = -\frac{1}{2}$$

$$a * e = -\frac{1}{2} * 0 = -\frac{1}{2} + 0 + 2 * \left(-\frac{1}{2}\right) * 0 = -\frac{1}{2} = a$$

$$e * a = a$$

$$e + a + 2ea = a \rightarrow e + 2ae = 0$$

$$e(1 + 2a) = 0 \rightarrow e = 0 \wedge a = -\frac{1}{2}$$

$$e * a = 0 * -\frac{1}{2} = 0 + \left(-\frac{1}{2}\right) + 2 * (0) * \left(-\frac{1}{2}\right) = -\frac{1}{2} = a$$

$$\therefore Existe \; e = 0$$

3.1 Unicidad del neutro

Si existe neutro en A respecto de *, entonces es único.

Supongamos que *e* y *e'* son neutros respecto de *, entonces, por ser *e* neutro y por serlo *e'*, se tiene

$$\boldsymbol{e' = e' * e = e * e' = e}$$

4. Existencia de inverso o simétrico

Sea * una ley interna en A, con elemento neutro *e*. Dado $a \in A$ interesa investigar si existe $a' \in A$, tal que compuesto a izquierda y a derecha con *a* dé por resultado *e*. El elemento neutro es un elemento del conjunto relativo a todos. El inverso, si existe, es relativo a cada elemento. Se propone la siguiente definición:

$$a' \in A \; es \; inverso \; de \; a \in A \; respecto \; de * \leftrightarrow \forall a \in A : \exists e \in A / \; a * a' = a' * a = e$$

Los elementos de A que admiten inversos respecto de * se llama inversibles.

Ejemplo:

$$* \; definida \; en \; R \; por \; a * b = a + b + 2ab \; y \; e = 0:$$

$$a * a' = e$$

$$a + a' + 2aa' = e \; \rightarrow a + a' + 2aa' = 0$$

$$a'(1 + 2a) = -a$$

$$a' = -\frac{a}{1 + 2a}$$

$$a * a' = a * \left(-\frac{a}{1 + 2a}\right) = a + \left(-\frac{a}{1 + 2a}\right) + 2a\left(-\frac{a}{1 + 2a}\right) = a + \left(-\frac{a}{1 + 2a}\right) - \frac{2a^2}{1 + 2a}$$
$$= \frac{a + 2a^2 - a - 2a^2}{1 + 2a} = \frac{0}{1 + 2a} = 0 = e$$

4.1 Unicidad del inverso respecto de una ley asociativa

Si un elemento $a \in A$ admite inverso respecto de la ley asociativa *, entonces dicho inverso es único. Supongamos que *a'* y *a''* son inversos de *a'*. Aplicando consecutivamente la definición de neutro, el hecho de que *a'* es inverso de *a*, la asociatividad, el supuesto que *a''* es inverso de *a*, y la definición de neutro se tiene:

$$a'' = a'' * e = a'' * (a * a') = (a'' * a) * a' = e * a' = a'$$

5. Distributividad de una ley de composición interna respecto de otra

Se considera el caso de dos leyes de composición interna "*****" y "**+**", definidas en un mismo conjunto A. Interesa caracterizar el comportamiento relativo de dichas leyes internas en el sentido de obtener elementos del tipo *(a * b) + c*, o bien *(a + b) * c*.

Definición:

"+" es distributiva a derecha respecto de "*" sí y sólo sí:

$$(a * b) + c = (a + c) * (b + c), \forall a, b, c \; \in A$$

La distributividad a izquierda de "+" respecto de "*" queda definida por:

$$c + (a * b) = (c + a) * (c + b), \forall a, b, c \; \in A$$

Se dice que "+" es distributiva respecto de "*" sí y solo sí lo es a izquierda y a derecha.

Análogamente se define la distributividad de "*" respecta de "+".

Estructura Algebraica

Una estructura algebraica es un objeto matemático consistente en un conjunto no vacío y una relación o ley de composición interna definidas en él.

Según sean las propiedades que deban satisfacer dichas leyes de composición, se tienen los distintos tipos de estructuras algebraicas.

Monoide

El par *(A, +)* donde $A \neq \emptyset$, y + es una función, es un monoide, sí y sólo sí, + es una ley de composición interna en A.

Son modelos de monoides los conjuntos ***N, Z, Q, R*** **y** ***C***, con la adición ordinaria de números.

Propiedades:

Sea $A \neq \emptyset$ y $+: A\ x\ A\ \rightarrow A$ una función

1. Si existe un elemento neutro *e* en A para +, éste es único.
2. Sea + asociativa y *e* pertenece al conjunto A. Si *a* posee inverso *a'* en A, este es único.

Semigrupo

El par *(A, +)* donde $A \neq \emptyset$, y + es una función, es un semigrupo, sí y sólo sí, + es ley interna y asociativa en A.

En otras palabras, un semigrupo es un monoide asociativo.

Semigrupo conmutativo

En particular, si la ley de composición es conmutativa, entonces el semigrupo se llama conmutativo.

Semigrupo con unidad

Si existe elemento neutro, se dice que el semigrupo tiene unidad.

El elemento neutro suele llamarse identidad.

El par *(N, +)* es un semigrupo conmutativo, sin neutro. En cambio (N_0, +) tiene elemento neutro 0.

El objeto (*N*, *) es un semigrupo conmutativo, con elemento neutro o identidad igual a 1.

Grupo

Sea un conjunto no vacío G, y una función *. El par (*G*, *) es un grupo sí y sólo si * es una ley de composición interna en G, asociativa, con elemento neutro y tal que todo elemento de G admite inverso respecto de *.

En forma simbólica:

Definición:

(*G*, *) es un grupo, sí y sólo si, se verifican los axiomas

$G_1 : *: G^2 \rightarrow G, \quad * \ es\ ley\ de\ composición\ interna$

$G_2 : Asociatividad: \quad \forall a, b, c \in G : (a * b) * c = a * (b * c)$

$G_3 : Existencia\ de\ elemento\ neutro\ o\ identidad: \exists e \in G / \forall a \in G : a * e = e * a = a$

$G_4 : Existencia\ de\ inversos: \forall a \in G : \exists a' \in G / a * a' = a' * a = e$

Grupo Abeliano

Un Grupo Abeliano es un Grupo que además también es conmutativo. Es decir que cumple con:

(*G*, *) es un grupo, sí y sólo si, se verifican los axiomas

$G_1 : *: G^2 \rightarrow G, \quad * \ es\ ley\ de\ composición\ interna$

$G_2 : Asociatividad: \quad \forall a, b, c \in G : (a * b) * c = a * (b * c)$

$G_3 : Existencia\ de\ elemento\ neutro\ o\ identidad: \exists e \in G / \forall a \in G : a * e = e * a = a$

$G_4 : Existencia\ de\ inversos: \forall a \in G : \exists a' \in G / a * a' = a' * a = e$

$G_5 : Conmutatividad: \forall a, b \in G : a * b = b * a$

(Z, *) es un grupo abeliano, (A, +) con A={1,-1} y + es el producto ordinario, es grupo abeliano.

Propiedades de los grupos

Sea (G, *) un grupo:

1. **Unicidad del neutro y del inverso:** El elemento neutro es único y el inverso de cada elemento es único.
2. **Regularidad:** Los elementos de todo grupo son regulares, es decir, la ley cancelativa o simplicación es válida para todos los elementos del grupo.
3. **Ecuaciones en un grupo:** cada una de las ecuaciones $\boldsymbol{a * x = a}$ y $\boldsymbol{x * a = b}$ admiten solución única en G para todo *a, b, x* en G.
4. **Inverso de la composición:** En todo grupo, el inverso de la composición de dos elementos es igual a la composición de los inversos en orden permutado. Es decir
$$(\boldsymbol{a} * \boldsymbol{b})' = \boldsymbol{b}' * \boldsymbol{a}'$$
5. $\forall \boldsymbol{x} \in \boldsymbol{G}: (\boldsymbol{x}')' = \boldsymbol{x}$

Subgrupo

El subconjunto no vacío H, del grupo G, es un subgrupo de (G, *), sí y sólo si, (H, *) es grupo.

Sea un conjunto $G \neq \emptyset$, H un subconjunto de G, con $H \neq \emptyset$, el grupo (H, *) es Subgrupo de (G, *) si:

1. H contiene el elemento neutro o identidad de G, $e \in H$
2. * es cerrada en H, $\forall a, b \in H: a * b \in H$
3. H contiene simétricos o inversos, $\forall a \in H, \exists a' \in H / a * a' = a' + a = e$

Propiedades de subgrupos

1. Todo Grupo G, tienen al menos dos subgrupos conocidos como subgrupos triviales.
$$\boldsymbol{H_1} = \{\boldsymbol{e}\}$$
$$\boldsymbol{H_2} = \boldsymbol{G}$$
2. Transitividad de los subgrupos.
Sean H_1, H_2 y H_3 subgrupos de G
Si H_1 es subgrupo de H_2 y H_2 es subgrupo de H_3, entonces H_1 es subgrupo de H_3.
3. La intersección de dos subgrupos es un subgrupo.
Sean *H* y *H'* dos subgrupos de *G*
$$e \in H \wedge e \in H' \rightarrow H \cap H' = \{e\}$$

Anillo

Sea un conjunto no vacío A, y dos funciones + y *

Definición

La terna (*A*, +, *) es un anillo si y sólo si

1. (*A*, +) es un grupo abeliano.
2. (*A*, *) es un semigrupo.
3. El producto es distributivo a izquierda y derecha respecto de la suma.

Estas condiciones se traducen en los siguientes axiomas:

A_1: La adición es ley de composición interna en A.

$$\forall a, b \in A: a + b \in A$$

A_2: La adición es asociativa en A.

$$\forall a, b, c \in A: (a + b) + c = a + (b + c)$$

A_3: Existe neutro en A, que denotamos con 0, respecto de la adición.

$$\exists 0 \in A / \forall a \in A: a + 0 = 0 + a = a$$

A_4: Todo elemento de A admite inverso aditivo u opuesto.

$$\forall a \in A, \exists - a \in A \; / \; a + (-a) = (-a) + a = 0$$

A_5: La adición es conmutativa.

$$\forall a, b \in A: a + b = b + a$$

A_6: El producto es ley de composición interna en A.

$$\forall a, b \in A: a * b \in A$$

A_7: El producto es asociativo en A.

$$\forall a, b, c \in A: (a * b) * c = a * (b * c)$$

A_8: El producto es doblemente distributivo respecto de la suma.

$$\forall a, b, c \in A: \begin{cases} a * (b + c) = a * b + a * c \\ (b + c) * a = b * a + c * a \end{cases}$$

Anillo conmutativo

Si, además, ocurre que la segunda ley de composición es conmutativa diremos que el anillo (*A*, +, *) es un **anillo conmutativo**.

Sea (A, +, *) donde A = Z, + es la suma aditiva y * es el producto ordinario

(A, +) es Grupo Abeliano

(A, *) es Semigrupo

* es doblemente distributivo respecto de +

(A, +, *) es anillo

Además (A, *) es conmutativo, por lo tanto, (A. +, *) es anillo conmutativo.

Anillo con unidad

Si el anillo posee elemento neutro o identidad respecto del producto *, que denotamos con I, entonces se llamará **anillo con identidad** o **con unidad**.

Anillo de división

Un anillo con identidad cuyos elementos no nulos son inversibles se llama **anillo de división**.

Cuerpo

Un anillo con unidad cuyos elementos no nulos son inversibles, es decir, es un anillo de división, y que además es conmutativo, es un **cuerpo**.

Definición:

La terna (*K*, +, *) es un cuerpo si y sólo si es un anillo conmutativo, con unidad, cuyos elementos no nulos admiten inverso multiplicativo.

Los axiomas que caracterizan la estructura de cuerpo son:

1. (*K*, +) es grupo abeliano.
2. (*K*, *) es grupo abeliano
3. El producto es distributivo respecto de la suma.

(R, +, *) donde + es la adición y * el producto ordinario es Cuerpo.

UNIDAD IV: Algebra de Boole

Contenidos: Definición. Propiedades. Principio de dualidad. Puertas lógicas y circuitos booleanos. Minimización de circuitos. Funciones booleanas. Diagrama de Karnaugh.

Definición

Sea S un conjunto no vacío que contiene dos elementos especiales 0 (el cero o elemento neutro) y 1 (el uno o elemento unidad), sobre el cual definimos las operaciones binarias cerradas +, * y una operación monaria (o unaria) ‘. Entonces $\{S, +, *, ', 0, 1\}$ un ***álgebra booleana*** si se cumplen las siguientes condiciones para todos $x, y, z \in B$.

a) Leyes asociativas:

$$(x + y) + z = x + (y + z)$$
$$(x * y) * z = x * (y * z)$$

b) Leyes conmutativas:

$$x + y = y + x$$
$$x * y = y * x$$

c) Leyes distributivas:

$$x * (y + z) = (x * y) + (x * z)$$
$$x + (y * z) = (x + y) * (x + z)$$

d) Leyes de identidad:

$$x + 0 = x$$
$$x * 1 = x$$

e) Leyes de complementos:

$$x + x' = 1$$
$$x * x' = 0$$

Si *B* es un álgebra booleana, se escribe:

$$B = \{S, +, *, ', 0, 1\}$$

Algebra de Boole aplicado al Algebra de Conjuntos

Sea U un conjunto universal y sea $S = P(U)$ el conjunto potencia de U. Si se definen las siguientes operaciones:

$$X + Y = X \cup Y \qquad X * Y = X \cap Y \qquad X' = \bar{X} \qquad \text{en } S$$

Entonces $(S,\cup,\cap,-,\emptyset,U)$ es un álgebra booleana. El conjunto vacío $\emptyset$ asume el papel de 0 y el conjunto universal U hace el papel de 1. Si X, Y y Z son subconjuntos de S, las propiedades se convierten en:

a) $(X \cup Y) \cup Z = X \cup (Y \cup Z)$
   $(X \cap Y) \cap Z = X \cap (Y \cap Z)$
b) $X \cup Y = Y \cup X$
   $X \cap Y = Y \cap X$

c) $X \cap (Y \cup Z) = (X \cap Y) \cup (X \cap Z)$
   $X \cup (Y \cap Z) = (X \cup Y) \cap (X \cup Z)$
d) $X \cup \emptyset = X$
   $X \cap U = X$
e) $X \cup \bar{X} = U$
   $X \cap \bar{X} = \emptyset$

Algebra de Boole aplicado al Algebra de Conjuntos

$$B = (Z,\vee,\wedge,\neg,0,1) \qquad Z = \{0,1\}$$

Propiedades del Álgebra de Boole

Sea $\forall x, y \in S$, se cumplen las siguientes propiedades:

1. Unicidad del complemento: el elemento x' es único. Si x + y = 1 y xy= 0, entonces y=x'
2. Ley de idempotencia: $x + x = x$ $xx = x$
3. Ley de acotación: $x + 1 = 1$ $x0 = 0$
4. Ley de absorción: $x + xy = x$ $x(x + y) = x$
5. Ley de involución: $(x')' = x$
6. Leyes de 0 y 1: $0' = 1$ $1' = 0$
7. Leyes de De Morgan: $(x + y)' = x'y'$ $(xy)' = x' + y'$

Principio de dualidad

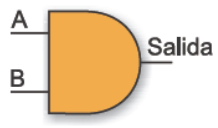
El *dual* de una afirmación que incluye expresiones booleanas se obtiene sustituyendo 0 por 1, 1 por 0, + por * y * por +.

Ejemplo: El dual de $(x + y)' = x'y'$ es $(xy)' = x' + y'$

Observación: Si dos expresiones booleanas son iguales, sus duales también lo son.

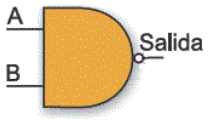
Puertas lógicas y circuitos booleanos

AND: A * B



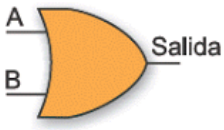
A	B	Salida
0	0	0
0	1	0
1	0	0
1	1	1

NAND: (A * B)'



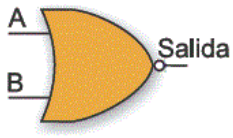
A	B	Salida
0	0	1
0	1	1
1	0	1
1	1	0

OR: A + B



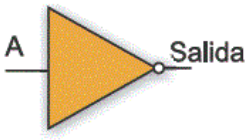
A	B	Salida
0	0	0
0	1	1
1	0	1
1	1	1

NOR: (A + B)'



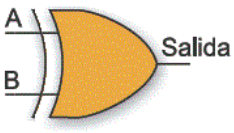
A	B	Salida
0	0	1
0	1	0
1	0	0
1	1	0

NOT: A'



A	Salida
0	1
1	0

XOR: (A ⊕B)



A	B	Salida
0	0	0
0	1	1
1	0	1
1	1	0

Funciones booleanas

Sea B = {0,1}. El producto cartesiano se puede definir como BxB= {0,1} * {0,1}.

Se puede definir una función booleana *f* de dos variables según $f(x) = B^2 \rightarrow B$.

Dominio: {(0, 0), (0, 1), (1, 0), (1, 1)}; Imagen: {0, 1}

Minitérmino

Dado un número n de variables, un **minitérmino** es un producto lógico, cuyos factores son todas las variables (negadas o no).

Es decir, es el producto lógico de *k* variables x_i. Siendo $\widetilde{x_i} = x_i$ ^ $\widetilde{x_i} = x_i{}'$.

Función canónica (o normal) disyuntiva

Si la expresión de la función booleana se expresa como una suma de minitérminos se define como **función canónica (o normal) disyuntiva**. También es llamado disyunción de conjunciones.

$$f(x_1, x_2, x_3) = (x_1 \cdot x_2 \cdot x_3') + (x_1 \cdot x_2 \cdot x_3) + (x_1 \cdot x_2' \cdot x_3)$$

Maxitérmino

Dado un número n de variables, un **maxitérmino** es una suma lógica, cuyos factores son todas las variables (negadas o no).

Es decir, es la suma lógica de *k* variables $\widetilde{x_i}$. Siendo $\widetilde{x_i} = x_i$ ^ $\widetilde{x_i} = x_i{}'$.

Función canónica (o normal) disyuntiva

Si la expresión de la función booleana se expresa como un producto de maxitérminos se define como una **función canónica (o normal) conjuntiva**. También es llamado conjunción de disyunciones.

$$f(x_1, x_2, x_3) = (x_1{}' + x_2{}' + x_3\ ) \cdot (x_1 + x_2{}' + x_3) \cdot (x_1 + x_2' + x_3{}')$$

Minimización de circuitos

Las expresiones de una función booleana pueden ser minimizadas utilizando recursos algebraicos.

Ejemplo:

$$f(x_1, x_2, x_3) = (x_1 \cdot x_2' \cdot x_3{}') + (x_1 \cdot x_2{}' \cdot x_3) + (x_1 \cdot x_2 \cdot x_3)$$

$$f(x_1, x_2, x_3) = x_1 \cdot x_2' \cdot (x_3{}' + x_3) + (x_1 \cdot x_2 \cdot x_3)$$

$$f(x_1, x_2, x_3) = x_1 \cdot x_2' \cdot 1 + (x_1 \cdot x_2 \cdot x_3)$$

$$f(x_1, x_2, x_3) = x_1 \cdot x_2' + x_1 \cdot x_2 \cdot x_3$$

Diagrama de Karnaugh

Un mapa de Karnaugh (también conocido como tabla de Karnaugh o diagrama de Veitch) es un diagrama utilizado para la simplificación de funciones algebraicas en forma canónica. A partir de la tabla de Karnaugh se puede obtener una forma canónica mínima (con el mínimo número de términos).

Los mapas de Karnaugh nos brindan maneras pictóricas de encontrar implicantes primos y formas de sumas minimizadas para las expresiones de las funciones booleanas. Son representados por cuadrados los productos fundamentales en las variables. Un implicante primo será una pareja de cuadrados adyacentes o un cuadro aislado.

Ejemplos:

Con dos variables: $f(x): B^2 \rightarrow B$

	x'	x
y'	$x' \cdot y'$	$x \cdot y'$
y	$x' \cdot y$	$x \cdot y$

00	10
01	11

Tabla de Conjunción	
0	0
0	1

Expresión minimizada: $x \cdot y$

Con dos variables: $f(x): B^2 \rightarrow B$

	$x'y'$	$x'y$	xy	xy'
z'	$x' \cdot y' \cdot z'$	$x' \cdot y \cdot z'$	$x \cdot y \cdot z'$	$x \cdot y' \cdot z'$
z	$x' \cdot y' \cdot z$	$x' \cdot y \cdot z$	$x \cdot y \cdot z$	$x \cdot y' \cdot z$

000	010	110	100
001	011	111	101

Tabla de Conjunción			
0	0	0	0
0	0	1	0

Expresión minimizada: $x \cdot y \cdot z$

UNIDAD V: Grafos

Contenidos: Definición. Vértices, lados, grados, bucles, caminos. Circuitos. Circuitos eulerianos. Circuitos hamiltonianos. Caminos de longitud mínima. Matrices de adyacencia y grafos. Dígrafos.

Definición

Grado no dirigido

Un *grafo* (o *grafo no dirigido*) G consiste en un conjunto V de *vértices* (o *nodos*) y un conjunto E de *aristas* (o *arcos*) tal que cada arista $e \in E$ se asocia con un par no ordenado de vértices. Si existe una arista única e asociada con los vértices u y w, se escribe $e = (v, w)$ o $e = (w, v)$. En este contexto. (v, w) denota una arista entre v y w en un grafo no dirigido y *no* es un par ordenado.

Grado dirigido

Un *grafo dirigido* (o *dígrafo*) G consiste en un conjunto V de *vértices* (o *nodos*) y un conjunto E de *aristas* (o *arcos*) tal que cada arista $e \in E$ se asocia con un par ordenado de vértices. Si existe una arista única e asociada con el par ordenado (v, w) de vértices, se escribe $e = (v, w)$ o $e = (w, v)$ que denota una arista de v hacia w.

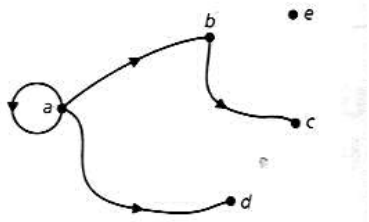
Es decir, los elementos de V son pares ordenados, y las aristas tienen un sentido definido.

Si G es un grafo (no dirigido o dirigido) con vértices V y aristas E, se escribe $G = (V, E)$.

A menos que se especifique lo contrario, se supone que los conjuntos E y V son finitos y que V es no vacío.

Vértices, lados, grados, bucles, caminos

Para cualquier arista, como (b, c), decimos que la arista es *incidente* con los vértices b, c: b es *adyacente* hacia c, mientras que c es *adyacente desde* b.



Grado de un vértice: Sea G un grafo o multigrafo no dirigido. Para cualquier vértice v de G, el grado de v, que se denota $grad(v)$ o $\delta(v)$, es el número de aristas en G que son incidentes en v.

Vértice origen o fuente: vértice desde donde sale una arista; b en (b, c).

Vértice término o terminal: vértice hacia donde se dirige una arista; c en (b, c).

Vértice aislado: vértice que no tiene aristas incidentes. Su grado es 0.

Lazo o Bucle: arista incidente en un mismo vértice. Su grado es mayor a 1.

Aristas paralelas: cuando dos aristas se asocian a un mismo par de vértices.

Grafo simple: Es un grafo sin lazos ni aristas paralelas.

Grafo ponderado: Un grafo con números en las aristas se llama **grado ponderado**. Si la arista e se etiqueta k, se dice que el **peso de la arista** e es k.

Camino

Sean *x*, *y* vértices (no necesariamente distintos) de un grafo no dirigido $G = (V, E)$. Un camino *x*-*y* en *G* es una sucesión alternada finita (sin lazos)

$$x = x_0, e_1, x_1, e_{2,} x_2, e_3 \ldots, e_{n-1}, x_{n-1}, e_n, x_n = y$$

de vértices y aristas de *G*, que comienza en el vértice ***x*** y termina en el vértice ***y*** y que contiene las ***n*** aristas $e_i = \{x_{i-1}, x_i\}$ donde $1 \leq i \leq n$.

En un camino se pueden repetir aristas y vértices.

Longitud del camino

La *longitud* de un camino de escribe como *n*, y el número de aristas que hay en el camino. Si $n = 0$, no existen aristas, $x = y$, y el camino se denomina *trivial*.

Camino cerrado y abierto

Cualquier camino *x*-*y* donde $x = y$ (y $n > 1$) es un ***camino cerrado***. En caso contrario, el camino es ***abierto***, es decir, $x \neq y$.

Circuitos

Considerando un camino *x*-*y* en un grafo no dirigido $G = (V, E)$, definimos:

Recorrido o trayectoria: si no se repite ninguna arista en el camino *x*-*y*.

Circuito: si dado un recorrido *x*-*x* este es cerrado.

Camino simple: cuando ningún vértice del camino *x*-*y* se presenta más de una vez.

Ciclo: si dado un camino simple x-x, éste es cerrado.

Grafo conexo e inconexo

Sea $G = (V, E)$ un grafo no dirigido. Decimos que *G* es ***conexo*** si existe un camino simple entre cualesquiera dos vértices distintos de *G*.

Un grafo que no es conexo es ***disconexo***.

Multígrafo

Un grafo $G = (V, E)$ es un *multígrafo* si existen $a, b \in V, a \neq b$, con dos o más aristas de la forma (a, b) (para un grafo dirigido), o $\{a, b\}$ (para un grafo no dirigido).

Es decir, que existe algún vértice cuyo grado es mayor a 1.

Se denomina *multiplicidad* cuando un vértice tiene más de una arista incidente.

Circuitos eulerianos

Definición:

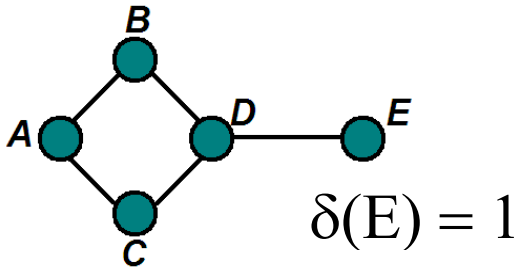
Sea $G = (V, E)$ un grafo o multígrafo no dirigido sin vértices aislados (grafo conexo). Entonces *G* tiene un ***circuito euleriano*** si existes un circuito en *G* que recorre cada arista del grafo exactamente una vez.

Si existe un recorrido abierto de *a* a *b* en *G* que recorre cada arista de *G* exactamente una vez, este recorrido se llamará ***recorrido euleriano***.

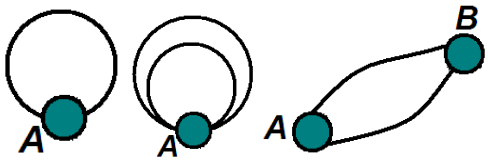
Teoremas de circuitos y trayectorias de Euler

Teorema 1 (de circuitos)

a) **Si un grafo *G* es conexo y tiene un vértice de grado impar, entonces, no puede existir un Circuito de Euler en *G*.**



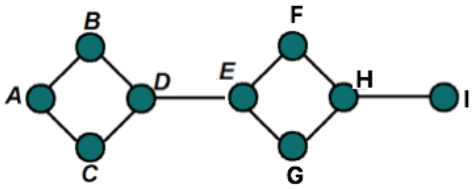
b) **Si *G* es un grafo conexo y todos los vértices tienen grado par, entonces existe un Circuito de Euler en *G*.**
Si *G* es conexo y tiene una o dos aristas, la mínima expresión será



Por inducción, mientras los vértices que se agregan sean de grado par (y los existentes sigan siendo de grado par), seguirá existiendo Circuito de Euler.

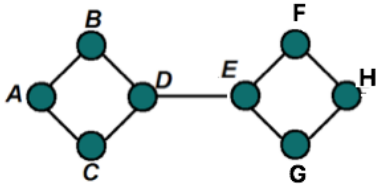
Teorema 2 (de las trayectorias)

a) **Si un grafo *G* tiene mas de dos vértices de grado impar, entonces no puede existir una trayectoria de Euler.**
Sean V_1, V_2, V_3 ... vértices de grado impar. Cualquier trayectoria de Euler debe salir (o llegar) a cada uno de los vértices, sin poder regresar (o salir) de ellos ya que tienen grado impar.

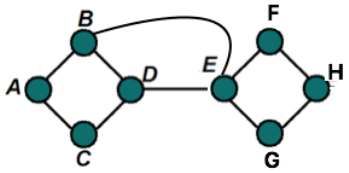


Uno de estos vértices V_1, V_2, V_3 ... de grado impar podría ser el comienzo de la trayectoria de Euler y otro el final; pero los demás vértices quedarán sin una trayectoria sin recorrer.

b) **Si *G* es conexo y tiene dos vértices de grado impar, entonces existe una trayectoria de Euler en *G*, que debe empezar en uno de los vértices de grado impar y terminar en el otro.**



Trayectoria
D, B, A, C, D, E, F, H, G, **E**



Trayectoria

B, A, C, D, B, E, F, H, G, E, **D**

Algoritmo de Fleury

Es un algoritmo para obtener un circuito de Euler para una gráfica conexa sin vértices de grado impar.

Una arista $\{vi, vj\}$ es un **puente** en una gráfica conexa de G si al eliminar $\{vi, vj\}$ se crea una gráfica disconexa.

Algoritmo: Sea $G = (V, E, \gamma)$ una gráfica conexa con todos sus vértices de grado par.

Paso 1. Se elige un miembro de v de V como vértice inicial para el circuito. Sea π: v el inicio de la trayectoria por construir.

Paso 2. Suponga que ya se ha construido π: $v, u, \ldots, w$. Si en w sólo existe una arista $\{w, z\}$, se extiende π a π: $v, u, \ldots, w, z$. Se elimina $\{w, z\}$ de E y w de V. Si en w existen varias aristas, se elige una que no sea un puente $\{w, z\}$. Extienda π a π: $v, u, \ldots, w, z$ y se elimina $\{w, z\}$ de E.

Paso 3. Repita el paso 2 hasta que no sobren aristas en E.

Circuitos hamiltonianos

Definición:

Si $G = (V, E)$ es un grafo o multígrafo con $|V| \geq 3$, decimos que G tiene un ***ciclo hamiltoniano*** si existe un ciclo en G que contenga cada vértice de V solo una vez, excepto por el vértice inicial y final que aparece dos veces.

Un ***camino hamiltoniano*** es un camino simple (y no un ciclo) de G que contiene todos los vértices solo una vez.

A diferencia de los ciclos de Euler, no se conocen condiciones necesarias y suficientes que se verifiquen con facilidad para la existencia de un ciclo de Hamilton en un grafo.

Caminos de longitud mínima

Recordando que un grafo ponderado es aquel al cual se le asigna valores a las aristas y que la longitud de una trayectoria en un grafo ponderado es la suma de los pesos de las aristas en la trayectoria. Sea $w(i, j)$ el peso de la arista (i, j). En los grafos ponderados con frecuencia se quiere encontrar la **ruta más corta**.

Algoritmo de Dijkstra

Este algoritmo encuentra la longitud de una ruta más corta del vértice a al vértice z en un grafo ponderado conexo. El peso de la arista (i, j) es $w(i, j) > 0$, y la etiqueta del vértice x es $L(x)$. Al terminar, $L(z)$ es la longitud de la ruta más corta de a a z.

Sea un grafo $G = (V, E)$, consideramos un vértice inicial, a, con $L(a) = 0$ y que todos los demás vértices $\forall x \in V: x \neq a$, están a una distancia infinita de a en principio, $L(x) = \infty$.

El algoritmo va a recorrer todos los vértices "acumulando" las distancias desde a hasta el vértice que se visita en la etiqueta $L(x)$.

Para etiquetar se utiliza la fórmula: $L(x) = \min \{L(x), L(v) + w(v, x)\}$

Pasos:

1. Se inicia el recorrido por los vértices adyacentes a a en el inicio, siguiendo por los adyacentes de los ya visitados.
2. Se compara los pesos de los vértices adyacentes, escogiendo el menor de ellos.
3. Al peso del vértice escogido, se suma el peso acumulado de los vértices escogidos anteriormente.
4. Se repite hasta llegar a la salida.

Teorema:

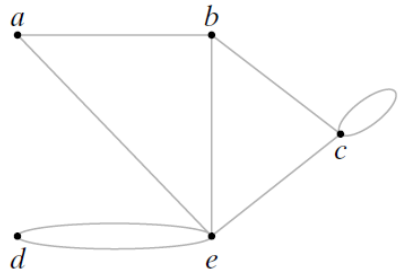
El algoritmo de la ruta más corta de Dijkstra encuentra correctamente la longitud de una ruta más corta desde *a* hasta *z*.

Matrices de adyacencia e incidencia

Matriz de adyacencia

La matriz de adyacencia es un método de representación de un grafo.

Para obtener la matriz de adyacencia de un grafo, primero se selecciona un orden de los vértices. Después se etiquetan las filas y columnas de una matriz con los vértices ordenados. El elemento en esta matriz en la fila *i* y la columna *j*, $i \neq j$, es el número de aristas incidentes en *i* y *j*. Si $i = j$, el elemento es dos veces el número de ciclos que inciden en *i*.

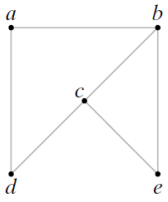


$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 2 \\ 1 & 1 & 1 & 2 & 0 \end{pmatrix} \end{array}$$

Teorema:

Si A es la matriz de adyacencia de un grafo simple, el elemento *ij* de A^n es igual al número de trayectorias de longitud *n* del vértice *i* al vértice *j*, con $n \in N$.

Si n = 1, A^1 es simplemente A. El elemento *ij* es 1 si hay una arista de *i* a *j*, que es una trayectoria de longitud 1, y 0 de otra manera.



$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 2 \\ 1 & 1 & 1 & 2 & 0 \end{pmatrix}$$

$$A^2 = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 2 \\ 1 & 1 & 1 & 2 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 2 \\ 1 & 1 & 1 & 2 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 2 & 0 & 1 \\ 0 & 3 & 1 & 2 & 1 \\ 2 & 1 & 3 & 0 & 1 \\ 0 & 2 & 0 & 2 & 1 \\ 1 & 1 & 1 & 1 & 2 \end{pmatrix}$$

Entonces, esto significaria que, por ejemplo, desde *b* a *b*, existen 3 trayectorias posibles: (*b*, *a*, *b*), (*b*, *c*, *b*) y (*b*, *e*, *b*)

Matriz de incidencia

Para obtener la matriz de incidencia, se etiquetan las filas con los vértices y las columnas con las aristas (en algún orden arbitrario). El elemento en el renglón *v* y la columna *e* es 1 si *e* es incidente en *v*, y 0 de otra manera.

Unidad VI: Árboles

***Contenidos*:** Árboles con raíz. Árboles etiquetados. Árboles binarios. Búsqueda en árboles. Árboles no dirigidos. Árboles generadores o de expansión. Árboles de expansión mínima. Algoritmo de Prim. Algoritmo de Kruskal. Algoritmo de árboles de deducción de una fórmula de la lógica proposicional.

Árboles con raíz

Sea *A* un conjunto y *T* una relación en *A*. *T* es un ***árbol*** si existe un vértice v_0 en *A* con la propiedad de que existe una única trayectoria en *T* de v_0 hacia cualquier otro vértice en *A*, pero no existe una trayectoria de v_0 a v_0.

El vértice v_0, es único y con frecuencia es llamado ***raíz*** del árbol *T*, y *T* es entonces un ***árbol con raíz***. Escríbase (T, v_0) para denotar un árbol *T* con raíz v_0.

Si (T, v_0) es un árbol con raíz sobre el conjunto *A*, un elemento *v* de *A* es un ***vértice en T***.

Teorema 1

Sea (T, v_0) un árbol con raíz. Entonces:

a) No existen ciclos en T.

Demostración: Supóngase que existe un ciclo *q* en *T*, que comienza y termina en el vértice *v*. Por la definición de árbol. Se sabe que $v \neq v_0$, y debe existir una trayectoria *p* de v_0 a v. Entonces $q \circ p$ (*p* compuesta en *q*) es una trayectoria de v_0 a v diferente de *p*, lo cual contradice la definición de árbol (que existe una única trayectoria de v_0 hacia cualquier otro vértice).

b) v_0 es la única raíz en T.
Demostración: Si v_0' es otra raíz de *T*, existe una trayectoria *p* de v_0 a v_0' y una trayectoria *q* de v_0' a v_0 (ya que v_0' es una raíz). Entonces $q \circ p$ es un ciclo de v_0 a v_0, lo que por definición es imposible. Por lo tanto, el vértice v_0 es la única raíz.

c) Cada vértice en T distinto de v_0 tiene grado interno uno, y v_0 tiene grado interno cero.
Demostración: Sea w_1 un vértice de *T* distinto de v_0. Entonces existe una única trayectoria $v_0, \ldots, v_k, w_1$ en *T*. Esto significa que $(v_k, w_1) \in T$, de modo que w_1 tiene al menos grado interno uno. Si el grado interno de w_1 es mayor que uno, deben existir vértices w_2 y w_3 distintos tales que (w_2, w_1) y (w_3, w_1) se encuentren ambos en *T*. Si $w_2 \neq v_0$ y $w_3 \neq v_0$, existen trayectorias p_2 *de* v_0 *a* w_2 y p_3 *de* v_0 *a* w_3, por definición. Entonces, $(w_2, w_1) \circ p_2$ y $(w_3, w_1) \circ p_3$ son dos trayectorias diferentes de v_0 *a* w_1, y esto contradice la definición de un árbol con raíz v_0. Por lo tanto, el grado interno de w_1 es uno.

Teorema 2

Sea (T, v_0) un árbol con raíz sobre un conjunto *A*. Entonces:

a) *T es irreflexivo.*
b) *T es asimétrico.*
c) *Si $(a, b) \in T$ y $(b, c) \in T$, entonces $(a, c) \notin T$, para toda a, b y c en A.*

Teorema 3

Si (T, v_0) es un árbol con raíz y $v \in T$, entonces $T(v)$ también es un árbol con raíz *v*. $T(v)$ es el ***subárbol*** de *T* que comienza en *v*.

Terminologías de un árbol:

Nivel de un vértice: es la longitud de la trayectoria simple de la raíz v_0 a *v*.

Altura de un árbol: es el número máximo de nivel que ocurre.

Sea *T* un árbol con raíz v_0. Supongamos que *x*, *y* y *z* son vértices en *T* y que $(v_0, v_1, \dots, v_n)$ es una trayectoria simple en *T*. Entonces:

a) v_{n-1} es el ***padre*** de v_n.
b) $v_0, \dots, v_{n-1}$ son ***ancestros*** de v_n.
c) v_n es un ***hijo*** de v_{n-1}.
d) Si *x* es un ancestro de *y*, *y* es un ***descendiente*** de *x*.
e) Si *x* e *y* son hijos de *z*, *x* e *y* son ***hermanos***.
f) Si *x* no tiene hijos, *x* es un ***vértice terminal*** (o una ***hoja***).
g) Si *x* no es un vértice terminal, *x* es un ***vértice interno*** (o una ***rama***).
h) El ***subárbol*** de *T* con raíz en *x* es la gráfica con el conjunto de vértices *V* y el conjunto de aristas *E*, donde *V* es *x* junto con los descendientes de *x* y *E* es el conjunto de aristas en una trayectoria simple de *x* a algún vértice en *V*.

Árboles etiquetados

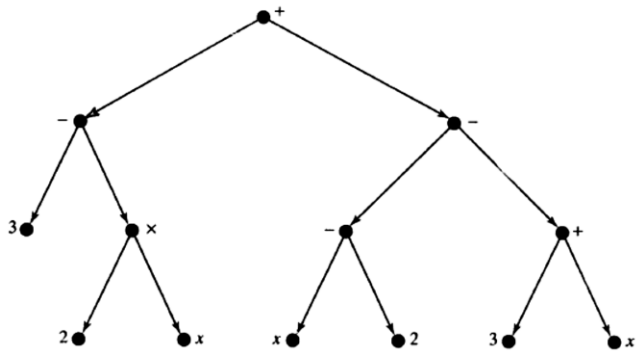
Es un árbol en el cada vértice tiene una única etiqueta.

A veces es útil etiquetar los vértices o aristas de un dígrafo para indicar su uso para un propósito específico. La utilidad del árbol se enfatiza mediante las etiquetas sobre los vértices. Así, se representará los vértices como puntos y se mostrará la etiqueta de cada vértice junto al punto que representa dicho vértice.

Considerando la expresión algebraica con cada operación entre paréntesis

$$\big(3 - (2 * x)\big) + ((x - 2) - (3 + x))$$

Cada una de estas expresiones tienen un **operador central** que corresponde al último cálculo que puede realizarse. Así, $+$ es central para la expresión principal anterior, $-$ es central para $\big(3 - (2 * x)\big)$, y así sucesivamente. En este árbol, se etiqueta la raíz con el operador central de la expresión principal.



Árboles binarios

Sea un entero positivo *n*, un árbol *T* es un ***n-árbol*** (árbol n-ario) si cada vértice tiene a lo más *n* hijos. Si todos los vértices de *T* distintos de las hojas tienen exactamente *n* hijos, *T* es un ***n-árbol completo***.

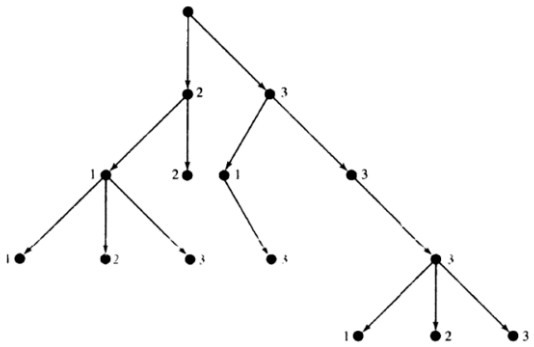
En particular, con frecuencia se dice que un 2-árbol es un ***árbol binario***, y un 2-árbol completo es un ***árbol binario completo***.

Es decir, que un árbol binario es aquel en el cual cada nodo puede tener un hijo a izquierda y un hijo a derecha, pero no más de 2 hijos.

El árbol binario será ***completo*** cuando todos sus nodos a excepción de los del último, tienen exactamente dos hijos, y será ***incompleto*** cuando exista al menos un nodo, a excepción de los del último nivel, que no tengo dos hijos.

Árbol posicional

Dado un n-árbol (T, v_0). Suponiendo que cada vértice tiene exactamente *n* hijos, ordenados de 1 a *n*, pero que puede faltar alguno de los hijos en la sucesión. Se etiqueta a los hijos restantes de acuerdo con la *posición* que ocupan en la sucesión hipotética. Así, se etiqueta los hijos de cualquier vértice con distintos números del conjunto $\{1, 2, \ldots, n\}$.

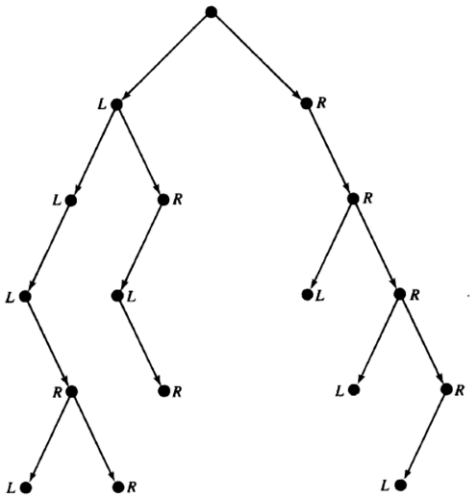


A este tipo de árboles etiquetados se los conoce como ***árbol posicional***.

En el ejemplo se ve un 3-árbol (árbol ternario), donde todas las posiciones reales están etiquetadas.

El ***árbol binario posicional*** es de particular importancia. Es este caso, se etiquetan las posiciones de los hijos potenciales como *izquierda* y *derecha*, en vez de 1 y 2.

En el siguiente árbol binario posicional se puede ver que se etiqueta los hijos como *L* para izquierda y *R* para derecha.



Árbol de búsqueda binaria

Un árbol de búsqueda binaria es un árbol binario *T* en el que se asocian datos a los vértices. Los datos están arreglados de manera que, para cada vértice *v* en *T*, cada dato en el subárbol de la izquierda de *v* es menor que el dato en *v*, y cada dato en el subárbol de la derecha de *v* es mayor que el dato en *v*.

Búsqueda en árboles

El proceso de visita de cada vértice de un árbol en cierto orden específico es una ***búsqueda en el árbol***. En algunos textos, se llama a este proceso ***caminar*** o ***recorrer*** el árbol.

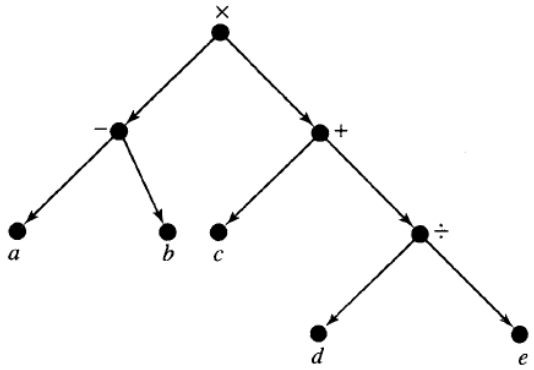
Considérese las búsquedas en árboles binarios posicionales. Sus hijos potenciales se denotarán como v_L (el hijo izquierdo) y v_R (el hijo derecho), donde uno o ambos pueden estar ausentes.

Sea *T* un árbol binario posicional con raíz *v*. Entonces, si existe v_L, el subárbol $T(v_L)$ es el ***subárbol izquierdo*** de *T* y si existe v_R, el subárbol $T(v_R)$ es el ***subárbol derecho*** de *T*.

Esta notación permite especificar algoritmos de búsqueda de una manera recursiva, natural y poderosa.

Algoritmos de búsqueda

Los nombres de las siguientes búsquedas indican el momento en que se visita la raíz del subárbol con respecto del momento en que se visita los subárboles izquierdo y derecho.



Árbol binario de ejemplo. Expresión algebraica (a - b) x (c + (d / e))

Búsqueda PREORDEN

Algoritmo:

PASO 1: Visite *v*.

PASO 2: Si existe v_L, entonces aplique este algoritmo a ($T(v_L)$, v_L).

PASO 3: Si existe v_R, entonces aplique este algoritmo a ($T(v_R)$, v_R).

Fin del algoritmo.

De manera informal:

1. Visite la raíz.
2. Busque en el subárbol izquierdo, si éste existe.
3. Busque en el subárbol derecho, si éste existe.

Para el árbol binario de ejemplo el resultado sería: $x - a\ b + c\ \div d\ e$

A ese resultado de la búsqueda se la conoce como ***forma prefija*** o ***polaca*** de la expresión algebraica dada. Para su interpretación, se mueve de izquierda a derecha hasta encontrar una cadena de la forma Fxy, donde *F* es el símbolo de una operación binaria, y *x* e *y* son números. Se evalúa xFy y se sustituye la respuesta en vez de la cadena Fxy. Se continuará con este procedimiento hasta que sólo quede un número.

Búsqueda ENTREORDEN

Algoritmo:

PASO 1: Busque en el subárbol izquierdo $(T(v_L), v_L)$, si éste existe.

PASO 2: Visite la raíz *v*.

PASO 3: Busque en el subárbol derecho $(T(v_R), v_R)$, si éste existe.

Fin del algoritmo.

De manera informal, el orden de visita es izquierdo, raíz, derecho.

Para el árbol binario de ejemplo el resultado sería: $a - b \times c + d \div e$

Obsérvese que ésta es precisamente la expresión original, sin los paréntesis. Como los símbolos algebraicos se encuentran entre sus argumentos, ésta es llamada con frecuencia ***notación entrefija***, la cual es ambigua sin paréntesis.

Búsqueda POSTORDEN

Algoritmo:

PASO 1: Busque en el subárbol izquierdo $(T(v_L), v_L)$, si éste existe.

PASO 2: Busque en el subárbol derecho $(T(v_R), v_R)$, si éste existe.

PASO 3: Visite la raíz *v*.

Fin del algoritmo.

De manera informal, el orden de visita es izquierdo, derecho, raíz.

Para el árbol binario de ejemplo el resultado sería: $ab - cde \div + \times$

La expresión obtenida a partir de esta búsqueda es la forma ***posfija, polaca inversa*** o ***sufija*** de la expresión. Se evalúa de manera similar a la forma polaca, pero el símbolo del operador está después de sus argumentos y no antes.

La forma polaca inversa también carece de paréntesis, y por medio de ella se puede recuperar el árbol de la expresión. Se utiliza con más frecuencia que la forma polaca.

Árboles no dirigidos

Un ***árbol no dirigido*** es es un árbol con todas sus aristas bidireccionales. La gráfica de un árbol no dirigido *T* tendrá una única línea sin flechas que une los vértices *a* y *b* siempre que (a, b) y (b, a) pertenezcan a *T*. El conjunto $\{a, b\}$, donde (a, b) y (b, a) están en *T*, es una **arista no dirigida** de *T*. En este caso, los vértices *a* y *b* son **vértices adyacentes**.

Teoremas

Teorema 1

Sea *R* una relación simétrica en un conjunto *A*. Entonces las siguientes proposiciones son equivalentes.

a. *R es un árbol no dirigido.*
b. *R es conexo y acíclico.*

Teorema 2

Sea *R* una relación simétrica en un conjunto *A*. Entonces *R* es un árbol no dirigido si y sólo si cualquiera de las siguientes proposiciones es verdadera.

a. *R es acíclica, y si se agrega cualquier arista no dirigida a R, la nueva relación no será acíclica.*
b. *R es conexa, y si se elimina cualquier asita no dirigida de R, la nueva relación no será conexa.*

Teorema 3

Un árbol con *n* vértices tiene $n - 1$ aristas.

Árboles generadores o de expansión

Si *R* es una relación simétrica conexa sobre un conjunto *A*, un árbol *T* en *A* es un ***árbol de expansión*** para *R* si *T* es un árbol con exactamente los mismos vértices que *R* y que se puede obtener de *R* eliminando algunas aristas de *R*. Los árboles de expansión no son únicos.

También se puede obtener un ***árbol de expansión no dirigido*** para una relación simétrica conexa *R*. En este caso el árbol de expansión es aquel que posee aristas bidireccionales.

Para su obtención, sólo hay que eliminar varias aristas no dirigidas de *R* hasta llegar a un punto donde la eliminación de una más de las aristas no dirigidas produciría una relación no conexa.

Árboles de expansión mínima

Dado un grafo ponderado, un ***árbol de expansión mínima*** es un árbol de expansión no dirigido para el cual el peso total de las aristas en el árbol sea el menor posible.

Algoritmo de Prim

Sea *R* una relación simétrica, conexa con *n* vértices.

PASO 1. Se elige un vértice v_1 de *R*. Sea $V = \{v_1\}$ y $E = \{\}$.

PASO 2. Se elige uno de los vecinos más cercanos a v_i de V, que sea adyacente a v_j, $v_j \in R$, y tal que la arista (v_i, v_j) no forme un ciclo con miembros de E. Se agrega v_j a V y (v_i, v_j) a E.

PASO 3. Se repite el paso 2 hasta que $|E| = n - 1$. Entonces *V* contiene los *n* vértices de *R* y *E* contiene las aristas de un árbol de expansión mínima para *R*.

Fin del algoritmo.

Algoritmo de Kruskal

Sea *R* una relación simétrica, conexa con *n* vértices y sea $S = \{e_1, e_2, \dots, e_k\}$ el conjunto de todas las aristas con pesos en R.

PASO 1. Se elige una arista e_1 en *S* de peso mínimo. Sea $E = \{e_1\}$. Se reemplaza *S* por $S - \{e_1\}$.

PASO 2. Se selecciona una arista e_i de menor peso que no forme un ciclo con los miembros de *E*. Se reemplaza *E* con $E \cup \{e_i\}$ y S con $S - \{e_i\}$.

PASO 3. Se repite el paso 2 hasta que $|E| = n - 1$.

Fin del algoritmo.

Unidad VII: Introducción a los Lenguajes Formales y Máquinas de Estado Finito

Contenido: Definición de lenguajes formales. Alfabeto. Gramáticas. Tipos de gramáticas. Forma de Backus-Naur. Definición de MEF. MEF como modelo de sistema físico. Autómatas de estado finito deterministas y no deterministas. Equivalencias de AEFD y AEFND. Representación de gramáticas regulares en autómatas.

Definición de lenguajes formales

Definición: Sea *A* un conjunto finito. Un ***lenguaje (formal)*** *L* sobre *A* es un subconjunto de *A**, el conjunto de todas las cadenas sobre *A*.

Un lenguaje formal está conformado por cadenas de símbolos y cada cadena debe tener un significado "preciso e inequívoco".

Un lenguaje posee Vocabulario y Gramática.

Vocabulario son los "bloques o signos" a partir de los cuales se construyen las cadenas del lenguaje.

El conjunto de todas las oraciones con construcción adecuada que pueden ser producidas aplicando las reglas de una gramática *G* es el ***lenguaje*** de *G* y se denota $L(G)$.

Alfabeto

Es un conjunto de símbolos *finito* distinto del vacío. Se lo representa por medio de **Σ.** Son alfabetos $\Sigma_1 = \{0, 1, *, \wedge\}$ o $\Sigma_2 = \{a, b, c\}$ (Σ se lee sigma).

Cadenas de símbolos o palabras: son las formadas con los símbolos del alfabeto. Éstas contienen un número finito de símbolos del alfabeto.

Si Σ es un alfabeto y $n \in Z^+$, definimos las potencias de Σ recursivamente de la siguiente manera:

1) $\Sigma^1 = \Sigma, y$
2) $\Sigma^{n+1} = \{xy | x \in \Sigma, y \in \Sigma^n\}$, donde xy denota la yuxtaposición de *x* e *y*.

Gramáticas

La gramática se define como el conjunto de reglas que permiten formar las palabras del lenguaje. Estas reglas se valen de producciones donde se involucran:

Símbolos terminales: son los símbolos que forman las cadenas del lenguaje.

Símbolos no terminales: son los símbolos auxiliares para la formación de las cadenas del lenguaje.

Una ***gramática de estructura de frases*** (o simplemente ***gramática***) *G* consiste en:

a) Un conjunto finito *N* de ***símbolos no terminales***
b) Un conjunto finito *T* de ***símbolos terminales*** donde $N \cap T = \emptyset$
c) Un subconjunto finito *P* de $[(N \cup T)^* - T^*] \times (N \cup T)^*$, llamado conjunto de ***producciones***.
d) Un ***símbolo de inicio*** $\sigma \in N$.

Se escribe $G = (N, T, P, \sigma)$

Una producción $(A, B) \in P$ suele escribirse $A \rightarrow B$.

En la producción $A \rightarrow B$, $A \in (N \cup T)^* - T^*$ y $B \in (N \cup T)^*$; entonces, A debe incluir al menos un símbolo no terminal, mientras que B puede consistir en cualquier combinación de símbolos no terminales y terminales.

Tipos de gramáticas: Jerarquías de Chomsky

Es una jerarquía de distintos tipos de gramáticas formales que generan lenguajes formales.

Sea G una gramática y sea λ la cadena nula.

Gramática Tipo 0

No se establecen restricciones sobre las producciones de G. Son producciones de cualquier tipo.

Por ser muy generales pierden fuerza descriptiva. Sus cadenas no son interesantes como lenguajes artificiales, pero si para las reglas de deducción en lógica.

Gramática Tipo 1

Si cada producción es de la forma $\alpha A \beta \rightarrow \alpha \delta \beta$, donde $\alpha, \beta \in (N \cup T)^*, A \in N, \delta \in (N \cup T)^* - \{\lambda\}$, G recibe el nombre de ***gramática sensible al contexto*** (o ***tipo 1***).

Es decir, una gramática es de tipo 1, si para cualquier producción $A \rightarrow B$, la longitud de A es menor o igual que la longitud de B (donde la **longitud** de una cadena es el número de palabras en esa cadena).

Gramática Tipo 2

Si cada producción es de la forma $A \rightarrow \delta$, donde $A \in N, \delta \in (N \cup T)^*$, G recibe el nombre de ***gramática libre de contexto*** (o ***tipo 2***).

Es decir, una gramática es de tipo 2, si el lado izquierdo de cada producción es un único símbolo no terminal y el lado derecho consta de uno o más símbolos.

Las gramáticas de tipo 2 se usan frecuentemente en lenguajes artificiales, en especial en lenguajes de programación.

Gramática Tipo 3

Si cada producción es de la forma $A \rightarrow a \; o \; A \rightarrow aB \; o \; A \rightarrow \lambda$, donde $A, B \in N, a \in T$, G recibe el nombre de ***gramática regular*** (o ***tipo 2***).

Es decir, una gramática es de tipo 3, si el lado izquierdo de cada producción es un único símbolo no terminal y el lado derecho tiene uno o más símbolos, incluyendo como máximo un solo símbolo no terminal, que debe estar en el extremo derecho de la cadena.

Los tipos de gramática en la jerarquía de Chomsky son abarcativos, es decir:

$$Tipo\ 3 \subset Tipo\ 2 \subset Tipo\ 1 \subset Tipo\ 3$$

Forma de Backus-Naur

Una manera alternativa para establecer la producción de una gramática es una la ***forma regular de Backus*** (o ***forma de Backus-Naur*** o ***BNF***). En la forma BNF, los símbolos no terminales suelen comenzar con "<" y terminar con ">". La producción $S \rightarrow T$ se escribe $S ::= T$. Las producciones de la forma $S ::= T_1, S ::= T_2, \ldots, S ::= T_n$ se puede combinar como $S ::= T_1 | T_2 | \ldots | T_n$.

La barra "|" se lee como "o".

Definición de MEF

Una máquina de estado finito es una máquina capaz de entregar señales de salida en función de las señales de entrada y de "lo que sucedió antes". Son máquinas con "memoria".

Definición:

Una ***máquina de estado finito*** es un modelo abstracto de una máquina con una memoria interna primitiva.

Una *máquina de estado finito M* consiste en:

a) Un conjunto finito *I* de ***símbolos de entrada***.
b) Un conjunto finito *O* de ***símbolos de salida***.
c) Un conjunto finito *S* de ***estados***.
d) Una ***función f del siguiente estado*** de $S \times I$ en *S*.
e) Una ***función g de salida*** de $S \times I$ en *S*.
f) Un ***estado inicial*** $\sigma \in S$.

Se escribe $M = (I, O, S, f, g, \sigma)$.

Diagrama de Estado o Transición

Sea $M = (I, O, S, f, g, \sigma)$ una máquina de estado finito. El ***diagrama de transición*** de *M* es un dígrafo *G* cuyos vértices son los miembros de *S*. Una flecha designa el estado inicial σ. Se tiene una arista dirigida (σ_1, σ_2) en *G* si existe una entrada *i* con $f(\sigma_1, i) = \sigma_2$. En este caso, si $g(\sigma_1, i) = o$, la arista (σ_1, σ_2) tiene etiqueta *i/o*.

		f		g	
S	I	a	b	a	b
σ_0		σ_0	σ_1	0	1
σ_1		σ_1	σ_1	1	0

Tabla de estados

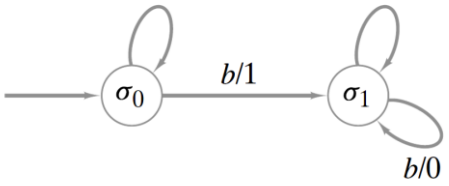


Diagrama de transición

Se puede ver la máquina de estado finito $M = (I, O, S, f, g, \sigma)$ como una computadora sencilla. Iniciamos en el estado σ, introducimos una cadena sobre *I* y producimos una cadena de salida.

Se puede representar una MEF por medio de una ***tabla de estados o transición*** que luego traduciremos al diagrama de estado o transición.

Ejemplo: Sean $I = \{a, b\}, O = \{0, 1\}, y\ S = \{\sigma_1, \sigma_2\}$

Tipos de Máquinas de Estado Finito

El tipo de maquina cuyas salidas corresponden a transiciones de entre estados se llama ***máquina de Mealy*** o máquina con salida asociada a la transición.

Otro tipo de máquina cuya salida está determinada sólo por el estado se llama ***máquina de Moore*** o máquina con salida asociada al estado.

MEF como modelo de sistema físico

Autómatas de estado finito deterministas y no deterministas

Existen ciertas máquinas de estado finito que están especialmente diseñadas para el reconocimiento de lenguajes. En lugar de producir una salida, estas máquinas tienen estados finales. Una cadena es reconocida por la máquina sí, y sólo si, se produce una transición del estado inicial a uno de estos estados finales.

Estas máquinas de estado finito sin salida también son conocidas como autómatas finitos o autómatas de estado finito.

Definición:

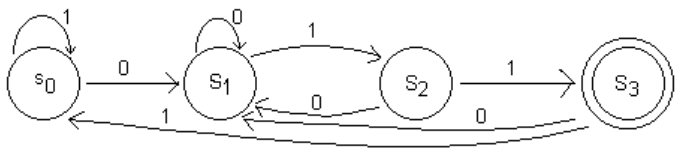
Un ***autómata finito*** $M = (S, I, f, s_0, F)$ consiste en un conjunto finito de *estados S*; un *alfabeto fuente I*, que contiene los símbolos de entrada; una función de transición *f*, que asigna a cada par de estado y entrada el estado siguiente (esto es, $f: S \times I \to S$); un estado inicial s_0, y un subconjunto *F* de *S* de *estados finales*.

Podemos representar un autómata finito utilizando los diagramas como las tablas de estados.

En los diagramas de estados representaremos los estados finales rodeados por círculos dobles

Autómata de estado finito determinista (AEFD)

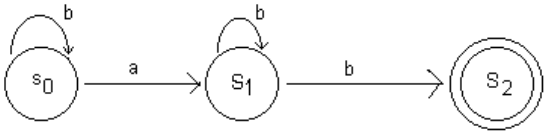
Los autómatas finitos cuyas funciones de transición asignan a cada par de estado y entrada un único estado siguiente, son conocidos como ***deterministas***.



Autómata de estado finito no determinista (AEFND)

Los autómatas finitos que pueden asignar varios posibles estados siguientes a cada pareja de entrada y estado son conocidos como ***no deterministas***. Éstos son importantes para determinar qué lenguajes pueden ser reconocidos por un autómata.

Es decir, un ***autómata finito no determinista*** $M = (S, I, f, s_0, F)$ consiste en un conjunto finito de *estados S*; un *alfabeto fuente I*, que contiene los símbolos de entrada; una función de transición *f*, que asigna un conjunto de estados a cada par de estado y entrada (esto es, $f: S \times I \to P(S)$); un estado inicial s_0, y un subconjunto *F* de *S* de *estados finales*.



Equivalencias de AEFD y AEFND

Teorema:

Si el lenguaje L es reconocido por un autómata finito no determinista M_0, entonces L también es reconocido por un autómata finito determinista M_1.

Sea $M_0 = (I, S, f, s_0, F)$ un autómata de estado finito no determinista. Sea

a) $S' = P(S)$ (subconjunto de partes de S)
b) $I' = I$
c) ${s_0}' = \{s_0\}$
d) $F' = \{X \subseteq S \mid X \cap F \neq \emptyset\}$
e) $f'(X, x) = \begin{cases} \emptyset & si\ X = \emptyset \\ \bigcup_{S \in X} f(S, x) & si\ X \neq \emptyset \end{cases}$

Entonces el autómata de estado finito $M_1 = (I', S', f', s_0', F')$ es equivalente a M_0.

Explicación:

Por lo tanto, un AEFND se diferencia de un AEFD en que el primero puede pasar de un estado, con la introducción de un símbolo de entrada, a uno o más estados, por lo que la imagen de la función de transición de estados puede ser $P(S)$; para ello, vamos a "redefinir" cualquier AEFND en un AEFD.

1) Definiendo un "nuevo" conjunto de estados con los elementos de $P(S)$.
2) Una nueva función de transición de estados $F: I \times P(S) en\ P(S)$

$$\begin{cases} \hat{F}(i_q, \{\}) = \{\}, & \forall i_q \\ \hat{F}(i_q, S_p) = \bigcup_{S_i \in S} f(i_q, S_i) = S_r & S_r \in S_p\ con\ S_i \neq \emptyset \end{cases}$$

   En cada elemento del dominio de la nueva función, S_p conformado por los estados S_r, para un mismo símbolo de entrada i_q; la imagen será la unión de los codominios que les correspondan a cada dominio (i_q, S_r) en la relación de transición de AEFND original.
3) Una nueva función de salida de $P(S) \times I\ en\ O$ definida

$$\begin{cases} \forall S_r \subset S_p, \hat{G}(i_q, S_r) = 1 \Leftrightarrow \exists S_i \in S_p\ tal\ que\ g(S_i) = 1 \\ \hat{G}(i_q, S_r) = 0 \quad en\ cualquier\ otro\ caso \end{cases}$$

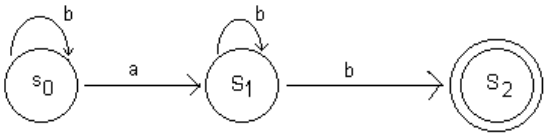
   Si el nuevo estado S_r contiene algún estado que en la relación de salidas del AEFND es de aceptación $(O = 1)$, el nuevo estado también es de aceptación $(O = 1)$.

Luego, redefinimos un AEF ND mediante:

$M = (I, O, S, s_0, F, G)$ donde:

a) *I* es un conjunto finito de *símbolos de entrada* $I = \{i_0, i_1, \dots, i_n\}$
b) *O* es un conjunto finito de *símbolos de salida* $O = \{0,1\}$
c) *S* es un conjunto finito de *estados* $S = \{s_0, S_1, S_2, \dots\}$
d) Un *estado inicial* s_0.
e) Una *función de transición de estados* $F: I \times P(S) \rightarrow P(S)$, que tiene como imagen el conjunto de partes de *S*.
f) Una *función de salida* $G = I \times P(S) \rightarrow O$

Ejemplo:



Los símbolos de entrada, de salida y el estado inicial no cambian, siguen siendo $I = \{a, b\}, O = \{0,1\}, s_0$

El conjunto de estados $S = \{s_0, S_1, S_2\}$ será ahora $P(S)$, donde:

$$P(S) = \{\{s_0\}; \{S_1\}; \{S_2\}; \{s_0, S_1\}; \{s_0, S_2\}; \{S_1, S_2\}; \{s_0, S_1, S_2\}, \emptyset\}$$

Los estados de aceptación en el nuevo autómata serán todos aquellos que contengan el estado de aceptación del AEFND que estamos estudiando (S_2 en este caso)

$$\{S_2\}; \{s_0, S_2\}; \{S_1, S_2\}; \{s_0, S_1, S_2\}$$

En el AEFD al estado $\{s_0, S_1\}$ con el símbolo de entrada *b*, le corresponderá el estado $\{s_0, S_1, S_2\}$

$$f(b \times \{s_0, S_1\}) = \{s_0, S_1, S_2\}$$

Porque en el AEFND:

1. s_0 con entrada b, le corresponde s_0, es decir, $f(b \times s_0) = s_0$
2. S_1 con entrada b, le corresponde S_1 ó S_2, es decir, $f(b \times S_1) = S_1$ y $f(b \times S_1) = S_2$

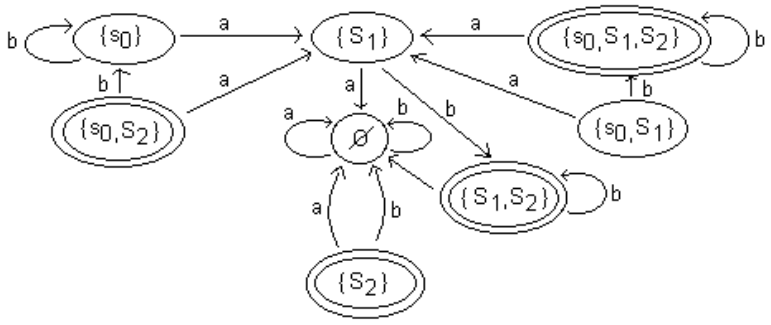
Así, las funciones de transición quedarán de la siguiente manera:

$$f(a \times \{s_0\}) = \{S_1\} \qquad f(b \times \{S_2\}) = \emptyset \qquad f(a \times \{S_1, S_2\}) = \emptyset$$
$$f(b \times \{s_0\}) = \{s_0\} \qquad f(a \times \{s_0, S_1\}) = \{S_1\} \qquad f(b \times \{S_1, S_2\}) = \{S_1, S_2\}$$
$$f(a \times \{S_1\}) = \emptyset \qquad f(b \times \{s_0, S_1\}) = \{s_0, S_1, S_2\} \qquad f(a \times \{s_0, S_1, S_2\}) = \{S_1\}$$
$$f(b \times \{S_1\}) = \{S_1 S_2\} \qquad f(a \times \{s_0, S_2\}) = \{S_1\} \qquad f(b \times \{s_0, S_1, S_2\}) = \{s_0, S_1, S_2\}$$
$$f(a \times \{S_2\}) = \emptyset \qquad f(b \times \{s_0, S_2\}) = \{s_0\}$$

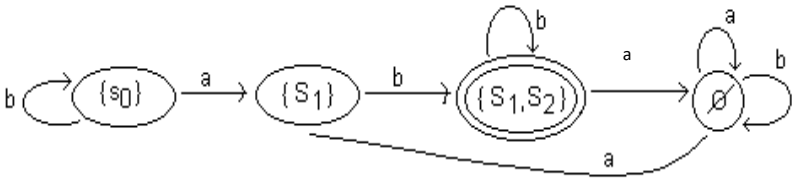
Se representará mediante la siguiente tabla de estados:

Estados	a	b	Salida
$\emptyset$	$\emptyset$	$\emptyset$	0
$\{s_0\}$	$\{S_1\}$	$\{s_0\}$	0
$\{S_1\}$	$\emptyset$	$\{S_1 S_2\}$	0
$\{S_2\}$	$\emptyset$	$\emptyset$	1
$\{s_0, S_1\}$	$\{S_1\}$	$\{s_0, S_1, S_2\}$	0
$\{s_0, S_2\}$	$\{S_1\}$	$\{s_0\}$	1
$\{S_1, S_2\}$	$\emptyset$	$\{S_1 S_2\}$	1
$\{s_0, S_1, S_2\}$	$\{S_1\}$	$\{s_0, S_1, S_2\}$	1

Y el diagrama de estados quedará de la siguiente manera:



En el diagrama se puede ver que los estados $\{S_2\}; \{s_0, S_1\}; \{s_0, S_2\}; \{s_0, S_1, S_2\}$ nunca se pueden alcanzar, en consecuencia, estos se pueden eliminar (ya que nunca serán usados en una decisión si no es posible alcanzarlos), quedan de la siguiente forma reducida:



Representación de gramáticas regulares en autómatas.