0. Introducción (recordatorio)

0.1. Lógica preposicional

La siguiente tabla muestra propiedades básicas a partir de las que podemos ir de una fórmula a otra para poder demostrar que dos fórmulas son equivalentes.

Distributiva	$\varphi \wedge (\psi \vee \theta) \equiv (\varphi \wedge \psi) \vee (\varphi \wedge \theta)$	$\varphi \vee (\psi \wedge \theta) \equiv (\varphi \vee \psi) \wedge (\varphi \vee \theta)$
De Morgan	$\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$	$\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$
Absorción	$\varphi \wedge (\varphi \vee \psi) \equiv \varphi$	$\varphi \vee (\varphi \wedge \psi) \equiv \varphi$
Idempotencia	$\varphi \wedge \varphi \equiv \varphi$	$\varphi \vee \varphi \equiv \varphi$
Conmutativa	$\varphi \wedge \psi \equiv \psi \wedge \varphi$	$\varphi \vee \psi \equiv \psi \vee \varphi$
Asociativa	$\varphi \wedge (\psi \wedge \theta) \equiv (\varphi \wedge \psi) \wedge \theta$	$\varphi \vee (\psi \vee \theta) \equiv (\varphi \vee \psi) \vee \theta$
Neutra	$\varphi \wedge 1 \equiv \varphi$	$\varphi \vee 0 \equiv \varphi$
	$\varphi \vee 1 \equiv 1$	$\varphi \wedge 0 \equiv 0$
Complementaria	$\varphi \vee \neg\varphi \equiv 1$	$\varphi \wedge \neg\varphi \equiv 0$
Doble negación	$\neg\neg\varphi \equiv \varphi$	
	$\neg 1 \equiv 0$	$\neg 0 \equiv 1$
Traducción de la $\rightarrow$	$\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$	$\neg(\varphi \rightarrow \psi) \equiv \varphi \wedge \neg\psi$
Traducción de la $\leftrightarrow$	$\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$ $\varphi \leftrightarrow \psi \equiv (\varphi \wedge \psi) \vee (\neg\varphi \wedge \neg\psi)$	$\neg(\varphi \leftrightarrow \psi) \equiv (\varphi \wedge \neg\psi) \vee (\neg\varphi \wedge \psi)$

0.2. Lógica de predicados

Cuantificadores:

- $\forall x\varphi \equiv$ Todos los individuos x del dominio cumplen φ.
- $\exists x\varphi \equiv$ Existe algún (al menos un) individuo x del dominio que cumpla φ.

Equivalencia

$\neg\forall x\varphi \equiv \exists x\neg\varphi$	$\neg\exists x\varphi \equiv \forall x\neg\varphi$
$\forall x\forall y\varphi \equiv \forall y\forall x\varphi$	$\exists x\exists y\varphi \equiv \exists y\exists x\varphi$
$\forall x(\varphi \wedge \psi) \equiv \forall x\varphi \wedge \forall x\psi$	$\exists x(\varphi \vee \psi) \equiv \exists x\varphi \vee \exists x\psi$

0.3. Demostraciones

Pasos lógicos (sirven para que en cualquier momento de una demostración puedan ser usados):

Pasos lógicos			Tautologías
$A,\ B$	$\Rightarrow$	A	$(p \wedge q) \rightarrow p$
A	$\Rightarrow$	$A\ o\ B$	$p \rightarrow (p \vee q)$
$A\ o\ B,\ no\ A$	$\Rightarrow$	B	$\big((p \vee q) \wedge \neg p\big) \rightarrow q$
$A,\ A \Rightarrow B$	$\Rightarrow$	B	$\big(p \wedge (p \rightarrow q)\big) \rightarrow q$
$no\ B,\ A \Rightarrow B$	$\Rightarrow$	$no\ A$	$\big(\neg q \wedge (p \rightarrow q)\big) \rightarrow \neg p$
$ABSURDO$	$\Rightarrow$	A	$0 \rightarrow p$
$A \Rightarrow B,\ B \Rightarrow C$	$\Rightarrow$	$A \Rightarrow C$	$\big((p \rightarrow q) \wedge (q \rightarrow r)\big) \rightarrow (p \rightarrow r)$

Nota: la coma se usa como si fuera ∧ (una conjunción)
Tautología: fórmula siempre cierta

Prueba directa

Salimos de la hipótesis A i llegamos a la tesis B. Esto se hace mediante pequeñas implicaciones que han de ser muy claras, estas implicaciones pueden ser, por ejemplo, los pasos anteriores.

Queremos demostrar $A \Rightarrow B$
$A \Rightarrow A' \Rightarrow A'' \Rightarrow \cdots \Rightarrow B$

Prueba del contrarrecíproco

Se basa en: $p \rightarrow q \equiv \neg q \rightarrow \neg p$

Queremos demostrar $A \Rightarrow B$
$\neg B \Rightarrow \cdots \Rightarrow \neg A$

Reducción al absurdo

Se basa en: $p \equiv \neg p \rightarrow 0$

Queremos demostrar A
$\neg A \Rightarrow \cdots \Rightarrow Contradicción$

Reducción al absurdo II

Se basa en: $p \rightarrow q \equiv (p \wedge \neg q) \rightarrow 0$

Queremos demostrar $A \Rightarrow B$
$A, \neg B \Rightarrow \cdots \Rightarrow Contradicción$

Prueba de una disyunción

Se basa en: $(q \vee r) \equiv (\neg q \rightarrow r)$

Queremos demostrar $B \vee C$

$\neg B \Rightarrow \cdots \Rightarrow C$

En caso de más disyuntandos: $p_1 \vee \ldots \vee p_n \equiv (\neg p_1 \vee \ldots \vee \neg p_{n-1}) \rightarrow p_n$

Queremos demostrar $B_1 \vee \ldots \vee B_n$
$\neg B_1, \neg B_2, \ldots, \neg B_{n-1} \Rightarrow \cdots \Rightarrow B_n$

Disyunción al consecuente

Se basa en: $p \rightarrow (q \vee r) \equiv (p \wedge \neg q \rightarrow r)$

Queremos demostrar $A \Rightarrow (B \vee C)$
$A, \neg B \Rightarrow \cdots \Rightarrow C$

Con más disyuntandos: $p \rightarrow (q_1 \vee \ldots \vee q_n) \equiv (p \wedge \neg q_1 \wedge \ldots \wedge \neg q_{n-1}) \rightarrow q_n$

Queremos demostrar $A \Rightarrow (B_1 \vee \ldots \vee B_n)$
$A, \neg B_1, \neg B_2, \ldots, \neg B_{n-1} \Rightarrow \cdots \Rightarrow B_n$

Prueba por casos

Se basa en la tautología: $(p_1 \vee \ldots \vee p_n) \rightarrow \big(p \leftrightarrow (p_1 \rightarrow p) \wedge \ldots \wedge (p_n \rightarrow p)\big)$

Queremos demostrar B, distinguimos los casos $A_1, \ldots, A_n$
Caso 1: A_1 $A_1 \Rightarrow \cdots \Rightarrow B$ … Caso n: A_n $A_n \Rightarrow \cdots \Rightarrow B$

Nota: hay que hacerlo con todos los casos

Disyunción al antecedente

Se basa en: $(q \vee r) \rightarrow p \equiv (q \rightarrow p) \wedge (r \rightarrow p)$

Es equivalente a hacer una prueba por casos (distinguimos según B o C).

Queremos demostrar $(B \vee C) \Rightarrow A$
$B \Rightarrow \cdots \Rightarrow A$ $C \Rightarrow \cdots \Rightarrow A$

Cuando hay más casos:

Queremos demostrar $(B_1 \vee \ldots \vee B_n) \Rightarrow A$
$B_1 \Rightarrow \cdots \Rightarrow A$ …

$B_n \Rightarrow \cdots \Rightarrow A$

Demostración de una equivalencia

Se basa en: $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$

Queremos demostrar $A \Leftrightarrow B$
$A \Rightarrow B$ $B \Rightarrow A$

Cuando hay más casos:

Queremos demostrar que $A_1,\ A_2, \dots, A_n$ son equivalentes (dos a dos)
$A_1 \Rightarrow A_2$ $A_2 \Rightarrow A_3$ ... $A_{n-1} \Rightarrow A_n$ $A_n \Rightarrow A_1$

Nota: Se puede cambiar el orden de los enunciados $A_1, A_2, \dots, A_n$

Demostración por unicidad

Cuando decimos que "hay como mucho una x que satisface $P(x)$" o bien "si hay una x que satisface $P(x)$ este es único", estamos expresando: $\forall x, y (P(x) \wedge P(y) \rightarrow x = y)$

Queremos ver que hay como mucho una x tal que $P(x)$.
$P(x), P(y) \Rightarrow \cdots \Rightarrow x = y$

Nota: No afirmamos que el elemento x exista, sólo que no hay dos diferentes o, mejor dicho, que hay como mucho uno (puede ser que no haya ninguno).

Nota: Cuando queramos ver que "hay una única x tal que $P(x)$" habrá que ver dos cosas: que existe el elemento x y que es único.

Inducción

Se usa cuando la variable n depende de una infinidad de enteros (una sucesión de números).

Se basa en: $\forall n \geq n_0\ P(n) \equiv P(n_0) \wedge \forall n > n_0 \big(P(n-1) \rightarrow P(n)\big)$

Queremos demostrar $\forall n \geq n_0\ P(n)$
$P(n_0)$ $\forall n > n_0 \ \big(P(n-1) \rightarrow P(n)\big)$

Se presenta así:

- Paso base $P(n_0)$ | Si es necesario más de un caso inicial: $P(n_0), \dots, P(n_i)$
- Paso inductivo Sea $n > n_0$:
  - Hipótesis de inducción: $P(n-1)$

Queremos ver (tesis): $P(n)$ Procedemos:

Ejemplo: $\sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{n}{n+1}$ para $n \geq 1$

Queremos demostrar: $\forall n \geq 1 \sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{n}{n+1}$

- Paso base: $n = 1$

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

- Paso inductivo $n > 1$:
  - H.I. sustituimos n por $n-1$ $\frac{n-1}{(n-1)+1} = \frac{n-1}{n}$

Queremos ver: $\sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{n}{n+1}$

Identidad notable: $(a+b)(a-b) = a^2 - b^2$

Procedemos:

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

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

1. Lenguajes formales

1.1. Alfabetos y motes

Un **alfabeto** es un conjunto finito no vacío de unos elementos que llamamos **símbolos**. Ejemplos:

- Alfabeto latín $\{A, \dots, Z\}$ $\Sigma_1 = \{a, b, c, \dots\}$
- Alfabeto decimal $\{0, \dots, 9\}$ $\Sigma_2 = \{0, 3, 9\}$
- Alfabeto binario $\{0,1\}$ $\Sigma_3 = \{0,1\}$
- Alfabeto ASCII

Usamos Σ para referirnos a un alfabeto cualquiera. Un **mote** es una secuencia finita de símbolos yuxtapuestas de este alfabeto. La longitud de un mote es el número de símbolos que lo componen. Ejemplos prácticos:

- Decimos **palabra vacía** a un mote que no esta representado por ningún símbolo: $w = \lambda$
- Representación de un mote: $w_1 = ab$, $w_2 = ababa$
- Representación de la longitud de un mote: $|w|$
- Representación del número de repeticiones de una secuencia de símbolos:
  - $|w_1|_a = 1$ Número de veces que tenemos a en w_1
  - $|w_2|_{aba} = 2$ Número de veces que tenemos aba en w_2
- Representamos la cardinalidad o número de motes en un conjunto finito: $\|C\|$

Llamamos **submote** o factor a cualquier subcadena de símbolos consecutivos de estos motes. Los **prefijos** y **sufijos** son casos particulares de estos submotes, cuando se encuentran al principio o l final del mote, respectivamente. Un factor se denomina **propio** cuando no es λ ni el mote considerado. Ejemplo:

Tenemos el mote aba

El conjunto de factores es el siguiente: $\{\lambda, a, b, ab, ba, aba\}$

De los cuales prefijos: $\{\lambda, a, ab, aba\}$

De los cuales sufijos: $\{\lambda, a, ba, aba\}$

Representamos Σ^* como el conjunto infinito de todos los motes posibles sobre un cierto alfabeto Σ. Ordenamos por su longitud y orden lexicográfico estos conjuntos. Por ejemplo, por $\Sigma = \{a, b\}$ y considerando que $a < b$, obtenemos el siguiente orden:

$$\lambda < a < b < aa < ab < ba < bb < aaa < \cdots$$

1.2. Operaciones con motes

1.2.1. Concatenación

Es la yuxtaposición del primer y segundo mote. Ejemplo: si $w_1 = abb$, $w_2 = ba$, entonces, $w_1 w_2 = abbba$

Podemos ver que es un tipo de monoide, es decir, es una estructura formada por un conjunto y la operación es asociativa $\big((w_1 \cdot w_2) \cdot w_3 = w_1 \cdot (w_2 \cdot w_3)\big)$ y tiene elemento neutro (λ). Por lo que si un elemento x en la ecuación aparece multiplicando por la izquierda o por la derecha podemos simplificar la ecuación borrando el elemento x, reduciendo la igualdad.

$$\forall y, z \in M \quad \begin{cases} x \cdot y = x \cdot z \Longrightarrow y = z \\ y \cdot x = z \cdot x \Longrightarrow y = z \end{cases}$$

Potencias con motes:

- $w^n = w \cdot w \cdot \ldots \cdot w \qquad = w \cdot w^{n-1}, \ n \geq 1$
- $w^2 = w \cdot w$
- $w^1 = w$
- $w^0 = \lambda$

1.2.2. Revesado

Llamamos revesado de un mote w el mote formado por los mismos símbolos de w escritos al revés. Lo representamos como w^R. Formalmente:

$$\lambda^R = \lambda, \ \ (a_1 a_2 \ldots a_n)^R = a_n \ldots a_2 a_1$$

Por todo mote w, se obtiene que $(w^R)^R = w$. Un palíndromo es un mote que es igual a su revesado. Una definición recursiva del palíndromo es la siguiente:

- λ es un palíndromo
- Todos los motes formados por un solo símbolo son palíndromos.
- Si w es un palíndromo, entonces por todo símbolo a el mote awa también es palíndromo.
- No hay otros palíndromos que los obtenidos por las reglas anteriores.

1.3. Lenguajes

Un **lenguaje** es un conjunto cualquiera (finito o infinito) de motes sobre un alfabeto determinado. Así, sobre el alfabeto $\Sigma = \{a, b, c\}$ podemos considerar el lenguaje formado por todos los motes que tienen el mismo nombre de a's, b's y c's. Formalmente:

$$L = \{w \mid |w|_a = |w|_b = |w|_c\} = \{\lambda, abc, acb, bac, bca. cab. cba, \ldots\}$$

Otro ejemplo, es que el lenguaje contenga un número de a's par:

$$L = \{w \in \{a, b, c\}^* \mid |w|_a \in 2\}$$

Dónde $w \in \{a, b, c\}^*$ hace referencia a todas las posibles combinaciones (infinitas) entre los símbolos a, b y c, y $|w|_a \in 2$ a que el número a's en el mote ha de ser par para pertenecer al lenguaje L.

La concatenación de lenguajes es el lenguaje formado por todos los motes obtenidos concatenando los motes del primer lenguaje con los del segundo. Ejemplo:

Sea $L_1 = \{a, ab, baa\}$ y $L_2 = \{ab, ba\}$

Se obtiene $L_1 L_2 = \{aab, aba, abab, abba, baaab, baaba\}$

Otro ejemplo con lenguajes con motes infinitos:

Sea $L_1 = \{a\}^*$ y $L_2 = \{b\}^*$

Se obtiene $L_1 L_2 = \{\lambda, a, b, aa, ab, bb, \ldots\}$

A diferencia del monoide formado por los motes sobre un cierto alfabeto Σ, en el cual todos los elementos son simplificables, el monoide formado por lenguajes no cumple esta propiedad. Decimos que un lenguaje L es simplificable por la derecha cuando satisface la condición:

$$\forall L_1, L_2 \quad L_1 \cdot L = L_2 \cdot L \Rightarrow L_1 = L_2$$

Simétricamente, la condición de ser simplificable por la izquierda:

$$\forall L_1, L_2 \quad L \cdot L_1 = L \cdot L_2 \Rightarrow L_1 = L_2$$

Un ejemplo:

$$\{\lambda, a, a^2\} \cdot \{\lambda, a\} = \{\lambda, a^2\} \cdot \{\lambda, a\}$$

Igualdades que se pueden deducir de la definición de concatenación:

$$L \cdot \emptyset = \emptyset \cdot L = \emptyset$$

$$L \cdot \lambda = \lambda \cdot L = L$$

Utilizamos la notación exponencial para representar la concatenación repetida de un lenguaje consigo mismo.

$$L^0 = \{\lambda\} \quad i \quad \forall i \geq 0 \quad L^{i+1} = L^i L$$

1.4. Otras operaciones con lenguajes

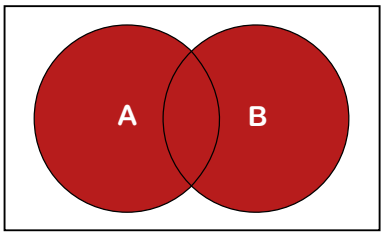
Los lenguajes, como los conjuntos, admiten las operaciones propias de los conjuntos:

1.4.1. Unión

Se puede expresar de dos formas:

$$A \cup B = \{x \mid x \in A \vee x \in B\}$$

$$x \in A \cup B \Leftrightarrow x \in A \vee x \in B$$



Propiedades:

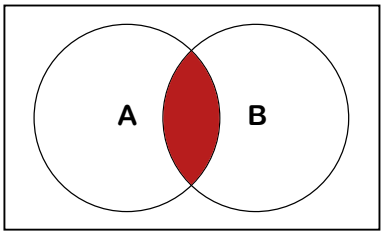
1. $A \cup A = A$
2. $A \cup \emptyset = A$
3. $A \cup B = B \cup A$
4. $A \cup (B \cup C) = (A \cup B) \cup C$
5. $A \subseteq A \cup B,\ B \subseteq A \cup B$
6. $A \subseteq B \Leftrightarrow A \cup B = B$
7. $A \cup B \subseteq C \Leftrightarrow A \subseteq C,\ B \subseteq C$

1.4.2. Intersección

Se puede expresar de dos formas:

$$A \cap B = \{x \mid x \in A \wedge x \in B\}$$

$$x \in A \cap B \Leftrightarrow x \in A \wedge x \in B$$



Propiedades:

1. $A \cap A = A$
2. $A \cap \emptyset = \emptyset$
3. A$\cap B = B \cap A$
4. $A \cap (B \cap C) = (A \cap B) \cap C$
5. $A \cap B \subseteq A,\ A \cap B \subseteq B$

6. $A \subseteq B \Leftrightarrow A \cap B = A$
7. $C \subseteq A \cap B \Leftrightarrow C \subseteq A \wedge C \subseteq B$

Cuando dos conjuntos A, B no tienen elementos comunes se dice que son disjuntos:

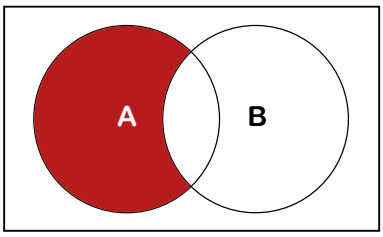
$$A\ i\ B\ son\ disjuntos \Leftrightarrow A \cap B = \emptyset$$

1.4.3. Diferencia

Se puede expresar de dos formas:

$$A - B = \{x \mid x \in A \wedge x \notin B\}$$

$$x \in A - B \Leftrightarrow\ x \in A \wedge x \notin B$$



Propiedades:

1. $A - A = \emptyset$
2. $A - \emptyset = A$
3. $\emptyset - A = \emptyset$
4. $A - B \subseteq A$
5. $(A - B) \cap B = \emptyset$
6. $A \subseteq B \Leftrightarrow A - B = \emptyset$
7. $C \subseteq A - B \Leftrightarrow C \subseteq A \wedge C \cap B = \emptyset$

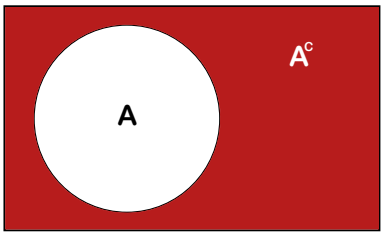
1.4.4. Complementario

Suponemos que hay un conjunto *grande* o *universo* Ω que engloba a todos los elementos.

Se puede expresar de dos formas:

$$A^C = \Omega - A = \{x \in \Omega \mid x \notin A\}$$

$$x \in A^C \Leftrightarrow\ x \in \Omega \wedge x \notin A$$



Propiedades:

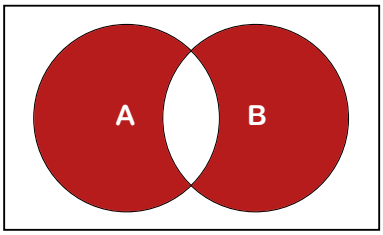
1. $(A^C)^C = A$
2. $\emptyset^C = \Omega,\ \Omega^C = \emptyset$
3. $A \cap A^C = \emptyset,\ A \cup A^C = \Omega$
4. $(A \cup B)^C = A^C \cap B^C, (A \cap B)^C = A^C \cup B^C$ (De Morgan)
5. $A - B = A \cap B^C$
6. $B = A^C \Leftrightarrow A \cap B = \emptyset,\ A \cup B = \Omega$
7. $A \subseteq B \Leftrightarrow B^C \subseteq A^C \Leftrightarrow A \cap B^C = \emptyset \Leftrightarrow A^C \cup B = \Omega$
8. $A \subseteq B^C \Leftrightarrow B \subseteq A^C \Leftrightarrow A \cap B = \emptyset \Leftrightarrow A^C \cup B^C = \Omega$
9. $A^C \subseteq B \Leftrightarrow B^C \subseteq A \Leftrightarrow A^C \cap B^C = \emptyset \Leftrightarrow A \cup B = \Omega$

1.4.5. Diferencia simétrica

Se puede expresar de dos formas:

$$A \triangle B = \{x \mid (x \in A \wedge x \notin B) \vee (x \in B \wedge x \notin A)\}$$

$$x \in A \triangle B \Leftrightarrow (x \in A \wedge x \notin B) \vee (x \in B \wedge x \notin A)$$



Propiedades:

1. $A \triangle B = (A - B) \cup (B - A)$
2. $A \triangle B = (A \cup B) - (A \cap B)$

A continuación, definiremos operaciones específicas de los lenguajes y de conjuntos de motes:

1.4.6. Clausura de Kleene

Llamamos clausura de Kleene (también llamada estrella de Kleene o cierre estrella) de un lenguaje L y representado por L^*, el lenguaje siguiente:

$$L^* = \bigcup_{n \geq 0} L^n$$

Se trata del lenguaje formado por λ y todas las concatenaciones posibles de los motes de L. Es, por ello, el mínimo monoide que contiene L. Ejemplos:

- $\{a\}^* = \{\lambda, a, aa, aaa, aaaa, aaaaa, \dots\}$
- $\{ab, c\}^* = \{\lambda\ , ab, c, abab, abc, cab, cc, ababab, ababc, abcab, abcc, cabab, \dots\}$
- $\{\mathrm{a}, \mathrm{b}, \mathrm{c}\}^* = \{\lambda, \mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{aa}, \mathrm{ab}, \mathrm{ac}, \mathrm{ba}, \mathrm{bb}, \mathrm{bc}, \dots\}$

L^+ se diferencia de L^* porque esta no obliga necesariamente a incluir la palabra vacía. De modo que:

$L^* = \{\lambda\} \cup L^+$ $\quad$ L^* es la unión de L^+ y la palabra vacía

$L^+ = L^* \Leftrightarrow \lambda \in L \Leftrightarrow \lambda \in L^+$

$\mathrm{L}^*\mathrm{L} = L\mathrm{L}^* = \mathrm{L}^+\mathrm{L}^* = \mathrm{L}^*\mathrm{L}^+ = \mathrm{L}^+$

$L^*L^* = L^*$ pero $L^+L^+ = L^2L^*$

$(L^*)^* = (L^+)^* = (L^*)^+ = L^*$

$(L^+)^+ = L^+$

$\emptyset^* = \{\lambda\},\ \emptyset^+ = \emptyset$

1.4.7. Revesado

Llamamos revesado del lenguaje L, el lenguaje formado por los revesad0s de los motes de L. Representado por L^R, es decir:

$$L^R = \{w^R \mid w \in L\}$$

1.5. Morfismos y substituciones

1.5.1. Morfismos

Consideramos dos alfabetos Σ_1 y Σ_2. Llamamos morfismo a una aplicación de la forma:

$$h: \Sigma_1^* \longrightarrow \Sigma_2^*$$

que satisface la condición:

$$\forall x, y \in \Sigma_1^* \quad h(x \cdot y) = h(x) \cdot h(y)$$

Ejemplo:

Dados los alfabetos $\Sigma_1 = \{a, b, c, d\}$ y $\Sigma_2 = \{0, 1\}$, podemos definir un morfismo a partir de los valores $h(a) = 01$, $h(b) = 10$, $h(c) = 011$ y $h(d) = 110$. La imagen para este morfismo del mote $abbdd$ es 011010110110.

Como tratamos con monoides se cumple la condición $h(\lambda) = \lambda$

Propiedades elementales de los morfismos $h: \Sigma_1^* \longrightarrow \Sigma_2^*$

Si $L \subseteq \Sigma_1^*$	**Si $L \subseteq \Sigma_2^*$**
$h(L) = \emptyset \Leftrightarrow L = \emptyset$	$h^{-1}(\emptyset) = \emptyset$
$L_1 \subseteq L_2 \Rightarrow h(L_1) \subseteq h(L_2)$	$L_1 \subseteq L_2 \Rightarrow h^{-1}(L_1) \subseteq h^{-1}(L_2)$
$h(L_1 \cup L_2) = h(L_1) \cup h(L_2)$	$h^{-1}(L_1 \cup L_2) = h^{-1}(L_1) \cup h^{-1}(L_2)$
$h(L_1 \cap L_2) = h(L_1) \cap h(L_2)$	$h^{-1}(L_1 \cap L_2) = h^{-1}(L_1) \cap h^{-1}(L_2)$
$L \subseteq h^{-1}\big(h(L)\big)$	$h^{-1}\big(h(L)\big) \subseteq L$
	$h^{-1}\big(\overline{L}\big) = \overline{h^{-1}(L)}$

Otras propiedades:

- $h(L_1 \cdot L_2) = h(L_1) \cdot h(L_2)$
- $h(L^*) = \big(h(L)\big)^*$

1.5.2. Substituciones

Sean Σ_1 y Σ_2 dos alfabetos cualesquiera. Una substitución es una aplicación de la forma:

$$\sigma: \Sigma_1^* \longrightarrow P(\Sigma_2^*)$$

Que es el morfismo de monoides. Es decir, que si x e y son motes de Σ_1^*, y $\sigma(x)$, $\sigma(y)$ y $\sigma(xy)$ son los lenguajes de Σ_2^* correspondientes a x, y y xy, respectivamente, se obtiene:

$$\sigma(\lambda) = \{\lambda\}$$
$$\sigma(x) \cdot \sigma(y) = \sigma(xy)$$

Hay que remarcar que ahora $\sigma(x)$, $\sigma(y)$ y $\sigma(xy)$ son lenguajes y no motes como en el caso de los morfismos.

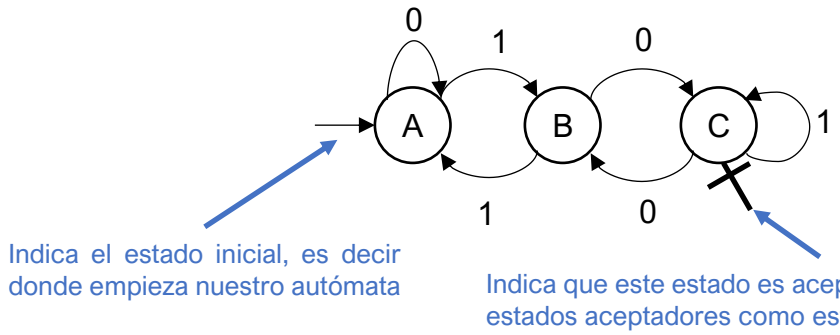
Propiedades:

- $\sigma(\{\lambda\}) = \sigma(\lambda) = \{\lambda\}$
- $\sigma(L_1 \cdot L_2) = \sigma(L_1) \cdot \sigma(L_2)$
- $\sigma(L_1 \cup L_2) = \sigma(L_1) \cup \sigma(L_2)$
- $\sigma(L_1^*) = \big(\sigma(L_1)\big)^*$

2. Autómatas finitos

Idea: leemos una palabra definida en un alfabeto de izquierda a derecha, una vez acabada la palabra si se encuentra en un estado aceptador la palabra, se da como válida dentro del lenguaje, en caso contrario, dicha palabra no se encuentra dentro del lenguaje.

Ejemplo de autómata representado con un diagrama de transiciones (dibujo):



Ejemplo de autómata representado con una tabla de transiciones:

	0	*1*	
A	A	B	
B	C	A	
C	B	C	+

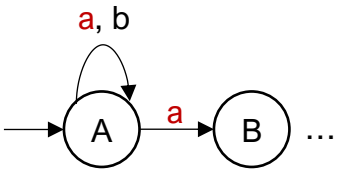
La primera columna representa el estado actual, en la segunda columna denotamos que pasa si desde el estado actual viene el símbolo 0, y de la misma forma la última columna denota que pasa si en el estado actual viene un 1, finalmente denotamos con una **cruz** los estados **aceptadores**.

La **primera fila** siempre corresponde al **estado inicial**.

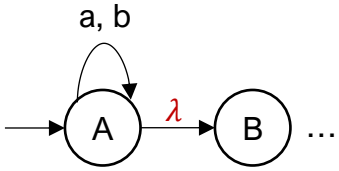
En el ejemplo anterior solo tenemos 0 y 1 como alfabeto de entrada, pero podría haber más símbolos.

Existen 3 tipos de autómatas finitos:

- Autómata finito determinista (DFA)
- Autómata finito indeterminista (NFA): cuando por un mismo símbolo en el mismo estado puede ir a parar a más de un estado a la vez.



- Autómatas finitos con λ-transiciones (λ-NFA): cuando permitimos al utómata cambiar de estado sin tener que leer ningún símbolo de entrada.

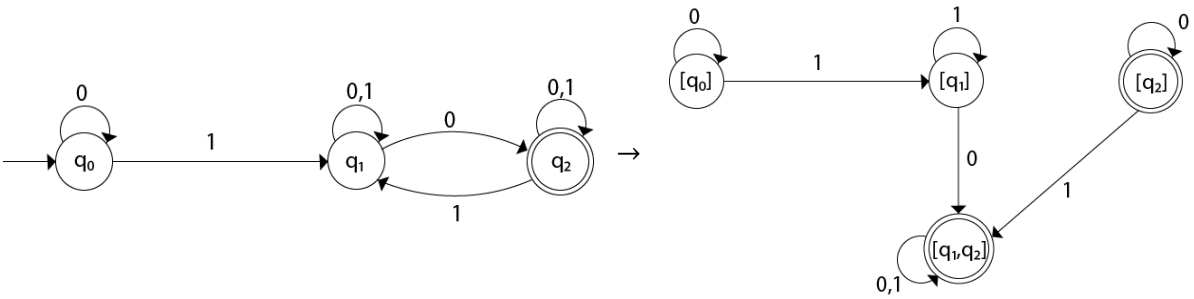


2.1. Pasar de NFA a DFA

Receta

1. Empezamos con el estado inicial.
2. Encontramos el conjunto de posibles símbolos para cada símbolo de entrada. Si este conjunto es nuevo lo añadimos al diagrama.
3. El estado final corresponde a todos los estados que contienen los estados finales del diagrama original

Ejemplo:



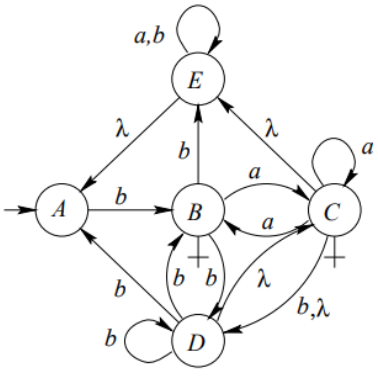
2.2. Pasar de λ-NFA a NFA

Receta:

- Creamos una tabla con todos los caminos con λ desde todos los nodos.
- Hacemos la unión de los estados de $\Lambda(q)$
- Los estados de aceptación del grafo NFA son todos aquellos estados finales originales a los que podemos ir $\Lambda(q)$ desde un estado q.

Ejemplo:

	a	*b*	λ	
A	-	B	-	
B	C	D, E	-	+
C	B, C	D	D, E	+
D	-	A,B,D	C	
E	E	E	A	



Construimos la tabla de los caminos de λ:

q	$\Lambda(q)$
A	A
B	**B**
C	A, **C**, D, E
D	A, **C**, D, E
E	A, E

Los estados finales antes eran B y C por lo que ahora son los estados B, C y D

Hacemos la unión

q	$\Lambda(q)$	a	b	
A	A	-	B	
B	B	C	D, E	+
C	A, C, D, E	B, C, E	A, B, D, E	+
D	A, C, D, E	B, C, E	A, B, D, E	+
E	A, E	E	B, E	

- Empezamos mirando desde A cuales son las posibilidades, es decir, de A si la entrada es una a no hacemos nada, si la entrada es una b, entonces nos movemos al estado B.
- Si estamos en el estado B y la entrada es una a vamos a C y si la entrada es una b vamos al estado D y E.
- Si estamos en el estado $ACDE$, vemos que si la entrada es una a y estamos en el estado A no hacemos nada, si estamos en el estado C vamos al estado BC si estamos en el estado D no hacemos nada y si estamos en el estado E vamos al E, por lo que podemos decir que si la entrada es a nos movemos al estado BCE, en cambio si la entrada es b y estamos en A nos movemos a B, estando en C nos movemos en D, si estamos en D a ABD y en E a E, por lo que si la entrada es b nos movemos a $ABDE$.
- $AE \rightarrow \begin{matrix} a \rightarrow \begin{matrix} A \rightarrow - \\ E \rightarrow E \end{matrix} \rightarrow E \\ b \rightarrow \begin{matrix} A \rightarrow B \\ E \rightarrow E \end{matrix} \rightarrow BE \end{matrix}$

2.3. Operaciones básicas con autómatas

2.3.1. Complementación

Para obtener el complementario de un autómata basta con cambiar los estados aceptadores por no aceptadores y los no aceptadores por aceptadores.

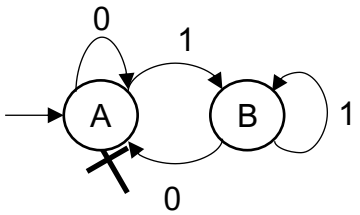
2.3.2. Producto cartesiano

Si L_1 es un lenguaje y L_2 otro lenguaje, el producto cartesiano

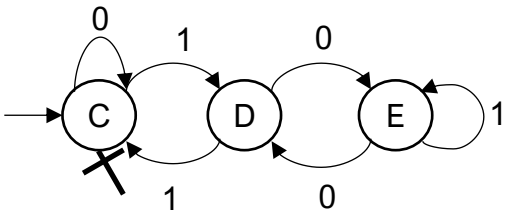
$$L_1 \times L_2 = \{w_{11}, w_{12}, \dots, w_{1n}\} \times \{w_{21}, w_{22}, \dots, w_{2m}\} = \\ = \begin{Bmatrix} (w_{11}, w_{21}), (w_{11}, w_{22}), \dots, (w_{11}, w_{2m}), \\ (w_{12}, w_{21}), (w_{12}, w_{22}), \dots, (w_{12}, w_{2m}), \\ \dots, \\ (w_{1n}, w_{21}), (w_{1n}, w_{22}), \dots (w_{1n}, w_{2m}) \end{Bmatrix}$$

- El estadio inicial es aquel formado por todos los estados iniciales originales.
- Para comprobar por cada estado de L_3 a que estado corresponde ir, en función del símbolo que venga, miramos el estado que corresponde de L_1 y L_2 y hacía que otros dos estados va.

Ejemplo:



	0	1	
A	A	B	+
B	A	B	



	0	1	
C	C	D	+
D	E	C	
E	D	E	

El producto cartesiano:

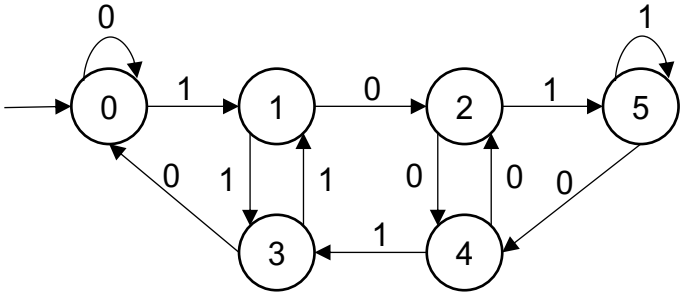
$$L_3 = L_1 \times L_2 = \{A, B\} \times \{C, D, E\} = \{AC, AD, AE, BC, BD, BE\}$$

- El estado inicial es AC dado que anteriormente los estados iniciales eran A de L_1 y C de L_2.
- Empezamos con el estado AC y vemos que tanto para el estado A como para el C cuándo viene un 0, nos quedamos en el mismo estado, es decir que si estamos en AC y viene un 0 nos quedamos en AC, en cambio cuando estamos en el estado A y viene un 1 cambiamos al estado B y cuando estamos en el estado C cambiamos al D por lo que si estamos en AC y viene un 1 cambiaremos al estado BD.
- En el estado AD si en A viene un 0 sigo en A, si estoy en D voy a E, por lo que si estoy en AD y viene un 0 voy a AE, si estoy en A y viene un 1 voy a B y si estoy en D y viene un 1 voy a C, por lo que si estoy en AD y viene un 1 voy a BC.

- $AE \rightarrow \begin{matrix} 0 \rightarrow \begin{matrix} A \rightarrow A \\ E \rightarrow D \end{matrix} \rightarrow AD \\ 1 \rightarrow \begin{matrix} A \rightarrow B \\ E \rightarrow E \end{matrix} \rightarrow BE \end{matrix}$
- $BC \rightarrow \begin{matrix} 0 \rightarrow \begin{matrix} B \rightarrow A \\ C \rightarrow C \end{matrix} \rightarrow AC \\ 1 \rightarrow \begin{matrix} B \rightarrow B \\ C \rightarrow D \end{matrix} \rightarrow BD \end{matrix}$
- $BD \rightarrow \begin{matrix} 0 \rightarrow \begin{matrix} B \rightarrow A \\ D \rightarrow E \end{matrix} \rightarrow AE \\ 1 \rightarrow \begin{matrix} B \rightarrow B \\ D \rightarrow C \end{matrix} \rightarrow BC \end{matrix}$
- $BE \rightarrow \begin{matrix} 0 \rightarrow \begin{matrix} B \rightarrow A \\ E \rightarrow D \end{matrix} \rightarrow AD \\ 1 \rightarrow \begin{matrix} B \rightarrow B \\ E \rightarrow E \end{matrix} \rightarrow BE \end{matrix}$

El resultado de la intersección sería:

		0	1
0	AC	AC	BD
1	BD	AE	BC
2	AE	AD	BE
3	BC	AC	BD
4	AD	AE	BC
5	BE	AD	BE



2.3.3. Intersección

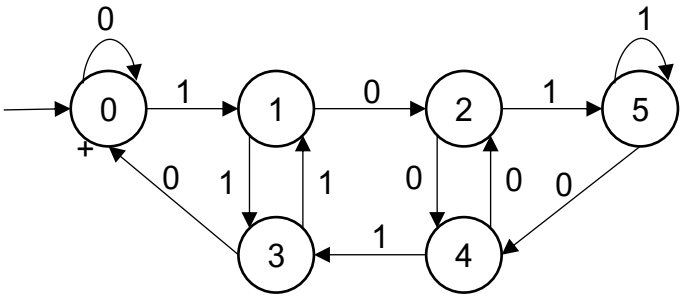
La intersección de dos lenguajes L_1 y L_2, $L_1 \cap L_2$, lo que podríamos definir como un nuevo lenguaje L_3 que contiene las palabras que pertenece a L_1 y a L_2.

Receta:

1. Hacemos el producto cartesiano.
2. Los estados aceptadores son aquellos que contienen únicamente los que anteriormente eran estados aceptadores.

El ejemplo anterior quedaría:

	0	*1*	
0	AC	BD	+
1	AE	BC	
2	AD	BE	
3	AC	BD	
4	AE	BC	
5	AD	BE	



2.3.4. Unión

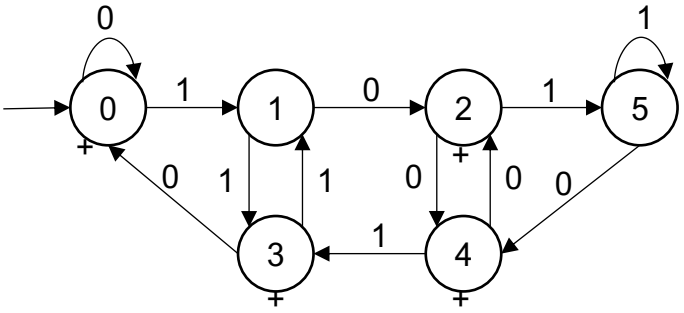
La unión de dos lenguajes L_1 y L_2, $L_1 \cup L_2$, lo que podríamos definir como un nuevo lenguaje L_3 que contiene las palabras que o bien pertenecen a L_1 o bien a L_2.

Receta:

1. Hacemos el producto cartesiano de ambos autómatas.
2. Los estados aceptadores son aquellos que contienen al menos 1 de los que anteriormente eran estados aceptadores.

El ejemplo anterior quedaría:

	0	*1*	
0	AC	BD	+
1	AE	BC	
2	AD	BE	+
3	AC	BD	+
4	AE	BC	+
5	AD	BE	



2.3.5. Concatenación

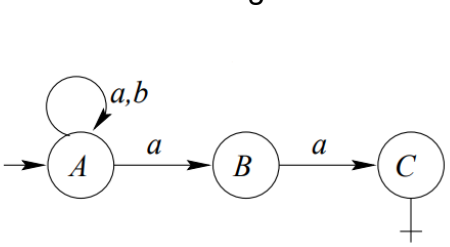
La concatenación de dos lenguajes L_1 y L_2, entonces con la concatenación un nuevo lenguaje $L_3 = L_1 \cdot L_2$.

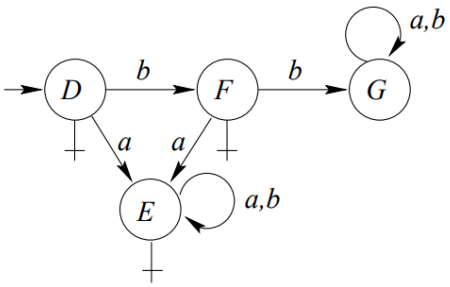
Receta:

1. Se dibuja el diagrama original de L_1
2. Desde cada estado de aceptación del diagrama de L_1 se dibuja un arco hacia cada estado de L_2 que sea el destino de un arco del estado inicial de L_2 y se rotula con el mismo símbolo
3. Dejar que los estados de aceptación de L_1 lo sean si el estado inicial de L_2también lo es.

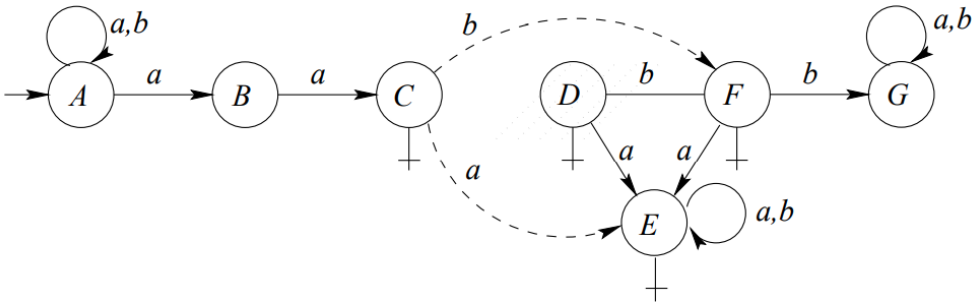
Ejemplo:

- Queremos unir los siguientes dos diagramas:

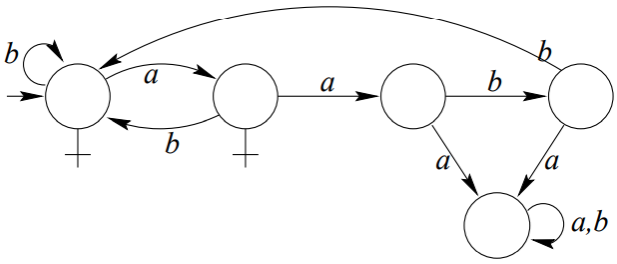




- Aplicamos el paso 1 y 2



- Aplicamos el paso 3



Pasos que seguir para construir un autómata:

1. Si tenemos un universal (∀) pasar a existencial (∃), por lo que habrá que hacer el complementario.
2. Construir un diagrama por cada condición que tengamos y/o concatenación.
3. Ir haciendo las intersecciones, uniones y concatenaciones necesarias.
4. Minimizar
5. Hacer el complementario del autómata, si es necesario.

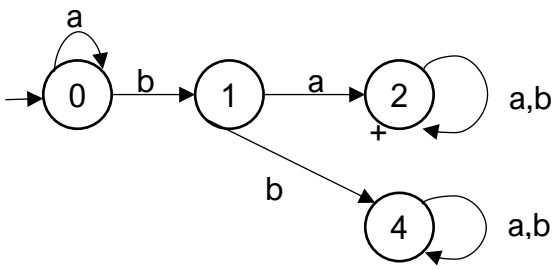
2.3.6. Rebasamiento

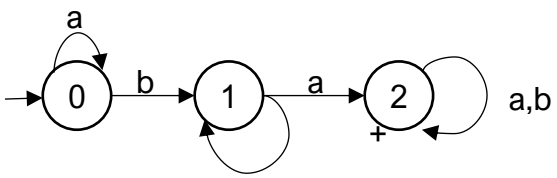
Damos la vuelta al autómata que reconoce L al que reconoce L^R. Esto quiere decir que cambiamos el sentido de las transiciones y intercambiamos los estados iniciales y aceptadores.

3. Minimización

Para minimizar hay 3 formas que a veces hay que combinar:

- Mirar estados que lleven a bucles que no aportan nada





- Por equivalencia, es decir, si las entradas de dos estados llevan al mismo sitio, entonces son equivalentes.
- Mediante el algoritmo de minimización, donde primeramente creamos dos conjuntos {estados no finales} {estados finales}, paso por paso vamos partiendo estos conjuntos, de manera que si un estado por una entrada se queda en el conjunto, es decir va a uno de los estados dentro del conjunto y por otra entrada va a otro conjunto separamos dicho estado en un conjunto nuevo {estados no finales - a} {a} {estados finales} y así sucesivamente hasta que no queden más particiones que hacer de modo que todos los estados que estén en un mismo conjunto son realmente el mismo.