

Grado en Ingeniería Informática  
2019-2020

*Apuntes*

# Teoría de Automatas y Lenguajes Formales

---

Jorge Rodríguez Fraile<sup>1</sup>



Esta obra se encuentra sujeta a la licencia Creative Commons  
**Reconocimiento - No Comercial - Sin Obra Derivada**

---

<sup>1</sup>Universidad: [100405951@alumnos.uc3m.es](mailto:100405951@alumnos.uc3m.es) | Personal: [jrf1616@gmail.com](mailto:jrf1616@gmail.com)



## ÍNDICE GENERAL

<b>I</b>	<b>TEMA 1. Introducción</b>	<b>3</b>
<b>II</b>	<b>TEMA 2. Autómatas Formales</b>	<b>15</b>
<b>III</b>	<b>TEMA 3. Autómatas Finitos</b>	<b>31</b>
<b>IV</b>	<b>TEMA 4. Gramáticas y Lenguajes Formales</b>	<b>71</b>
<b>V</b>	<b>TEMA 5. Expresiones Regulares</b>	<b>139</b>
<b>VI</b>	<b>TEMA 6. Autómatas a Pila</b>	<b>187</b>
<b>VII</b>	<b>TEMA 7. Maquina de Turing</b>	<b>211</b>
<b>VIII</b>	<b>TEMA 8. Complejidad Computacional</b>	<b>239</b>
1.	RECURSOS . . . . .	249
1.1.	Tema 3 . . . . .	249
1.1.1.	$AFD \rightarrow AFD \text{ mínimo}$ . . . . .	249
1.1.2.	$AFND \rightarrow AFD$ . . . . .	249
1.2.	Tema 4 . . . . .	249
1.2.1.	$G3 LD \rightarrow G3 LI$ . . . . .	249
1.2.2.	Lenguaje vacío (G2) . . . . .	250
1.2.3.	Lenguaje infinito (G2) . . . . .	250
1.2.4.	Limpieza y bien-formación de gramáticas . . . . .	250
1.2.5.	$G2 \rightarrow FNC$ . . . . .	250
1.2.6.	$G2 \rightarrow FNG$ . . . . .	251

1.2.7. Quitar recursividad a izquierdas . . . . .	251
1.2.8. Paso de G3LD (FNG) $\rightarrow$ AF y viceversa . . . . .	251
1.3. Tema 5 . . . . .	252
1.3.1. Teoria de síntesis . . . . .	252
1.4. Tema 6 . . . . .	252
1.4.1. $APF \rightarrow APV$ . . . . .	252
1.4.2. $APV \rightarrow APF$ . . . . .	252
1.4.3. $G2 (FNG) \rightarrow APV$ . . . . .	253
1.4.4. $APV \rightarrow G2$ . . . . .	253
1.4.5. Equivalencia de EERR. . . . .	254
1.4.6. Analisis . . . . .	254
1.4.7. Formatos . . . . .	255
1.4.8. Jeraquia de Chomsky . . . . .	255



# **Parte I**

## **TEMA 1. Introducción**



# 1. Introducción a la Teoría de Autómatas y Lenguajes Formales

Grado Ingeniería Informática  
Teoría de Autómatas y Lenguajes Formales

[ 1 ]



## Objetivos

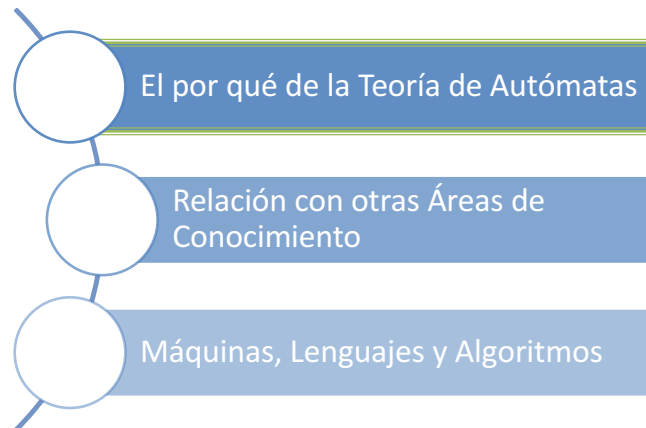
- Presentar la normativa, los contenidos y objetivos de la asignatura poniendo énfasis en las aplicaciones prácticas de la materia que se va a estudiar.
- Conocer la contextualización histórica de la Teoría de Autómatas y lenguajes formales. Desde los orígenes hasta los distintos campos de los que se ha nutrido esta área de conocimiento (Ingeniería, Lenguajes y Gramáticas, y Matemáticas y Computabilidad).
- Conocer el esquema básico que se seguirá a través de la jerarquía de Chomsky sobre los autómatas, gramáticas y lenguajes formales.
- Conocer otras máquinas abstractas relacionadas que se encuentran fuera de la jerarquía de Chomsky.
- Conocer los límites de las máquinas abstractas que se estudiarán y sus problemas de complejidad.

[ 2 ]





# Índice



[ 3 ]



## El por qué de la Teoría de Autómatas

Disciplinas de la Computación según la *"Educational Activities Board of IEEE"*:

- Computer Engineering , **Computer Science**, Information Systems, Information Technologies, Software Engineering

### Computer Science:

- "A pesar de la enorme amplitud de la informática, existen conceptos y habilidades que son comunes a la informática en su conjunto."
- "Todos los estudiantes de informática tienen que aprender a integrar la teoría y la práctica, a reconocer la importancia de la abstracción para apreciar el valor del buen diseño de ingeniería"

[ 4 ]

Fuente: Computing Curricula 2005. The Overview Report.  
[http://www.acm.org/education/curric\\_vols/CC2005-March06Final.pdf](http://www.acm.org/education/curric_vols/CC2005-March06Final.pdf)



## El por qué de la Teoría de Autómatas

- **Ciencias de la Computación:** cuerpo de conocimiento que se ocupa del estudio de los fundamentos teóricos de la información y la computación y de su implementación y aplicación en sistemas computacionales.
- Gibbs y Tucker (1986):
  - “No se debe entender que el objetivo de las Ciencias de la Computación sea la construcción de programas sino el estudio sistemático de los algoritmos y estructuras de datos, específicamente de sus propiedades formales”
  - Gibbs, N. E. and Tucker, A. B. 1986. A model curriculum for a liberal arts degree in computer science. Commun. ACM 29, 3 (Mar. 1986), 202-210. DOI= <http://doi.acm.org/10.1145/5666.5667>

[ 5 ]



## El por qué de la Teoría de Autómatas

### Primera inmersión en la “Teoría de la Computación”:

- Es anterior al invento del Computador (incluso del transistor)
- Propiedades **MATEMÁTICAS FUNDAMENTALES** de Software, Hardware y aplicaciones de los mismos.
- Responder a preguntas como:
  - ¿Cómo puede construirse un programa para resolver un problema?
  - ¿Resuelve el programa realmente el problema?
  - ¿Cuánto se tarda en realizar un cómputo (complejidad temporal)?.
  - ¿Cuanta memoria se necesita para realizar el cómputo (complejidad espacial)?.
  - Y el “modelo de computación” (Imperativo, POO, Programación Lógica, etc.)
  - ¿Qué se puede computar y qué NO se puede computar?.

[ 6 ]



## El por qué de la Teoría de Autómatas.

Aplicación directa de conceptos propios de las Ciencias de la Computación:

- Videojuegos
  - Comportamiento de personajes
- Compiladores y Procesamiento de Lenguaje Natural
  - Análisis Léxico en lenguajes programación (compilador)
  - Búsqueda de cadenas o comparación de “patrones”
  - Diseño de nuevos lenguajes de programación o ampliación
- Implementación de Protocolos Robustos
  - Para clientes o usuarios
  - E.g. Sistemas de Seguridad
- Criptografía Moderna (sus protocolos)
- ...



[ 7 ]

## El por qué de la Teoría de Autómatas.

Aplicación directa de conceptos propios de las Ciencias de la Computación:

- ...
- Construcción de sistemas computacionales más elegantes y sencillos.
- Diseño (Maquina Secuencial --> Código)
- Diseño de estructuras y “parsing”: gramaticas (ej: XML)
  - Búsqueda de cadenas o comparación de “patrones”
- SW para diseñar y evaluar circuitos digitales.
- “Escanear” grandes cantidades de texto (web)
- SW para verificar sistemas que tiene un número finito de “estados”



[ 8 ]

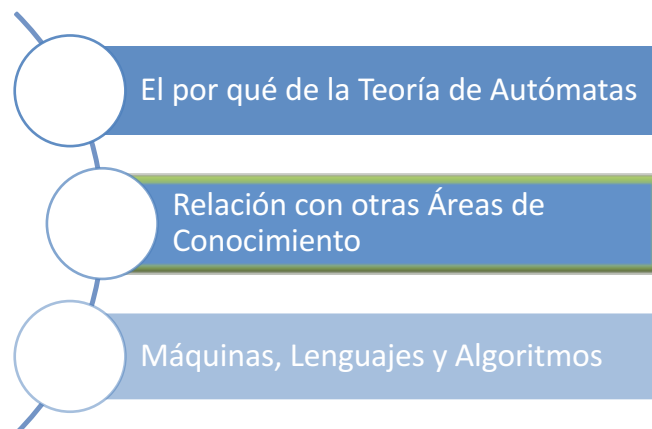
# El por qué de la Teoría de Autómatas

- Teoría de la Computación:
  - ¿Aburrida y arcaica? NO, es Comprensible e Interesante.
- Proporciona al Ingeniero:
  - Aspectos teóricos (permite innovación)
    - Autómatas,
    - Representación Estructural (Gramáticas)
    - Autómatas y Máquinas para establecer los límites de la Computabilidad.
  - Aspectos prácticos (ingeniería)

[ 9 ]



# Índice



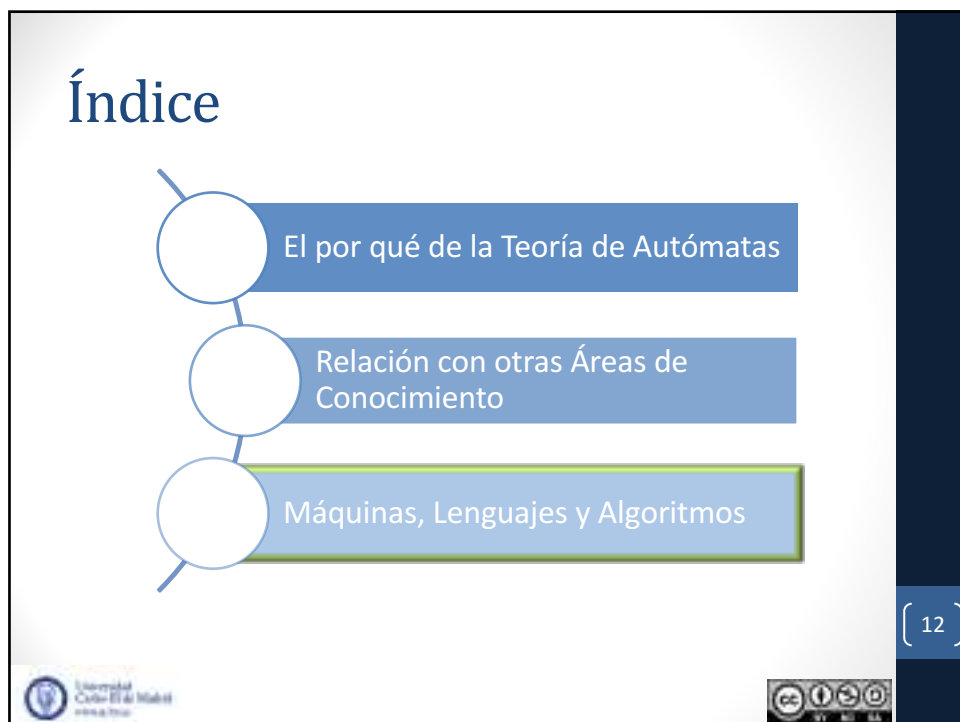
[ 10 ]



## Relación con otras áreas.



## Índice



# Máquinas, Lenguajes y Algoritmos

Tres pilares sustentan la  
Teoría de Lenguajes,  
Gramáticas y Autómatas

## AUTÓMATAS (ingeniería)

- Leonardo Torres, 1915
- Shannon, 1938
- Mc Culloch-Pitts, 1943
- Moore, 1956

## COMPUTABILIDAD (matemáticas)

- Hilbert, 1928
- Gödel, Kleene, Post y Turing, ≈1930
- Church, 1936
- Rabin, 1960
- Cobhan, 1964
- Cook, 1972
- Aho, Hopcroft, Ullman, 1974

## LENGUAJES y GRAMÁTICAS (lingüística)

- Panini, entre el 400 y 200 AC
- Chomsky, 1967
- Backus, ≈1960
- Kleene, 1951
- Hirst, Tennant y Carbonell, 1981

[ 13 ]



# Máquinas, Lenguajes y Algoritmos



Máquinas o  
Autómatas

- Aplicación en campos muy diversos
- Manejan conceptos como “control”, “acción”, “memoria”
- Los objetos son controlados o recordados con símbolos, palabras o frases de algún tipo.
- Máquina de Moore y máquina de Mealy
- Circuitos combinatorios
- Autómatas Probabilísticos (incertidumbre en las transiciones)
- McCulloch-Pitts (1943) describieron los cálculos lógicos inmersos en un dispositivo denominado neurona artificial.
  - Redes de Neuronas Artificiales
- Autómatas Celulares (J.H. Conway, el juego de la vida).

[ 14 ]





Máquina de Turing Universal, Jim Wiked.

<http://blogs.20minutos.es/mati-una-profesora-muy-particular/2012/09/10/1957/>



[ 15 ]

## Máquinas, Lenguajes y Algoritmos

### Lenguajes y Gramáticas

- Origen en la lingüística
- Noam Chomsky
  - Jerarquía de Chomsky (1956)
- “Backus normal form”  
(para gramática de ALGOL)
  - Lenguajes de Programación
  - Lenguajes Naturales
  - Sistemas de Comandos



Noam Chomsky  
(1928 - )

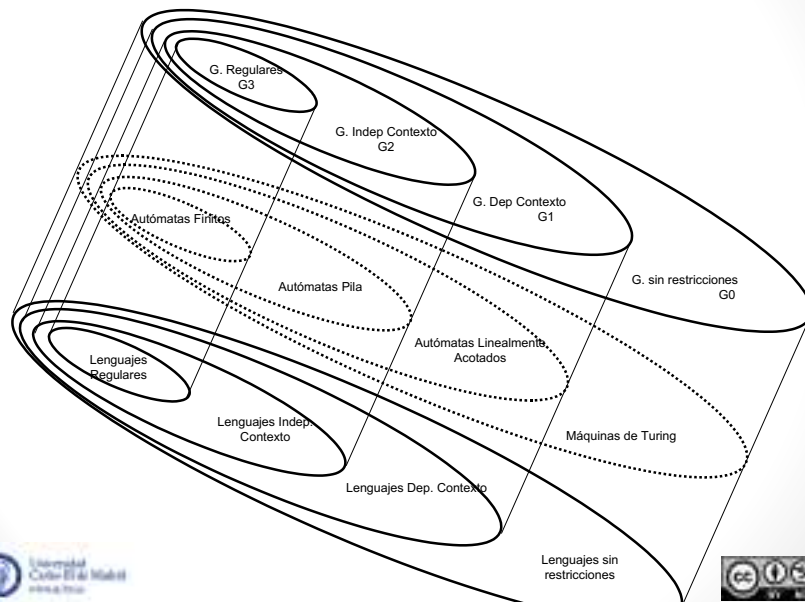


John Backus  
(1924 - 2007)



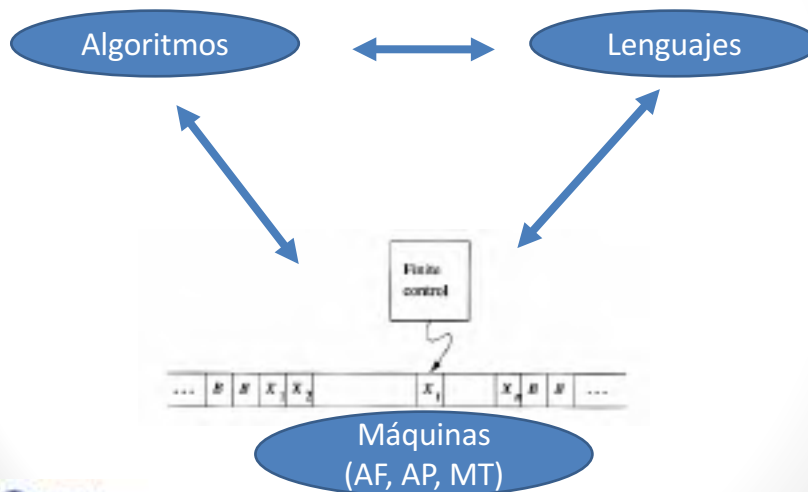
[ 16 ]

# Máquinas, Lenguajes y Algoritmos



[ 17 ]

# Máquinas, Lenguajes y Algoritmos



[ 18 ]



## Bibliografía

- **Referencias básicas :**

1. J. E. Hopcroft, R. Motwani, J. D. Ullman. *Introducción a la Teoría de Autómatas, Lenguajes y Computación*. Ed. Pearson Addison Wesley , 2008  
Capítulo 1. Introducción a lo Autómatas
2. E. Alfonseca Cubero, M. Alfonseca Moreno, R. Moriyón Salomón. *Teoría de Autómatas y Lenguajes Formales*. Ed. McGraw-Hill, 2007  
Capítulo 1. Máquinas, Lenguajes y Problemas.

- **Referencias complementarias:**

1. P. Isasi, P. Martínez, D. Borrajo. *Lenguajes, Gramáticas y Autómatas: Un enfoque práctico*. Ed. Addison-Wesley, 1997  
Capítulo 2. Lenguajes y Gramáticas Formales
2. D. M Kelley. *Teoría de autómatas y lenguajes formales*. Prentice-Hall, 1995  
Capítulo 2. Lenguajes Regulares.
3. R. Penrose. *La Nueva Mente del Emperador*. DeBolsillo, 2011  
Capítulo 1. ¿Puede tener mente un computador?  
Capítulo 2. Algoritmos y máquinas de Turing
4. R. Penrose. *Las sombras de la mente: hacia una comprensión científica de la consciencia*. Mondadori. 1996
5. D.R. Hofstadter. *Gödel, Escher, Bach : un eterno y grácil bucle*. Tusquets, 1998

[ 19 ]



# 1. Introducción a la Teoría de Autómatas y Lenguajes Formales

Grado Ingeniería Informática  
Teoría de Autómatas y Lenguajes Formales

[ 20 ]



## **Parte II**

### **TEMA 2. Autómatas Formales**



# 2. Teoría de Autómatas

Grado Ingeniería Informática  
Teoría de Autómatas y Lenguajes Formales

uc3m

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso




- Introducción
- Tipos de autómatas
- Aplicaciones
- Lenguajes Formales

uc3m

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

( 2 )



# Introducción y definiciones

- Se trata de saber qué (y qué no) se puede computar.

Y además...

**cómo de rápido,  
con cuánta memoria y  
con qué modelo de computación.**

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

{ 3 }

uc3m



# Introducción y definiciones

- Qué se entiende por **computación**?
- La Teoría de Autómatas se centra en la computación en sí, no en detalles sobre dispositivos de entrada y salida.  
(Así, no se trata de crear modelos matemáticos para un video juego, por ejemplo).

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

{ 4 }

uc3m



# Autómata

Definición RAE

## **autómata.**

(Del lat. automāta, t. f. de -tus, y este del gr. αὐτόματος, espontáneo).

1. m. Instrumento o aparato que encierra dentro de sí el mecanismo que le imprime determinados movimientos.
2. m. Máquina que imita la figura y los movimientos de un ser animado.
3. m. coloq. Persona estúpida o excesivamente débil, que se deja dirigir por otra.

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

[ 5 ]

uc3m



# Modelo Matemático

## **Autómata:**

Modelo Matemático de computación.

Dispositivo abstracto con capacidad de computación.

## **Teoría de Autómatas:**

Abstracción de cualquier tipo de computador y/o lenguaje de programación.

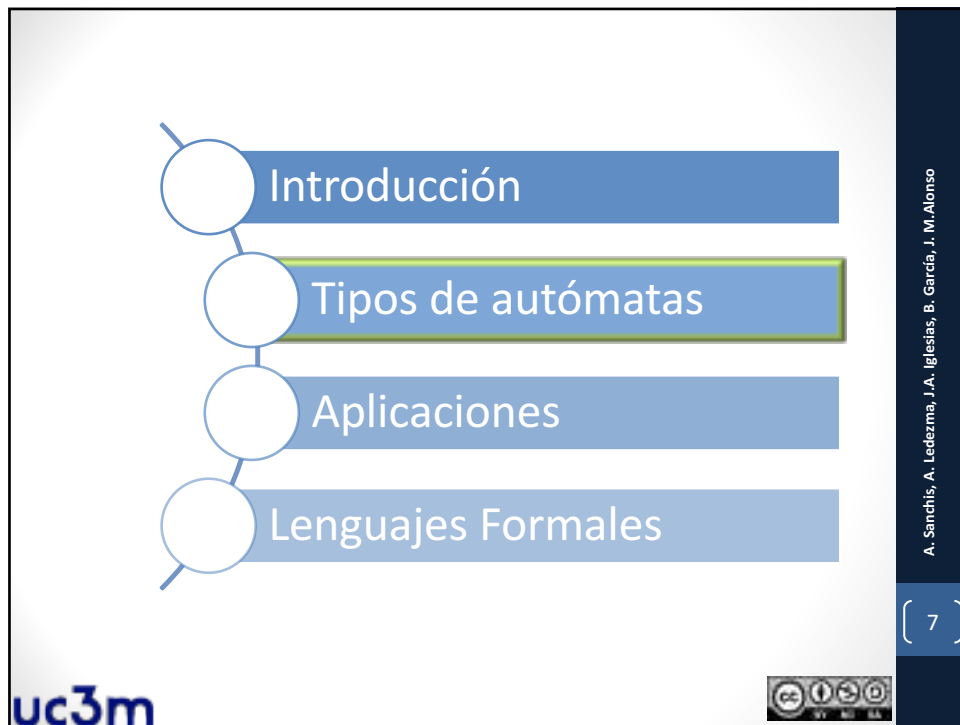
Desglose en sus elementos básicos (Entrada, Estado, Transición, Salidas y elementos auxiliares)

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

[ 6 ]

uc3m





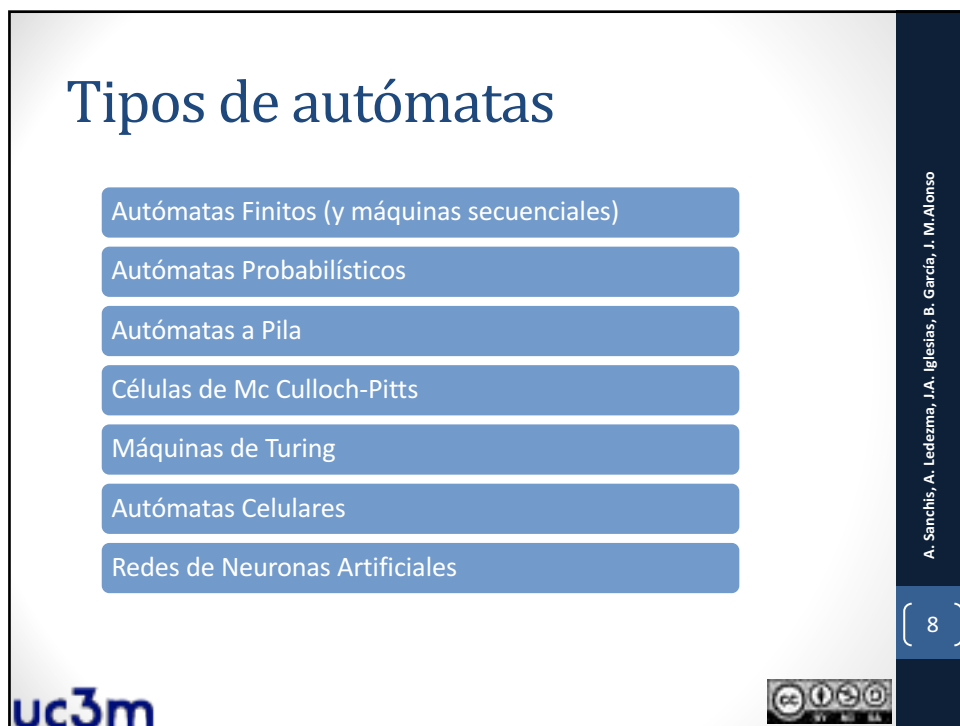
A vertical table of contents with four items, each preceded by a white circle and a horizontal blue bar. The items are: 'Introducción', 'Tipos de autómatas' (highlighted with a green border), 'Aplicaciones', and 'Lenguajes Formales'. The slide includes a 'uc3m' logo at the bottom left, a Creative Commons license icon at the bottom right, and a vertical sidebar on the right with the authors' names and a page number '7' in brackets.

- Introducción
- Tipos de autómatas
- Aplicaciones
- Lenguajes Formales

uc3m

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

[ 7 ]



A slide titled 'Tipos de autómatas' featuring a list of seven automata types, each in a blue rounded rectangle. The types are: 'Autómatas Finitos (y máquinas secuenciales)', 'Autómatas Probabilísticos', 'Autómatas a Pila', 'Células de Mc Culloch-Pitts', 'Máquinas de Turing', 'Autómatas Celulares', and 'Redes de Neuronas Artificiales'. The slide includes a 'uc3m' logo at the bottom left, a Creative Commons license icon at the bottom right, and a vertical sidebar on the right with the authors' names and a page number '8' in brackets.

## Tipos de autómatas

- Autómatas Finitos (y máquinas secuenciales)
- Autómatas Probabilísticos
- Autómatas a Pila
- Células de Mc Culloch-Pitts
- Máquinas de Turing
- Autómatas Celulares
- Redes de Neuronas Artificiales

uc3m

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

[ 8 ]

## Tipos de autómatas

Autómatas Finitos

Autómatas Probabilísticos

Autómatas a Pila

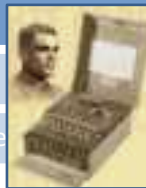
Células de Mc Culloch-Pitts

Máquinas de Turing

Autómatas Celulares

Redes de Neuronas Artificiales

Turing estudió una máquina abstracta con la misma capacidad que los computadores actuales desde el punto de vista de lo que son capaces de hacer.



A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso



uc3m



## Tipos de autómatas

Autómatas Finitos (y máquinas secuenciales)

Autómatas Probabilísticos

Autómatas a Pila

Células de Mc Culloch-Pitts

Máquinas de Turing

Autómatas Celulares

Redes de Neuronas Artificiales

Mayor  
capacidad de  
cómputo.

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso



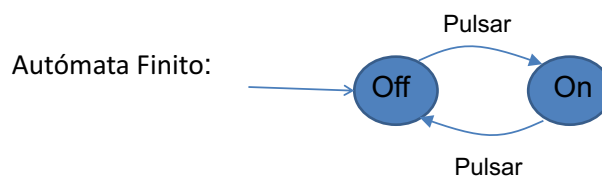
uc3m





# Autómatas y Algoritmos

- La máquina de Turing es un modelo matemático abstracto que formaliza el concepto de algoritmo
- Todo Autómata puede ser transformado en un algoritmo y a la inversa.



A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

[ 11 ]

uc3m



# Autómatas Discretos, continuos e híbridos

Criterio: Entradas

- Suelen ser DISCRETOS:

Autómatas Finitos (y máquinas secuenciales)

Autómatas a Pila

Máquinas de Turing

- Son DISCRETOS, CONTÍNUOS Y/O HÍBRIDOS:

Autómatas Celulares

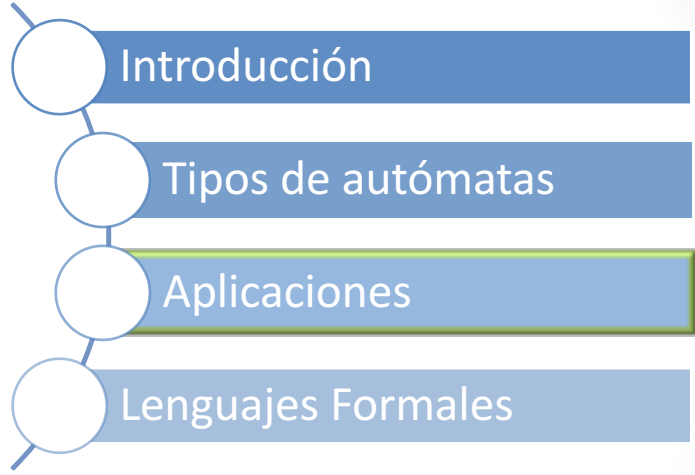
Redes de Neuronas Artificiales

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

[ 12 ]

uc3m





Introducción

Tipos de autómatas

Aplicaciones

Lenguajes Formales

uc3m

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

( 13 )

## Aplicaciones de los autómatas

### El Juego de la Vida

- Ejemplo de un juego implementado usando un **autómata celular**. Diseñado por el matemático británico John Conway en 1970.
- El todo es más que la suma de las partes.
- Las transiciones dependen del número de células vecinas vivas:
  - Una célula muerta con exactamente 3 células vecinas vivas "nace" (al turno siguiente estará viva).
  - Una célula viva con 2 ó 3 células vecinas vivas sigue viva, en otro caso muere o permanece muerta (por "soledad" o "superpoblación").

<http://www.youtube.com/watch?v=XcuBvj0pw-E>

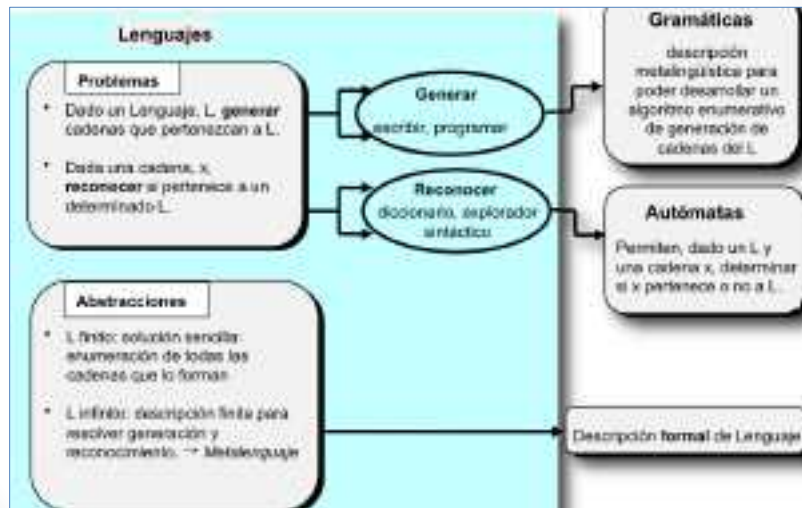
uc3m

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

( 14 )



# Lenguajes Formales



A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

uc3m



## Lenguajes Formales. Definiciones

**Símbolo:** Entidad abstracta, realmente no se define (análogo al punto en geometría). Son letras, dígitos, caracteres, etc. Forman parte de un alfabeto. También posible encontrar símbolos formados por varios caracteres, pej: IF, THEN, ELSE, ...

**Alfabeto ( $\Sigma$ ):** Conjunto finito no vacío de letras o símbolos.

Sea " $a$ " una letra y  $\Sigma$  un alfabeto, si  $a$  pertenece a ese alfabeto  $\Rightarrow$

$$a \in \Sigma$$

Ejemplos:

- $\Sigma_1 = \{A, B, C, \dots, Z\}$  alfabeto de las letras mayúsculas
- $\Sigma_2 = \{0, 1\}$  alfabeto binario
- $\Sigma_3 = \{IF, THEN, ELSE, BEGIN, END\}$  alfabeto de símbolos para programación.

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

( 18 )

uc3m



## Lenguajes Formales. Definiciones

**Palabra:** toda secuencia finita de símbolos del alfabeto.

$\Sigma_1 = \{A, B, C, \dots, Z\}$ ; palabras sobre  $\Sigma_1$  JUAN, ISABEL, etc.

$\Sigma_2 = \{0, 1\}$ ; palabras sobre  $\Sigma_2$  00011101

$\Sigma_3 = \{IF, THEN, ELSE, BEGIN, END\}$ ;

palabras sobre  $\Sigma_3$  IFTHENELSEEND

Notación: se representan por letras minúsculas del final del alfabeto ( $x, y, z$ )

Ejem:  $x = \text{JUAN}$ ;  $y = \text{IFTHENELSEEND}$ ;  $z = 00001111111111$

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

( 19 )

uc3m



## Lenguajes Formales. Definiciones

**Longitud de palabra:** número de símbolos que componen una palabra.

Se representa por  $|x|$

Ejemplos:

$\Sigma_1 = \{A, B, C, \dots, Z\}$ ;  $|x| = |\text{JUAN}| = 4$

$|y| = |\text{IFTHENELSEEND}| = 13$

$\Sigma_3 = \{IF, THEN, ELSE, BEGIN, END\}$ ;

$|y| = |\text{IFTHENELSEEND}| = 4$  **OJO!!!!**

**Palabra vacía  $\lambda$ :** Es aquella palabra cuya longitud es cero

Se representa por  $\lambda$ ,  $|\lambda| = 0$

Sobre cualquier alfabeto es posible construir  $\lambda$

Utilidad: será el elemento neutro en muchas operaciones (concatenación, etc.) con palabras y lenguajes

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

( 20 )

uc3m



## Lenguajes Formales. Definiciones

**Universo del discurso,  $W(\Sigma)$ :** conjunto de todas las palabras que se pueden formar con los símbolos de un alfabeto  $\Sigma$

También se denomina **Lenguaje Universal del alfabeto  $\Sigma$**

Se representa como  $W(\Sigma)$

Es un conjunto infinito (i.e. número infinito de palabras)

Ejemplo: sea  $\Sigma_4 = \{A,B\}$ ,  $W(\Sigma_4) = \{\lambda, A, B, AA, AB, BA, BB, AAA, \dots\}$  con un número  $\infty$  de palabras

### COROLARIO:

$\forall \Sigma, \lambda \in W(\Sigma) \Rightarrow$  La palabra vacía pertenece a todos los lenguajes universales de todos los alfabetos posibles

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

( 21 )

uc3m



## Lenguajes Formales. Operaciones

**Algunas operaciones importantes con palabras**, sobre palabras de un universo del discurso dado:

### Concatenación de palabras:

sean dos palabras  $x, y$  tal que  $x \in W(\Sigma)$ ,  $y \in W(\Sigma)$ , y sea

$|x| = i = |x_1x_2\dots x_i|$  e  $|y| = j = |y_1y_2\dots y_j|$ ,

se llama concatenación de  $x$  con  $y$ , a:

$x \cdot y = x_1x_2\dots x_i y_1y_2\dots y_j = z$ , donde  $z \in W(\Sigma)$

#### Propiedades de la concatenación:

- Operación cerrada
- Propiedad Asociativa
- Con elemento neutro
- No conmutativa

#### Definiciones:

- cabeza
- cola
- longitud de palabra

( 22 )

uc3m



A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

## Lenguajes Formales. Operaciones

**Potencia de una palabra:** Reducción de la concatenación a los casos que se refieren a una misma palabra

- potencia *i-ésima* de una palabra es el resultado de concatenar esa palabra consigo misma *i* veces
- La concatenación es asociativa  $\Rightarrow$  no especificar el orden
- $x^i = x \cdot x \cdot x \cdot \dots \cdot x$  ("x" *i* veces)
- $|x^i| = i \cdot |x|$  ( $i > 0$ )
- se cumple:
  - $x^1 = x$
  - $x^{1+i} = x \cdot x^i = x^i \cdot x$  ( $i > 0$ )
  - $x^{j+i} = x^j \cdot x^i = x^i \cdot x^j$  ( $i, j > 0$ )
- Si se define  $x^0 = \lambda$

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

[ 23 ]

uc3m



## Lenguajes Formales. Definiciones

**Lenguaje (L):** Se denomina lenguaje sobre el alfabeto  $\Sigma$ :

- a todo subconjunto del lenguaje universal de  $\Sigma$ ,  $L \subset W(\Sigma)$
- a todo conjunto de palabras sobre un determinado  $\Sigma$   
(generado a partir del alfabeto  $\Sigma$ )

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

[ 24 ]

uc3m



# Lenguajes Formales

## Lenguajes Especiales:

1.  $\phi$  = Lenguaje vacío,  $\phi \subset W(\Sigma)$
2.  $\{\lambda\}$  = Lenguaje de la palabra vacía
  - se diferencian en el número de palabras (cardinalidad) que los forman  $C(\phi) = 0$  mientras que  $C(\{\lambda\})=1$
  - se parecen en que  $\phi$  y  $\{\lambda\}$  son lenguajes sobre cualquier alfabeto
3. Un alfabeto es uno de los lenguajes generados por el mismo:  
 $\Sigma \subset W(\Sigma)$ , por ejemplo el chino

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

[ 25 ]

uc3m



# Lenguajes Formales

## Concatenación de Lenguajes : Sobre un alfabeto dado $\Sigma$

Sean  $L_1$  y  $L_2$  definidos sobre el mismo alfabeto  $\Sigma$ ,  $L_1, L_2 \subset W(\Sigma)$ ;  
 se llama **concatenación** de dos lenguajes,  $L_1, L_2$  y se representa por  $L_1 \cdot L_2$  al lenguaje así definido:

$$L_1 \cdot L_2 = \{ x.y / x \in L_1 \text{ AND } y \in L_2 \}$$

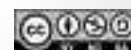
Es el conjunto formado por palabras formadas por parte del primero seguidas de parte del segundo.

Operación cerrada Asociativa Elemento Neutro  $\{\lambda\}$  NO comutativa

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

[ 26 ]

uc3m





# Lenguajes Formales

**Unión de Lenguajes :** Sobre un alfabeto dado  $\Sigma$

Sean  $L_1$  y  $L_2$  definidos sobre el mismo alfabeto  $\Sigma$ ,  $L_1, L_2 \subset W(\Sigma)$ ;  
se llama **unión** de dos lenguajes,  $L_1, L_2$  y se representa por  $L_1 \cup L_2$  al  
lenguaje así definido:

$$L_1 \cup L_2 = \{x / x \in L_1 \text{ ó } x \in L_2\} =$$

Es el conjunto formado indistintamente por palabras de uno u otro  
de los dos lenguajes (equivale a la suma).  $L_1 + L_2 = L_1 \cup L_2$

Op. Cerrada, Asociativa, Conmutativa y Elemento neutro  $\phi$

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

[ 27 ]

uc3m



## 2. Teoría de Autómatas

Araceli Sanchis de Miguel  
Agapito Ledezma Espino  
José A. Iglesias Martínez  
Beatriz García Jiménez  
Juan Manuel Alonso Weber

Grado Ingeniería Informática  
Teoría de Autómatas y Lenguajes Formales

A. Sanchis, A. Ledezma, J.A. Iglesias, B. García, J. M. Alonso

uc3m



## **Parte III**

### **TEMA 3. Autómatas Finitos**



# 3. Autómatas Finitos

Grado Ingeniería Informática  
Teoría de Autómatas y Lenguajes Formales

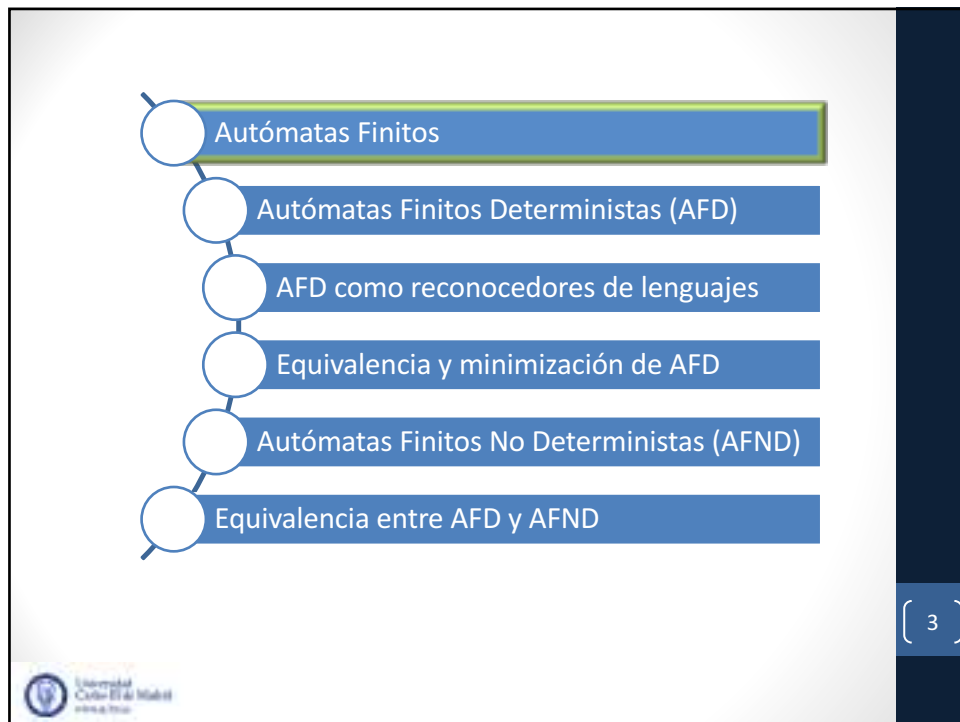


## Objetivos

- Definir el concepto de Autómata Finito Determinista (AFD).
- Definir el concepto de Autómata Finito No Determinista (AFND).
- Establecer las equivalencias entre AFD.
- Convertir un AFND en un AFD.
- Minimizar AFD.
- Identificar el tipo de lenguaje aceptado por un AFND.

[ 2 ]

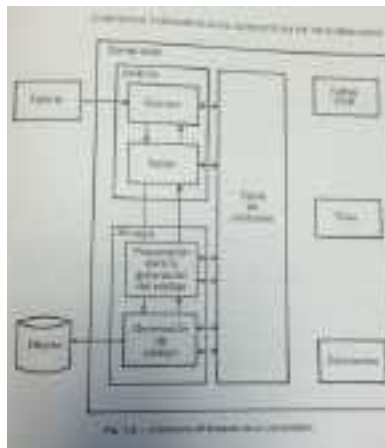




[ 3 ]

## Aplicaciones de AFs

- **Compilador:** Proceso de traducción que convierte un programa fuente escrito en un lenguaje de alto nivel a un programa objeto en código máquina y listo para ejecutarse en un ordenador –con poca o ninguna preparación adicional.
- **Fases:** análisis y síntesis



[ 4 ]

# Autómatas Finitos

- Los Autómatas Finitos son de dos tipos:

- Deterministas:**

- cada combinación (estado, símbolo de entrada) produce un solo (estado).

- No Deterministas:**

- cada combinación (estado, símbolo de entrada) produce varios (estado1, estado 2, ..., estado i).
    - son posibles transiciones con  $\lambda$

[ 5 ]



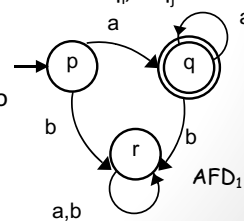
# Autómatas Finitos. Representación

- Se pueden representar mediante:

1. Diagramas de transición o
2. Tablas de transición

- Diagramas de transición:**

- Nodos etiquetados por los **estados** ( $q_i \in$  Conjunto de estados)
- Arcos** entre nodos  $q_i$  a  $q_j$  etiquetados con  $e_i$  ( $e_i$  es un símbolo de entrada) si existe la transición de  $q_i$  a  $q_j$  con  $e_i$
- El **estado inicial** se señala con  $\rightarrow$
- El **estado final** se señala con  $*$  o doble círculo



[ 6 ]



# Autómatas Finitos. Representación

## 2. Tablas de transición:

- Filas encabezadas por los estados ( $q_i \in \text{Conjunto de estados}$ )
- Columnas encabezadas por los símbolos de entrada ( $e_i \in \text{alfabeto de entrada}$ )

	$e_1$	$e_2$	...	$e_n$
$q_1$		$f(q_1, e_2)$		
...				
$*q_m$				

Estados

Símbolos de Entrada

[ 7 ]



[ 8 ]



## Autómatas Finitos Deterministas

- AF Deterministas, **AFD's**: se definen mediante una quintupla  $(\Sigma, Q, f, q_0, F)$ , donde:
  - $\Sigma$ : alfabeto de entrada
  - $Q$ : conjunto de estados, es conjunto finito no vacío, realmente un alfabeto para distinguir a los estados
  - $f: Q \times \Sigma \rightarrow Q$ , función de transición
  - $q_0 \in Q$ , estado inicial
  - $F \subseteq Q$ : conjunto de estados finales o de aceptación



[ 9 ]

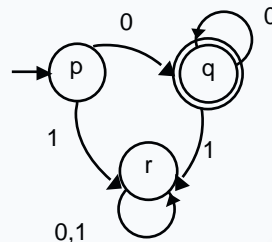
## Autómatas Finitos Deterministas

- Ejemplo: El  $AFD_1 = (\{0,1\}, \{p,q,r\}, f, p, \{q\})$ , donde  $f$  está definida por:

$$\begin{array}{ll} f(p,0) = q & f(p,1) = r \\ f(q,0) = q & f(q,1) = r \\ f(r,0) = r & f(r,1) = r \end{array}$$

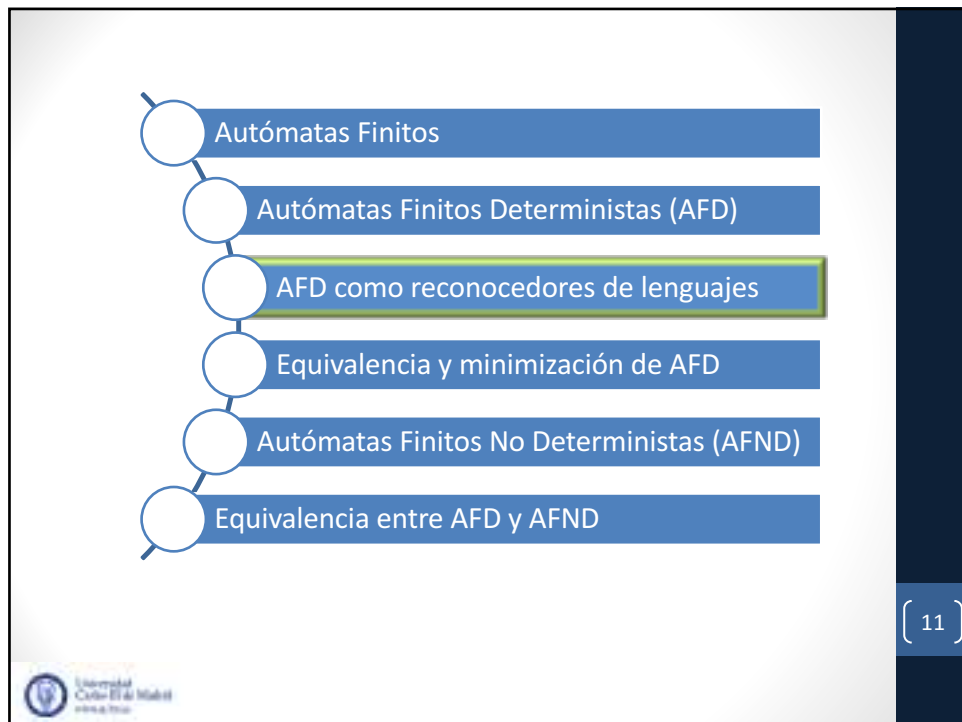
Tiene la tabla de transición y el diagrama de estados siguientes:

	0	1
p	q	r
*q	q	r
r	r	r



[ 10 ]





## AFD como reconocedores de Lenguajes

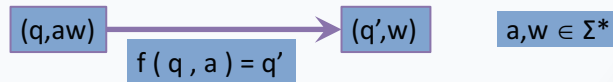
- Cuando un AF transita desde  $q_0$  a un estado final en varios movimientos, se ha producido el RECONOCIMIENTO o ACEPTACIÓN de la cadena de entrada
- Cuando un AF no es capaz de alcanzar un estado final, se dice que el AF NO RECONOCE la cadena de entrada y que ésta NO PERTENECE al lenguaje reconocido por el AF

[ 12 ]

## AFD. Conceptos Básicos

- **Configuración:** es un par ordenado de la forma  $(q,w)$  donde:
  - $q$ : estado actual del AF
  - $w$ : cadena que le queda por leer en ese instante,  $w \in \Sigma^*$  Universo del Discurso
- **Configuración inicial:**  $(q_0, t)$ 
  - $q_0$ : estado inicial
  - $t$ : cadena de entrada a reconocer por el AFD,  $t \in \Sigma^*$
- **Configuración final:**  $(q_f, \lambda)$ 
  - $q_f$ : estado final
  - $\lambda$  la cadena de entrada ha sido leída completamente

- **Movimiento:** es el tránsito entre dos configuraciones.



[ 13 ]



## AFD. Conceptos Básicos

### Extensión a palabra de la función de transición $f, f'$ :

Es la ampliación de la definición de  $f$  a palabras de  $\Sigma^*$ , i.e.  $w \in \Sigma^*$

- $f': Q \times \Sigma^* \rightarrow Q$   
a partir de  $f$ , que sólo considera palabras de longitud 1,  
hay que añadir:
  - $f'(q, \lambda) = q \quad \forall q \in Q$
  - $f'(q, a-x) = f'(f(q, a), x) \quad \forall q \in Q, a \in \Sigma, x \in \Sigma^*$

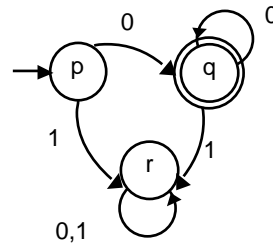
[ 14 ]



## AFD. Conceptos Básicos

- En el AFD<sub>1</sub> (de la figura), indicar el resultado de las siguientes expresiones:

- $f'(p, \lambda)$
- $f'(p, 0^n)$
- $f'(p, 11)$
- $f'(p, 0011010)$
- $f'(p, 100)$



[ 15 ]



## AFD. Conceptos Básicos

### Lenguaje asociado a un AFD:

- Sea un AFD  $= (\Sigma, Q, f, q_0, F)$ , se dice que una palabra  $x$  es aceptada o **reconocida** por el AFD si  $f'(q_0, x) \in F$
- Se llama **lenguaje asociado a un AFD** al conjunto de todas las palabras aceptadas por éste:

$$L = \{ x / x \in \Sigma^* \text{ and } f'(q_0, x) \in F \}$$

- Si  $F = \{ \} = \emptyset \Rightarrow L = \emptyset$
- Si  $F = Q \Rightarrow L = \Sigma^*$
- Otra definición:

$$L = \{ x / x \in \Sigma^* \text{ and } (q_0, x) \rightarrow (q, \lambda) \text{ and } q \in F \}$$

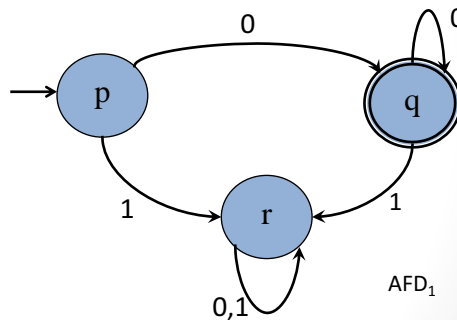
[ 16 ]



## AFD. Conceptos Básicos

En el AFD<sub>1</sub>

- Cuál es  $L(\text{AFD}_1) = \text{¿?}$
- Y si se hace  $F = \{r\}$ ,  
cuál es  $L(\text{AFD}_1) = \text{¿?}$

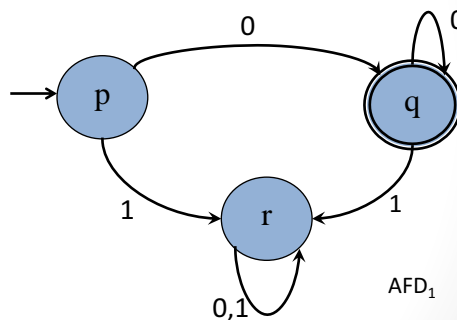


[ 17 ]

## AFD. Conceptos Básicos

En el AFD<sub>1</sub>

- Cuál es  $L(\text{AFD}_1) = \{0^n / n > 0\}$ .

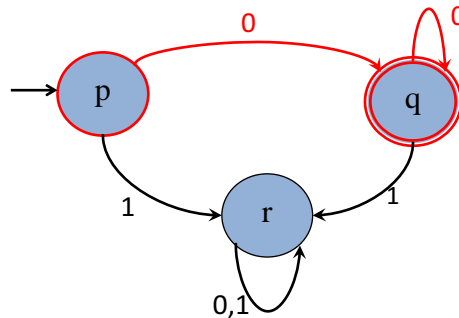


[ 18 ]

## AFD. Conceptos Básicos

En el AFD<sub>1</sub>

- Cuál es  $L(\text{AFD}_1) = \{0^n / n > 0\}$ . **Comprobación**



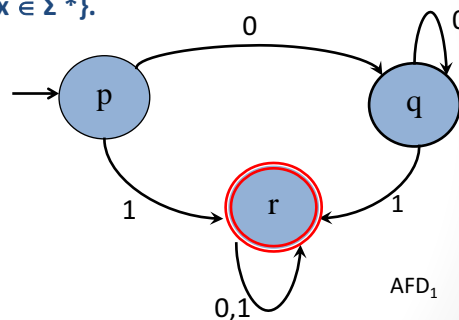
Desde p, con el número de 0's que sea, pero siempre al menos uno, se llega al estado final

[ 19 ]

## AFD. Conceptos Básicos

En el AFD<sub>1</sub>

- Y si se hace  $F = \{r\}$ ,  
 $L(\text{AFD}_1) = \{0^n 1x / n \geq 0, x \in \Sigma^*\}$ .



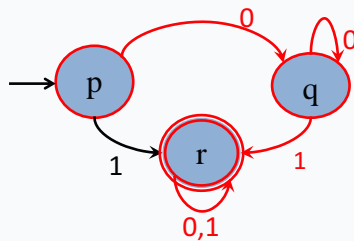
AFD<sub>1</sub>

[ 20 ]

## AFD. Conceptos Básicos

En el AFD<sub>1</sub>,

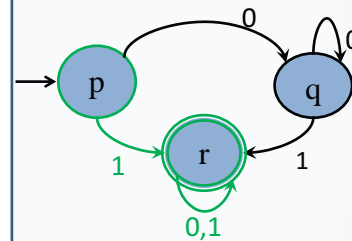
- comprobar que si se hace  $F = \{r\}$ ,  $L(\text{AFD}_1) = \{0^n 1x \mid n \geq 0, x \in \Sigma^*\}$ .



Desde p, con un "0" llego al estado q y desde allí se pueden aceptar tantos 0s como sean.

Luego con un 1 salto al estado final y allí puedo terminar o reconocer cualquier cadena de 0s y 1s.

**Expresión regular:  $L_A = 0^*1(0+1)^*$**



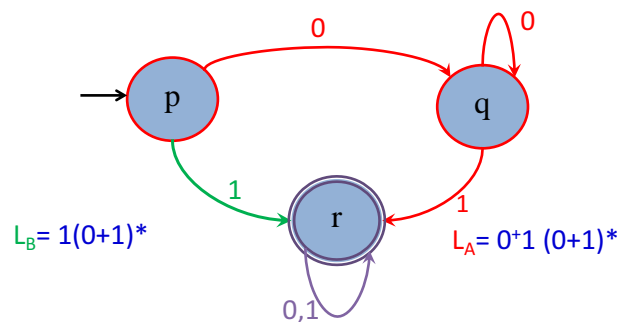
**Expresión regular:  $L_B = 1(0+1)^*$**

[ 21 ]

## AFD. Conceptos Básicos

En el AFD<sub>1</sub>,

- comprobar que si se hace  $F = \{r\}$ ,  $L(\text{AFD}_1) = \{0^n 1x \mid n \geq 0, x \in \Sigma^*\}$ .



**Expresión regular  $L_A \cup L_B = 0^*1(0+1)^*$**

[ 22 ]

## AFD. Conceptos Básicos

### Estados accesibles y Autómatas conexos:

- Sea un AFD  $= (\Sigma, Q, f, q_0, F)$ , el estado  $p \in Q$  es ACCESIBLE desde  $q \in Q$  si  $\exists x \in \Sigma^* f'(q, x) = p$ . En otro caso se dice que INACCESIBLE.  
Todo estado es accesible desde sí mismo pues  $f'(p, \lambda) = p$

#### Teoremas:

- teorema 3.2.2, libro 1 de la bibliografía.  
Sea un AFD,  $|Q| = n, \forall p, q \in Q$   $p$  es accesible desde  $q$   
**sii**  $\exists x \in \Sigma^*, |x| < n / f'(q, x) = p$
- teorema 3.2.3, libro 1 de la bibliografía  
Sea un AFD,  $|Q| = n$ , entonces  $L_{AFD} \neq \emptyset$  **sii** el AFD acepta al menos una palabra  $x \in \Sigma^*, |x| < n$   
Nota: sii= "si y solo si"

[ 23 ]



## AFD. Conceptos Básicos

### Estados accesibles y Autómatas conexos:

Sea un AFD  $= (\Sigma, Q, f, q_0, F)$ . Diremos que el autómata es conexo si todos los estados de  $Q$  son accesibles desde  $q_0$

Dado un autómata no conexo, podemos obtener a partir de él otro autómata equivalente conexo eliminando los estados inaccesibles desde el estado inicial. Los autómatas reconocen el mismo lenguaje.

#### Eliminación de estados inaccesibles.

- ¿Qué algoritmo, para ser implementado en un programa, se podría implementar para marcar los accesibles?

[ 24 ]



## AFD. Ejercicios

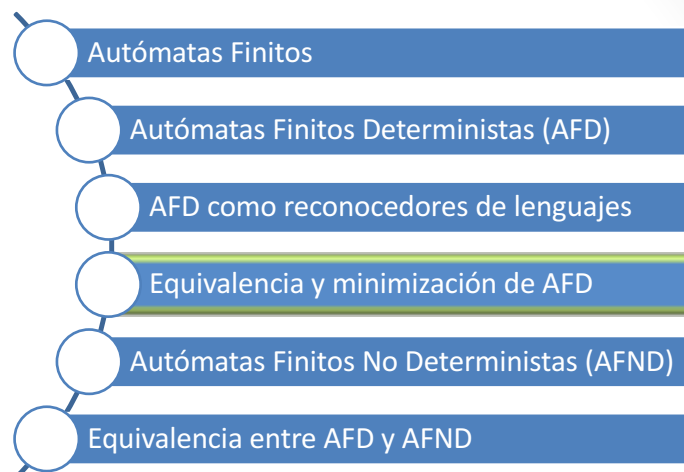
- Hallar el AFD conexo equivalente al dado:  $AF = (\{0,1\}, \{p,q,r,s\}, p, f, \{q,r,s\})$ , donde  $f$  viene dada por la tabla.

- Se eliminan todos los estados innacesibles y todas las transiciones (i.e. arcos) que salen desde dichos estados innacesibles.

	0	1
p	r	p
*q	r	p
*r	r	p
*s	s	s

- Indicar, además el lenguaje reconocido por ambos AFD's (original y conexo).

[ 25 ]



[ 26 ]





## AFD. Equivalencia y Minimización

- Es posible tener varios autómatas que reconozcan el mismo lenguaje.
- Para todo autómata se puede obtener un autómata equivalente (i.e. reconoce el mismo lenguaje) donde el número de estados del autómata sea el mínimo.
- **¿Por qué interesa obtener el mínimo?** (Apartado 4.4 Libro 2 bibliografía)

[ 27 ]



## AFD. Equivalencia y Minimización

### ¿Por qué interesa obtener el AFD mínimo? (Ap. 4.3 y 4.4 Libro 2 bibliografía)

- Se dispone de un descriptor del lenguaje (lenguaje regular): gramática tipo 3, AFD, AFND, expresión regular.
- Se plantean problemas de decisión:
  - ¿El lenguaje descrito es vacío?
  - ¿Existe una determinada cadena  $w$  en el lenguaje descrito?
  - ¿Dos descripciones de un lenguaje describen realmente el mismo lenguaje?
    - Nota: usualmente los lenguajes son infinitos, con lo que no es posible plantear la pregunta y recorrer el conjunto INFINITO de cadenas.

- Los algoritmos para responder a las dos primeras preguntas son sencillos.  
¿Pero y para la última pregunta ?

**¿Dos descripciones de un lenguaje describen realmente el mismo lenguaje?**

**Consecuencia de esta comprobación: es necesario obtener el AFD mínimo equivalente**

[ 28 ]



## AFD. Equivalencia y Minimización

### Teoremas:

- **Equivalencia de estados:**  
 $p E q$ , donde  $p, q \in Q$ , si  $\forall x \in \Sigma^*$  se verifica que  

$$f'(p, x) \in F \Leftrightarrow f'(q, x) \in F$$
- **Equivalencia de orden (o de longitud) "n"**  
 $p E_n q$ , donde  $p, q \in Q$ , si  $\forall x \in \Sigma^* / |x| \leq n$  se verifica que  

$$f'(p, x) \in F \Leftrightarrow f'(q, x) \in F$$

**$E$  y  $E_n$  son relaciones de equivalencia.**

[ 29 ]



## AFD. Equivalencia y Minimización

### Equivalencia de estados – Casos particulares:

- **$E_0$** , x palabra  $|x| \leq 0 \Rightarrow x = \lambda$  se verifica que  
 $p E_0 q$ ,  $\forall p, q \in Q$ , si  $\forall x \in \Sigma^* / |x| \leq 0$  se verifica que  

$$f'(p, x) \in F \Leftrightarrow f'(q, x) \in F$$
  
 $x$  es lambda  

$$f'(p, x) = f'(p, \lambda) = p \text{ (por definición de } f')$$
  

$$f(p, \lambda) \in F \Leftrightarrow f(q, \lambda) \in F \rightarrow p \in F \Leftrightarrow q \in F$$

Todos los estados finales de son  $E_0$  equivalentes.

- ✓  $p, q \in F$  se cumple que  $p E_0 q$
- ✓  $p, q \in Q - F$  se cumple que  $p E_0 q$

[ 30 ]



# AFD. Equivalencia y Minimización

## Equivalencia de estados – Casos particulares:

- $E_1$ , x palabra  $|x| \leq 1$ , ( $x \in \Sigma$ ) se verifica que

$p E_1 q, \forall p, q \in Q$ , si  $\forall x \in \Sigma^* / |x| \leq 1$  se verifica que

$$f'(p, x) \in F \Leftrightarrow f'(q, x) \in F$$

x es lambda o símbolo del alfabeto.

$$f'(p, x) = f'(p, a) = f(p, a) \quad \text{ó} \quad f'(p, x) = f'(p, \lambda) = p \quad (\text{por definición de } f')$$

$$f(p, a) \in F \Leftrightarrow f(q, a) \in F$$

Partiendo de p y q con una sola transición se debe llegar a un estado final para ambos casos o uno no final para ambos casos.

[ 31 ]



# AFD. Equivalencia y Minimización

## Propiedades

Nota: en estas expresiones matemáticas, "n" no significa  $|Q|$

- Lema:  $p E q \Rightarrow p E_n q, \forall n, p, q \in Q$
- Lema:  $p E_n q \Rightarrow p E_k q, \forall n > k$
- Lema:  $p E_{n+1} q \Leftrightarrow p E_n q \text{ and } f(p, a) E_n f(q, a) \forall a \in \Sigma$

- **Teorema:**  $p E q \Leftrightarrow p E_{n-2} q$ , donde  $n = |Q| > 1$

Aquí "n" sí significa  $|Q|$

(Teorema 5.1 (pag 117 libro 4 bibliografía))

$p E q$  sii  $\forall x \in \Sigma^*, |x| = m \leq n-2$  se verifica que  $f(p, x) \in F \Leftrightarrow f(q, x) \in F$

$m = n-2$  es el valor más pequeño que cumple este teorema

(n-1 sí lo cumple, pero n-3 no se garantiza que se cumpla)

[ 32 ]



## AFD. Equivalencia y Minimización

"E" es una relación de equivalencia. ¿Qué significa Q/E?

- Q/E es una partición de Q,
- $Q/E = \{C_1, C_2, \dots, C_m\}$ , donde  $C_i \cap C_j = \emptyset$ 
  - $p E q \Leftrightarrow (p, q \in C_i)$ , por lo tanto

$$\forall x \in \Sigma^* \text{ se verifica que } f'(p, x) \in C_i \Leftrightarrow f'(q, x) \in C_i$$

Nota: en libro 1 biblio,  $p, q \in C_i$  se representa por  $p = q = C_i$

- Para la relación de orden n
  - $E_n: Q/E_n = \{C_1, C_2, \dots, C_m\}$ ,  $C_i \text{ intersección } C_j = \emptyset$
  - $p E_n q \Leftrightarrow p, q \in C_i$
  - por lo tanto  $\forall x \in \Sigma^*, |x| \leq n$  se verifica que

$$f'(p, x) \in C_i \Leftrightarrow f'(q, x) \in C_i$$



[ 33 ]

## AFD. Equivalencia y Minimización

Propiedades. (Lemas)

- Lema: Si  $Q/E_n = Q/E_{n+1} \Rightarrow Q/E_n = Q/E_{n+i} \forall i = 0, 1, \dots$
- Lema: Si  $Q/E_n = Q/E_{n+1} \Rightarrow Q/E_n = Q/E$  conjunto cociente
- Lema: Si  $|Q/E_0| = 1 \Rightarrow Q/E_0 = Q/E_1$
- Lema:  $n = |Q| > 1 \Rightarrow Q/E_{n-2} = Q/E_{n-1}$
- $p E_{n+1} q \Leftrightarrow (p E_n q \text{ and } f(p, a) E_n f(q, a) \forall a \in \Sigma)$



[ 34 ]

## AFD. Equivalencia y Minimización

### Interpretación lemas anteriores:

El objetivo es obtener la partición  $Q/E$ , puesto que será el autómata mínimo, sin estados equivalentes .

- En cuanto se obtienen dos particiones consecutivas  $Q/E_k = Q/E_{k+1}$ , se para.
- Para obtener  $Q/E$ , hay que empezar por  $Q/E_0$ ,  $Q/E_1$ , etc.
- Para obtener  $Q/E$ , hay que obtener  $Q/E_{n-2}$  en el peor caso, ya que si se obtiene  $Q/E_{n-k} = Q/E_{n-k+1}$ , con  $k \geq 3$ , se habría obtenido ya  $Q/E$ .
- El lema  $p E_{n+1} q \Leftrightarrow p E_n q \text{ and } f(p,a) E_n f(q,a) \forall a \in \Sigma$ , permite es extender la equivalencia de orden  $n$  desde  $E_0$  y  $E_1$

[ 35 ]



## AFD. Equivalencia y Minimización

### Teorema:

$$pEq \Leftrightarrow pE_{n-2}q \text{ donde } |Q| = n > 1 (**)$$

Es decir,  $p E q \text{ sii } \forall x \in \Sigma^*, |x| \leq n-2, f'(p,x) \in F \Leftrightarrow f'(q,x) \in F$

$n-2$  es el valor más pequeño que cumple este teorema

[ 36 ]



## AFD. Equivalencia y Minimización

Algoritmo formal para obtener  $Q/E$ :

1.  $Q/E_0 = \{ F, \text{no } F \}$

1ª división en función de si son o no estados finales.

2.  $Q/E_{i+1}$

partiendo de  $Q/E_i = \{C_1, C_2, \dots, C_n\}$ , se construye  $Q/E_{i+1}$ :

$p$  y  $q$  están en la misma clase si:

$p, q \in C_k \in Q/E_i \forall a \in \Sigma \Rightarrow f(p,a) \text{ y } f(q,a) \in C_m \in Q/E_i$

3. Si  $Q/E_i = Q/E_{i+1}$  entonces  $Q/E_i = Q/E$

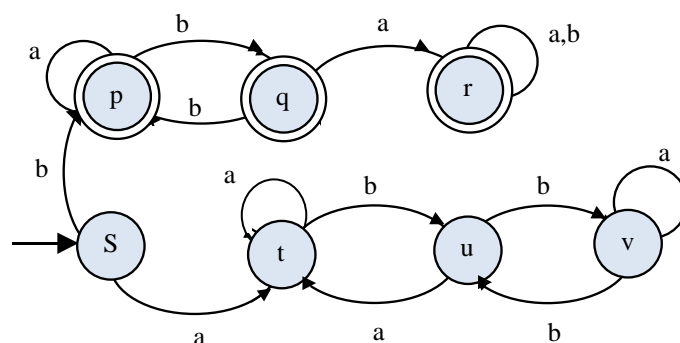
Si no, repetir el paso 2 partiendo de  $Q/E_{i+1}$

[ 37 ]



## AFD. Equivalencia y Minimización

- Ejercicio: Hallar el AFD mínimo equivalente



[ 38 ]



## AFD. Equivalencia

### Autómatas Equivalentes:

- **Estados equivalentes en AFD' s distintos:**
  - Sean 2 AFD' s:  $(\Sigma, Q, f, q_0, F)$  y  $(\Sigma', Q', f', q_0', F')$
  - Los estados  $p, q$  /  $p \in Q$  y  $q \in Q'$  son equivalentes ( $pEq$ ) si se verifica que  $f''(p, x) \in F \Leftrightarrow f''(q, x) \in F' \quad \forall x \in \Sigma^*$
- **Estados equivalentes en AFD' s distintos:**
  - Dos AFD' s son equivalentes si reconocen el mismo lenguaje, es decir: Si  $f(q_0, x) \in F \Leftrightarrow f(q_0', x) \in F' \quad \forall x \in \Sigma^*$ . Es decir:
  - **Dos AFD' s son equivalentes si lo son sus estados iniciales:  $q_0 E q_0'$**

[ 39 ]



## AFD. Equivalencia

### ¿Qué es la suma directa de 2 AFD's?

Sean 2 AFD' s:

$$A1 = (\Sigma, Q_1, f_1, q_{01}, F_1)$$

$$A2 = (\Sigma', Q_2, f_2, q_{02}, F_2)$$



Donde  $Q_1 \cap Q_2 = \phi$

y  $\Sigma = \Sigma'$

Se llama **suma directa de A1 y A2** al AF A:

$$A = A1 + A2 = (\Sigma, Q_1 \cup Q_2, f, q_0, F_1 \cup F_2), \text{ donde:}$$

$q_0$  es el estado inicial de uno de los AF' s

$$f: f(p, a) = f_1(p, a) \text{ si } p \in Q_1$$

$$f(p, a) = f_2(p, a) \text{ si } p \in Q_2$$

[ 40 ]



## AFD. Equivalencia

□ **Teorema:** (el teorema (\*\*)) aplicado a la suma directa de dos autómatas):

sean  $A_1, A_2$  /  $Q_1 \cap Q_2 = \emptyset$ ,  $|Q_1| = n_1$ ,  $|Q_2| = n_2$

$A_1 \equiv A_2$  si  $q_{01} \equiv q_{02}$  en  $A = A_1 + A_2$

Es decir, si  $A_1$  y  $A_2$  aceptan las mismas palabras  $x$  /  $|x| \leq n_1 + n_2 - 2$

además,  $n_1 + n_2 - 2$  es el valor mínimo que cumple el teorema

[ 41 ]



## AFD. Equivalencia

Autómatas equivalentes, comprobación:

**Algoritmo para comprobar la equivalencia de AFDs**

1. Se hace la suma directa de los dos AFD's
2. Se hace Q/E del AFD suma
3. Si los dos estados iniciales están en la misma clase de equivalencia de Q/E  $\Rightarrow$  los 2 AFD's son equivalentes

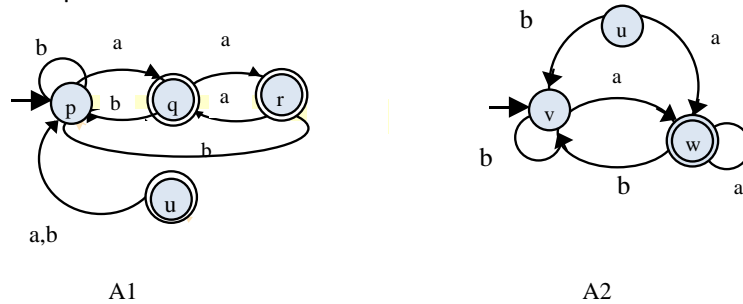
[ 42 ]





## AFD. Equivalencia

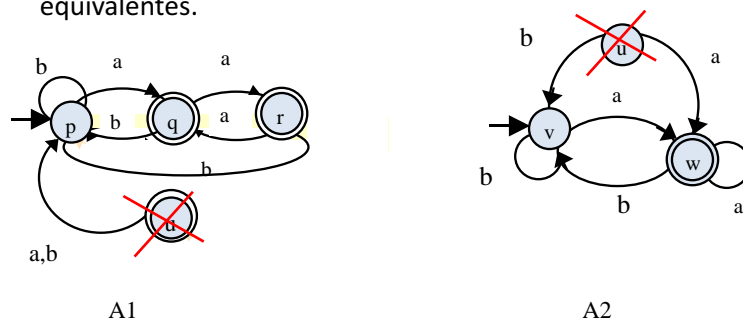
- **Ejercicio:** Comprobar que los autómatas A1 y A2 son equivalentes.



[ 43 ]

## AFD. Equivalencia

- **Ejercicio:** Comprobar que los autómatas A1 y A2 son equivalentes.

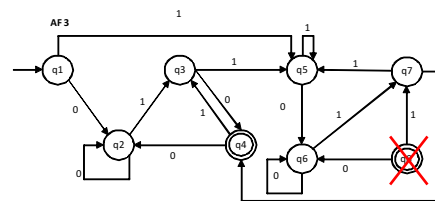
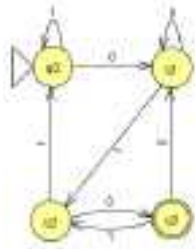


Los estados u de ambos autómatas son inaccesibles y habría que eliminarlos antes de ver si son equivalentes.

[ 44 ]

## AFD. Equivalencia

- Ejercicio 7 de la hoja 2 de ejercicios: Comprobar si los dos AFD son equivalentes (obteniendo el mínimo para cada uno).



[ 45 ]

## AFD. Equivalencia

- Ejercicio 7 de la hoja 2 de ejercicios: AF1 es mínimo. AF2:
- $Q/E0 = \{\{q4, q8\}, \{q1, q2, q3, q5, q6, q7\}\} = C1, C2$  **OJO, Q8 es inaccesible y habría que quitarlo.** Aparece tachado en la solución.
- $Q/E1 = \{\{q4, q8\}, \{q1, q2, q5, q6\}, \{q3, q7\}\} = C1, C2, C3$
- $Q/E2 = \{\{q4, q8\}, \{q1, q5\}, \{q2, q6\}, \{q3, q7\}\} = C1, C2, C3, C4$
- $Q/E3 = Q/E2 = \{\{q4, q8\}, \{q1, q5\}, \{q2, q6\}, \{q3, q7\}\} = C1, C2, C3, C4$

	0	1	0	1	0	1	0	1
->q1	q2	q5	C2	C2	C2	C2	C3	C2
q2	q2	q3	C2	C2	C2	C3	C3	C4
q3	q4	q5	C1	C2	c1	C2	c1	C2
*q4	q2	q3	C2	C2	C2	C3	C3	C4
q5	q6	q5	C2	C2	C2	C2	C3	C2
q6	q6	q7	C2	C2	C2	C3	C3	C4
q7	q4	q5	c1	C2	c1	C2	c1	C2
*q8	q6	q7	C2	C2	C2	C3	C3	C4

- El AF2 y el AF1 son ISOMORFOS y por tanto son equivalentes.

[ 46 ]

## AFD. Equivalencia

- Sean dos autómatas:
  - $A_1 = (\Sigma, Q_1, f_1, q_{01}, F_1)$  y  $A_2 = (\Sigma', Q_2, f_2, q_{02}, F_2)$ , tales que  $|Q_1| = |Q_2|$
- Se dice que  $A_1$  y  $A_2$  **son isomorfos**, si existe una aplicación biyectiva  $i : Q_1 \rightarrow Q_2$  que cumple:
  1.  $i(q_{01}) = q_{02}$ , es decir, los estados iniciales son correspondientes
  2.  $q \in F_1 \Leftrightarrow i(q) \in F_2$  es decir, los estados finales son correspondientes
  3.  $i(f_1(q, a)) = f_2(i(q), a) \quad \forall a \in \Sigma \quad q \in Q_1$
- En definitiva, a cada estado le corresponde otro equivalente que solo se diferencia en el nombre de sus estados.
- Dos AFDs isomorfos, también son equivalentes y reconocen el mismo **lenguaje**.

[ 47 ]



## AFD. Minimización

Sea el AFD,  $A = (\Sigma, Q, f, q_0, F)$ :

1. Partir del AFD conexo, i.e. eliminar estados inaccesibles desde el estado inicial
2. Construir  $Q/E$  del autómata conexo
3. El AFD mínimo, salvo isomorfismos, es:

$$A' = (\Sigma, Q', f', q_0', F')$$

donde:

$$Q' = Q/E$$

$$f' \text{ se construye: } f'(C_i, a) = C_j \text{ si } \exists q \in C_i, p \in C_j / f(q, a) = p$$

$$q_0' = C_0 \text{ si } q_0 \in C_0, C_0 \in Q/E$$

$$F' = \{C / C \text{ contiene al menos un estado de } F (\exists \text{ un } q \in F \text{ tal que } q \in C)\}$$

### COROLARIO:

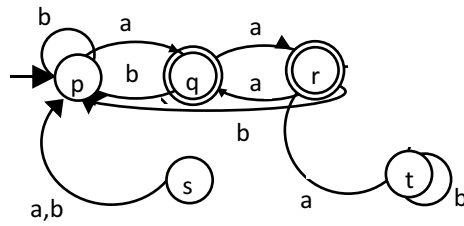
2 AFD's son equivalentes si sus AF mínimos respectivos son isomorfos.

[ 48 ]

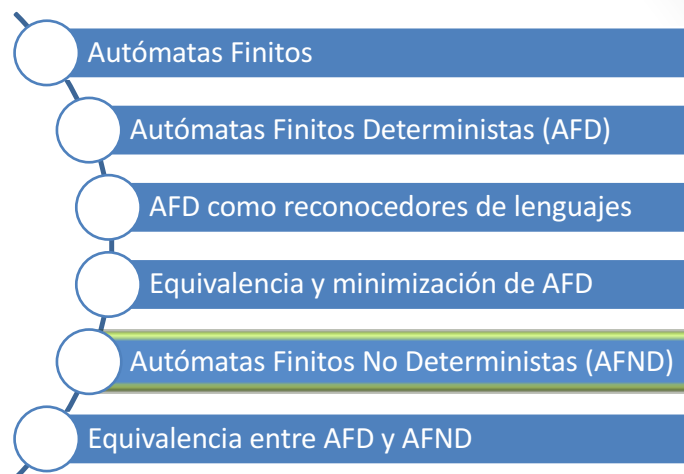


## AFD. Ejercicio

Hallar el AFD mínimo equivalente al dado:



[ 49 ]



[ 50 ]

# Autómatas Finitos No Deterministas

## Definiciones de AFND :

1. AFND =  $(\Sigma, Q, f, q_0, F)$ , donde

- $f: Q \times (\Sigma \cup \lambda) \rightarrow Q$  es No determinista,  
es decir, por ejemplo:  $f(p, a) = \{q, r\}$  y  $f(p, \lambda) = \{q, r\}$

2. AFND =  $(\Sigma, Q, f, q_0, F, T)$ , donde

- $f: Q \times \Sigma \rightarrow P(Q)$ : conjunto de las partes de  $Q$
- $T$ : Relación definida sobre pares de elementos de  $Q$ .

$pTq = (p, q) \in T$  si está definida la transición  $f(p, \lambda) = q$

Nota: "T" es la definición formal de la transición  $\lambda$ .

[ 51 ]



# Autómatas Finitos No Deterministas

Ejemplo: Sea el AFND siguiente:

$A = (\{a, b\}, \{p, q, r, s\}, f, p, \{p, s\}, T = \{(q, s), (r, r), (r, s), (s, r)\})$  donde  $f$ :

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

La tabla de transiciones es

	a	b	$\lambda$
$\rightarrow *p$	q		
q	{p,r,s}	p,r	s
r		p,s	r,s
*s			r

[ 52 ]



## AFNDs. Función de Transición extendida a palabras

- Se define a partir de  $f$ , una función de transición  $f''$ , que actúa sobre palabras de  $\Sigma^*$ ;

$f''$  es la función de transición sobre palabras.

- Es una aplicación:  $f'': Q \times \Sigma^* \rightarrow P(Q)$ . Donde :

$$1. f''(q, \lambda) = \{p / q T^* p \ \forall q \in Q\} \quad (T^* \text{ se define más adelante})$$

donde se cumple que  $q \in f'(q, \lambda)$

$$2. \text{ sea } x = a_1 a_2 a_3 \dots a_n, \ n > 0$$

$$f''(q, x) = \{p / p \text{ es accesible desde } q \text{ por medio de la palabra } \lambda^* a_1 \lambda^* a_2 \lambda^* a_3 \lambda^* \dots \lambda^* a_n \lambda^* \ \forall q \in Q\}$$

es idéntica a  $x$

Lectura recomendada: Apartado 3.3.4 del primer libro de la bibliografía básica

[ 53 ]



## AFNDs. Función de Transición extendida a palabras

### Calculo de $T^*$

Sea AFND =  $(\Sigma, Q, f, q_0, F, T)$ .

- Para calcular  $f''$  es necesario extender las transiciones con una  $\lambda$  a  $\lambda^*$ , es decir calcular  $T^*$  del AFND =  $(\Sigma, Q, f, q_0, F, \underline{T})$
- Para ello existe el método formal de las matrices booleanas, o el método de la matriz de pares (estado, estado).

[ 54 ]



## AFNDs. Función de Transición extendida a palabras

### Calculo de $T^*$ . Método de la matriz de pares de estados

- Se construye una matriz con tantas filas como estados.
- En la 1ª columna se coloca el par correspondiente al estado en cuestión, es decir, por ej.  $(p,p)$  puesto que cada estado es accesible desde sí mismo.
- En las columnas siguientes se añaden las transiciones  $\lambda$  definidas en el AFND, considerando si el hecho de añadirlas permite extender alguna transición más.
  - Pej. Si existe la transición  $\lambda (q,r)$  y se añade la transición  $\lambda (r,s)$ , habrá que añadir asimismo, la transición  $(q,s)$ .
- Cuando no sea posible añadir ningún par más, se habrá terminado  $T^*$

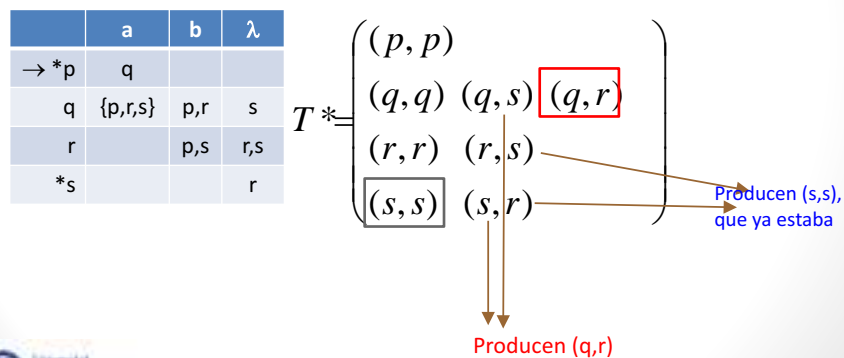
[ 55 ]



## AFNDs. Función de Transición extendida a palabras

### Calculo de $T^*$ . Ejemplo 1:

- Sea el AFND: A, definido anteriormente donde  $T = \{(q,s), (r,r), (r,s), (s,r)\}$ . Se trata de calcular  $T^*$



[ 56 ]



## AFNDs. Función de Transición extendida a palabras

### Calculo de $T^*$ . Ejemplo 2:

- Se extiende la tabla de transición anterior para contener  $T^*$ , insertando una nueva columna correspondiente a  $\lambda^*$

	a	b	<del><math>\lambda</math></del>	$\lambda^*$
$\rightarrow^*p$	q			p
q	p,r,s	p,r	<del>s</del>	q,s,r
r		p,s	<del>r,s</del>	r,s
$*s$			<del>r</del>	r,s

[ 57 ]



## AFNDs. Función de Transición extendida a palabras

### Calculo de $T^*$ . Ejemplo 3:

- Y ahora se calcula la tabla de transición correspondiente a  $f''$ , cambiando las transiciones con a por  $\lambda^*a\lambda^*$  y las de b por  $\lambda^*b\lambda^*$ .

	a	b	<del><math>\lambda</math></del>	$\lambda^*$		$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$
$\rightarrow^*p$	q			p	$\rightarrow^*p$	q,r,s	$\Phi$
q	p,r,s	p,r	<del>s</del>	q,s,r	q	p,r,s	p,r,s
r		p,s	<del>r,s</del>	r,s	r	$\Phi$	p,r,s
$*s$			<del>r</del>	r,s	$*s$	$\Phi$	p,r,s

[ 58 ]





## AFND. Lenguaje aceptado por un AFND

- Una palabra  $x \in \Sigma^*$  es aceptada por un AFND si:
  - $f'(q_0, x)$  y  $F$  tienen al menos un elemento común, es decir, que  $f'(q_0, x)$  contiene al menos un estado final.
- El conjunto de todas las palabras aceptadas por un AFND es el lenguaje aceptado por ese AFND.

Formalmente:

$$L_{\text{AFND}} = \{x / x \in \Sigma^* \vee \exists q_0 \rightarrow F\} = \{x / x \in \Sigma^* \vee f'(q_0, x) \cap F \neq \emptyset\}$$

[ 59 ]



## AFND. Lenguaje aceptado por un AFND

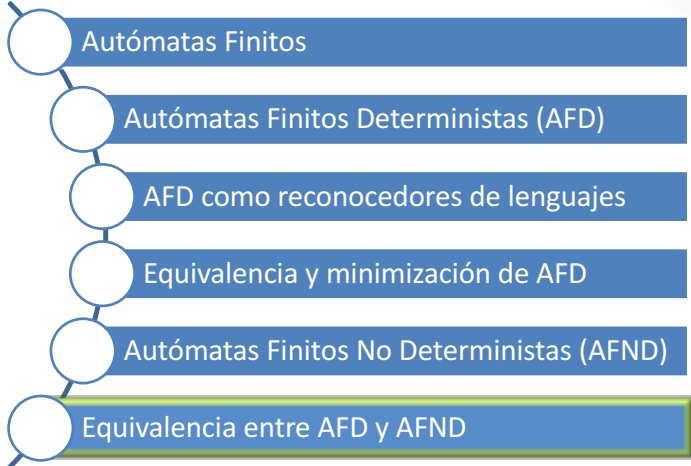
- Al ser un AFND, desde  $q_0$  puede haber más de un camino para la palabra " $x$ ", y " $x$ " es aceptada sólo con que uno de los caminos lleve a un estado final.
- Además:
 

$\lambda \in L_{\text{AFND}}$  si :

  - $q_0 \in F$  ó
  - $\exists$  un estado final,  $q \in F$ , tal que está en relación  $T^*$  con  $q_0$  ( $q_0 T^* q$ )

[ 60 ]





Autómatas Finitos

Autómatas Finitos Deterministas (AFD)

AFD como reconocedores de lenguajes

Equivalencia y minimización de AFD

Autómatas Finitos No Deterministas (AFND)

Equivalencia entre AFD y AFND

[ 61 ]

Universidad Carlos III de Madrid

## AFD equivalente a un AFND

- Dado un AFND siempre es posible encontrar un AFD que reconozca el mismo lenguaje:
  - El conjunto de los  $L_{AFND}$  = al conjunto de los  $L_{AFD}$ .
  - Un AFND no es más potente que un AFD, sino que un AFD es un caso particular de AFND.

**Paso de AFND a AFD:**

- Sea el AFND  $A = (\Sigma, Q, f, q_0, F, T)$ .
- Se define a partir de A el AFD B, donde:
  - $B = (\Sigma, Q', f', q_0', F')$ , tal que:
    - $Q' = P(Q)$  conjunto de las partes de Q que incluye a Q y a  $\emptyset$ .
    - $q_0' = f'(q_0, \lambda)$  ( $f'$  extensión a palabra de f, i.e. todos los estados que tengan relación  $T^*$  con  $q_0$ ).
    - $F' = \{C / C \in Q' \text{ y } \exists q \in C / q \in F\}$
    - $f'(C, a) = \{C' / C' = \bigcup_{q \in C} f(q, a)\}$

[ 62 ]

Universidad Carlos III de Madrid

## AFD equivalente a un AFND

- EXPLICACIÓN Paso de AFND a AFD:

1. Calculamos  $\lambda^*$  (usando  $T^*$ )
2. Calculamos  $\lambda^*a \lambda^* \forall a \in \Sigma$
3.  $\lambda^*a \lambda^* = a$
4. Transformamos los caminos múltiples en estados combinados:  
p, q, r pasa a ser {pqr}
5. Construimos en AFD conexo considerando que las transiciones de {pqr} son las resultantes de hacer la unión de las transiciones de cada uno de los estados que lo forman.
6. Será estado inicial  $q_0$  o  $q_0' = f'(q_0, \lambda)$  ( $f'$  extensión a palabra de  $f$ , i.e. todos los estados que tengan relación  $T^*$  con  $q_0$ ).
7. Será estado final TODO estado combinado que contenga a un estado final o aquellos finales que estén conectados.

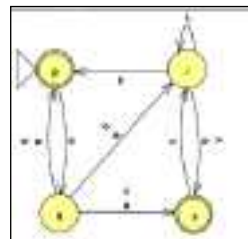
[ 63 ]



## AFD equivalente a un AFND. Ejemplo

- Obtener el AFD correspondiente al siguiente AFND

	a	b	$\lambda$
$\rightarrow^*p$	q		
q	p,r,s	p,r	s
r		p,s	r,s
$*s$			r



- Pasos:

1. Eliminar transiciones  $\lambda$ 
  - a) Determinar  $\lambda^*$  (el cierre de las transiciones  $\lambda$ ,  $T^*$ )
  - b) Obtener la tabla sin transiciones  $\lambda$
2. Aplicar algoritmo de creación de nuevos estados que pertenecen a  $P(Q)$ , añadiendo su transiciones.

[ 64 ]



## AFD equivalente a un AFND. Ejemplo

### 1. Eliminar transiciones $\lambda$

- a) Determinar  $\lambda^*$  (el cierre de las transiciones  $\lambda$ ) a partir de la tabla de transiciones.

	a	b	$\lambda$
$\rightarrow^*p$	q		
q	p,r,s	p,r	s
r		p,s	r,s
$*s$			r



	a	b	<del><math>\lambda</math></del>	$\lambda^*$
$\rightarrow^*p$	q			<b>p</b>
q	p,r,s	p,r	<del>s</del>	<b>q,s,r</b>
r		p,s	<del>r,s</del>	<b>r,s</b>
$*s$			<del>r</del>	<b>s,r</b>

[ 65 ]



## AFD equivalente a un AFND. Ejemplo

### 1. Eliminar transiciones $\lambda$

- a) Determinar  $\lambda^*$  (el cierre de las transiciones  $\lambda$ )

	a	b	<del><math>\lambda</math></del>	$\lambda^*$
$\rightarrow^*p$	q			<b>p</b>
q	p,r,s	p,r	<del>s</del>	<b>q,r,s</b>
r		p,s	<del>r,s</del>	<b>r,s</b>
$*s$			<del>r</del>	<b>r,s</b>

- b) Obtener la tabla sin transiciones  $\lambda$   
(transiciones con entrada  $\lambda^*a\lambda^*$ , para cada elemento, a, del alfabeto  $\Sigma$ )

	$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$
$\rightarrow^*p$	q,r,s	$\emptyset$
q	p,r,s	p,r,s
r	$\emptyset$	p,r,s
$*s$	$\emptyset$	p,r,s

[ 66 ]



## AFD equivalente a un AFND. Ejemplo

2. Aplicar algoritmo de creación de nuevos estados que pertenecen a  $P(Q)$ , añadiendo su transiciones.

	$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$
$\rightarrow^*p$	$q,r,s$	$\emptyset$
$q$	$p,r,s$	$p,r,s$
$r$	$\emptyset$	$p,r,s$
$*s$	$\emptyset$	$p,r,s$

	a	b
$\rightarrow^*p$	$\{q,r,s\}$	$\emptyset$
$q$	$\{p,r,s\}$	$\{p,r,s\}$
$r$	$\emptyset$	$\{p,r,s\}$
$*s$	$\emptyset$	$\{p,r,s\}$
	$\{q,r,s\}$	$\{p,r,s\} \cup \emptyset \cup \emptyset$
		$\{p,r,s\} \cup \{p,r,s\} \cup \{p,r,s\}$

	$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$
$\rightarrow^*p$	$\{q,r,s\}$	$\emptyset$
$q$	$\{p,r,s\}$	$\{p,r,s\}$
$r$	$\emptyset$	$\{p,r,s\}$
$*s$	$\emptyset$	$\{p,r,s\}$
	$\{q,r,s\}$	$\{p,r,s\}$



[ 67 ]

## AFD equivalente a un AFND. Ejemplo

2. Aplicar algoritmo de creación de nuevos estados que pertenecen a  $P(Q)$ , añadiendo sus transiciones.

	$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$
$\rightarrow^*p$	$q,r,s$	$\emptyset$
$q$	$p,r,s$	$p,r,s$
$r$	$\emptyset$	$p,r,s$
$*s$	$\emptyset$	$p,r,s$

	$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$
$\rightarrow^*p$	$\{q,r,s\}$	$\emptyset$
$q$	$\{p,r,s\}$	$\{p,r,s\}$
$r$	$\emptyset$	$\{p,r,s\}$
$*s$	$\emptyset$	$\{p,r,s\}$
	$\{q,r,s\}$	$\{p,r,s\}$

	$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$
$\rightarrow^*p$	$\{q,r,s\}$	$\emptyset$
$q$	$\{p,r,s\}$	$\{p,r,s\}$
$r$	$\emptyset$	$\{p,r,s\}$
$*s$	$\emptyset$	$\{p,r,s\}$
	$\{q,r,s\}$	$\{p,r,s\}$

	$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$
$\rightarrow^*p$	$\{q,r,s\}$	$\emptyset$
$q$	$\{p,r,s\}$	$\{p,r,s\}$
$r$	$\emptyset$	$\{p,r,s\}$
$*s$	$\emptyset$	$\{p,r,s\}$
$\{q,r,s\}$	$\{p,r,s\}$	$\{p,r,s\}$
$\{p,r,s\}$	$\{q,r,s\} \cup \emptyset \cup \emptyset$	$\emptyset \cup \{p,r,s\} \cup \{p,r,s\}$

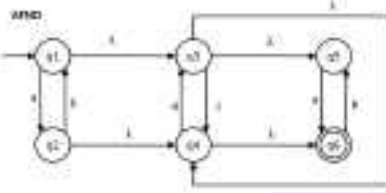
	$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$
$\rightarrow^*p$	$\{q,r,s\}$	$\emptyset$
$\{q,r,s\}$	$\{p,r,s\}$	$\{p,r,s\}$
$\{p,r,s\}$	$\{q,r,s\}$	$\{p,r,s\}$



[ 68 ]

## Ejercicio 12 – Parte 3

Indica el grafo del Autómata Finito Determinista que se corresponde con el siguiente AFND:



	a	b	c	d	$\lambda$
$\rightarrow q1$	q2				q3
q2		q1			q4
q3			q4		{q4, q5}
q4				q3	q6
q5	q6				
q6*		q5			



[ 69 ]

## Ejercicio 12 – Parte 3

	a	b	c	d	$\lambda$	$\lambda^*$	$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$	$\lambda^*c\lambda^*$	$\lambda^*d\lambda^*$
$\rightarrow q1$	q2				q3	{q1,q3,q4,q5,q6}	{q2,q4,q6}	{q5}	{q4,q6}	{q3,q4,q5,q6}
q2		q1			q4	{q2,q4,q6}	--	{q1,q3,q4,q5,q6}	--	{q3,q4,q5,q6}
q3			q4		{q4,q5}	{q3,q4,q5,q6}	{q6}	{q5}	{q4,q6}	{q3,q4,q5,q6}
q4				q3	q6	{q4,q6}	--	{q5}	--	{q3,q4,q5,q6}
q5	q6					{q5}	{q6}	--	--	--
q6*		q5				{q6}	--	{q5}	--	--



[ 70 ]

## Ejercicio 12 – Parte 3

	$\lambda^*$	$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$	$\lambda^*c\lambda^*$	$\lambda^*d\lambda^*$
$\rightarrow q1$	{q1,q3,q4,q5,q6}	{q2,q4,q6}	{q5}	{q4,q6}	{q3,q4,q5,q6}
q2	{q2,q4,q6}	--	{q1,q3,q4,q5,q6}	--	{q3,q4,q5,q6}
q3	{q3,q4,q5,q6}	{q6}	{q5}	{q4,q6}	{q3,q4,q5,q6}
q4	{q4,q6}	--	{q5}	--	{q3,q4,q5,q6}
q5	{q5}	{q6}	--	--	--
q6*	{q6}	--	{q5}	--	--

[ 71 ]



## Ejercicio 12 – Parte 3

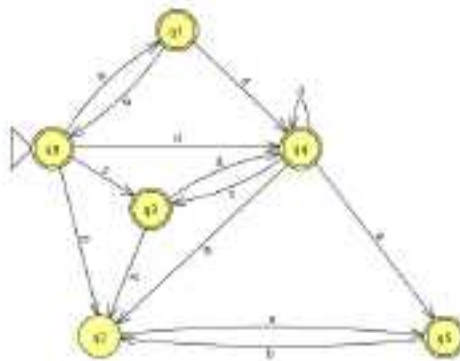
	a	b	c	d
$\rightarrow \{q1,q3,q4,q5,q6\}^*$	{q2,q4,q6}	{q5}	{q4,q6}	{q3,q4,q5,q6}
$\{q2,q4,q6\}^*$	$\phi$	{q1,q3,q4,q5,q6}	$\phi$	{q3,q4,q5,q6}
{q5}	{q6}	$\phi$	$\phi$	$\phi$
$\{q4,q6\}^*$	$\phi$	{q5}	$\phi$	{q3,q4,q5,q6}
$\{q3,q4,q5,q6\}^*$	{q6}	{q5}	{q4,q6}	{q3,q4,q5,q6}
$\{q6\}^*$	$\phi$	{q5}	$\phi$	$\phi$
$\phi$	$\phi$	$\phi$	$\phi$	$\phi$

[ 72 ]



## Ejercicio 12 – Parte 3

Solución:



-> {q1,q3,q4,q5,q6}\*=Q0

{q2,q4,q6}\*=Q1

{q5}=Q2

{q4,q6}\*=Q3

{q3,q4,q5,q6}\*=Q4

{q6}\*=Q5

[ 73 ]



Por simplicidad en el dibujo, el estado sumidero no se ha representado

## 3. Autómatas Finitos

Grado Ingeniería Informática

Teoría de Autómatas y Lenguajes Formales



Universidad  
Carlos III de Madrid  
www.uc3m.es





## **Parte IV**

### **TEMA 4. Gramáticas y Lenguajes Formales**



# 4. Lenguajes y Gramáticas Formales

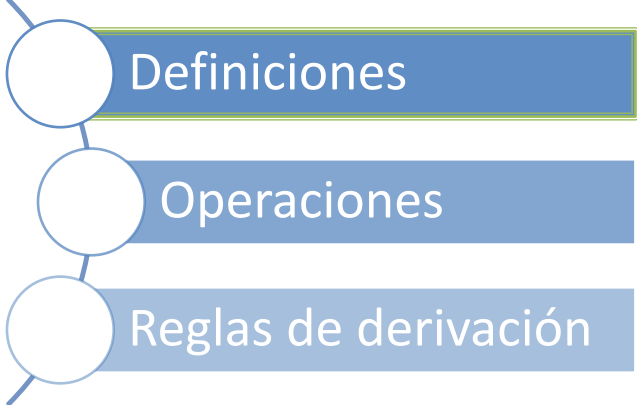
Grado Ingeniería Informática  
Teoría de Autómatas y Lenguajes Formales



LENGUAJES FORMALES



[ 2 ]



Definiciones

Operaciones

Reglas de derivación

[ 3 ]

Universidad  
Cádiz (UCA)  
Escuela de Ingeniería

# Definiciones

## Símbolo

Entidad abstracta, no se define (análogo al punto en geometría). Son letras, dígitos, caracteres, etc. También es posible encontrar símbolos formados por varios caracteres, pej: IF, THEN, ELSE, ...

## Alfabeto ( $\Sigma$ )

Conjunto **finito no vacío** de letras o símbolos.

Sea "a" una letra y  $\Sigma$  un alfabeto, si a pertenece a ese alfabeto  $\Rightarrow a \in \Sigma$

Ejemplos:

$$\Sigma_1 = \{A, B, C, \dots, Z\}$$

$$\Sigma_2 = \{0, 1\}$$

$$\Sigma_3 = \{IF, THEN, ELSE, BEGIN, END\}$$

[ 4 ]

Universidad  
Cádiz (UCA)  
Escuela de Ingeniería

# Definiciones

**Palabra, cadena, tira:** toda secuencia finita de símbolos del alfabeto.

Ejemplos:

palabras sobre  $\Sigma_1$  JUAN, ISABEL, etc.

palabras sobre  $\Sigma_2$  00011101

palabras sobre  $\Sigma_3$  IFTHENELSEEND

**Notación:** las palabras se representan por letras minúsculas del final del alfabeto

( $x, y, z$ ), pej  $x = \text{JUAN}$ ,  $y = \text{IFTHENELSEEND}$

[ 5 ]



# Definiciones

## Longitud de palabra

Es el número de símbolos que componen una palabra

La longitud de la palabra  $x$  se representa por  $|x|$

Ejemplos:

$$|x| = |\text{JUAN}| = 4$$

$$|y| = |\text{IFTHENELSEEND}| = 13 \text{ (en } \Sigma_1 \text{)}$$

$$|y| = |\text{IFTHENELSEEND}| = 4 \text{ (en } \Sigma_3 \text{)}$$

## Palabra vacía ( $\lambda$ )

Es aquella palabra cuya longitud es cero

Se representa por  $\lambda$ ,  $|\lambda| = 0$

Sobre cualquier alfabeto es posible construir  $\lambda$

**Utilidad:** es elemento neutro en muchas operaciones con palabras y lenguajes

[ 6 ]



# Definiciones

**Universo del discurso,  $W(\Sigma)$ :** Es el conjunto de todas las palabras que se pueden formar con los símbolos de un alfabeto  $\Sigma$

- ✓ También se denomina Lenguaje Universal de  $\Sigma$
- ✓ Se representa como  $W(\Sigma)$
- ✓ Es un conjunto infinito
- ✓ Ejemplo: sea  $\Sigma_4 = \{A\}$ ,  $W(\Sigma_4) = \{\lambda, A, AA, AAA, \dots\}$  con un número  $\infty$  de palabras

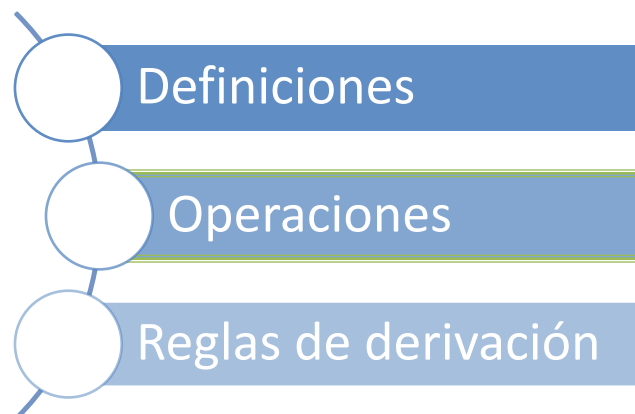
**COROLARIO:**

$\forall \Sigma, \lambda \in W(\Sigma) \Rightarrow$  La palabra vacía pertenece a todos los lenguajes universales de todos los alfabetos posibles

[ 7 ]



Universidad  
Carlos III de Madrid  
c3m.es



[ 8 ]



Universidad  
Carlos III de Madrid  
c3m.es



# Operaciones con palabras

Operaciones con palabras  
sobre palabras de un universo  
del discurso dado

- 1 Concatenación de palabras
- 2 Potencia
- 3 Reflexión

[ 9 ]



## 1 Concatenación de palabras

sean dos palabras  $x, y$  tal que  $x \in W(\Sigma)$ ,  $y \in W(\Sigma)$ , y sea

$$|x| = i = |x_1 x_2 \dots x_i| \text{ e } |y| = j = |y_1 y_2 \dots y_j|,$$

se llama **concatenación** de  $x$  con  $y$ , a:

$$x \cdot y = x_1 x_2 \dots x_i y_1 y_2 \dots y_j = z, \text{ donde } z \in W(\Sigma)$$

[ 10 ]





# 1 Concatenación de palabras

## Propiedades:

- ✓ Operación cerrada (o interna)
- ✓ Propiedad Asociativa
- ✓ Con elemento neutro.
- ✓ No conmutativa

## Definiciones:

- ✓ Cabeza
- ✓ Cola
- ✓ Longitud de palabra

[ 11 ]

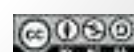


# 2 Potencia

Es la reducción de la concatenación a los casos que se refieren a una misma palabra

- ✓ potencia *i-ésima* de una palabra al resultado de concatenar esa palabra consigo misma  $i$  veces
- ✓ concatenación es asociativa  $\Rightarrow$  no especificar el orden
- ✓  $x^i = x \cdot x \cdot x \cdot \dots \cdot x$   $i$  veces
- ✓  $|x^i| = i \cdot |x|$
- ✓ se cumple:
  - $x^1 = x$
  - $x^{1+i} = x \cdot x^i = x^i \cdot x$  ( $i > 0$ )
  - $x^{j+i} = x^j \cdot x^i = x^i \cdot x^j$  ( $i, j > 0$ )
- ✓ Si se define  $x^0 = \lambda$ 
  - ( $i, j \geq 0$ )

[ 12 ]



### 3 Reflexión de una palabra

Sea la palabra  $x = a_1 \cdot a_2 \cdot a_3 \cdot \dots \cdot a_n$ ,

se denomina **palabra refleja** de  $x$ ,

$$x^{-1} = a_n \cdot a_{n-1} \cdot \dots \cdot a_2 \cdot a_1,$$

formada por los mismos símbolos en distinto orden

$$|x^{-1}| = |x|$$

[ 13 ]



## Operaciones con lenguajes

¿Qué es un Lenguaje?

Se denomina Lenguaje sobre el alfabeto  $\Sigma$  a todo subconjunto del lenguaje universal de  $\Sigma$  ( $L \subset W(\Sigma)$ )

i.e. a todo conjunto de palabras sobre un determinado  $\Sigma$

i.e. a todo conjunto de palabras generado a partir del alfabeto  $\Sigma$

[ 14 ]



# Operaciones con lenguajes

Son lenguajes especiales:

$\phi$  = Lenguaje vacío,  $\phi \subset W(\Sigma)$

$\{\lambda\}$  = Lenguaje de la palabra vacía

$\phi$  y  $\{\lambda\}$  se diferencian en el número de palabras (cardinalidad) que los forman  $C(\phi) = 0$  mientras que  $C(\{\lambda\})=1$  se parecen en que  $\phi$  y  $\{\lambda\}$  son lenguajes sobre cualquier alfabeto

Un alfabeto es uno de los lenguajes generados por el mismo:

$\Sigma \subset W(\Sigma)$ , por ejemplo el chino

[ 15 ]

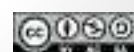


# Operaciones con lenguajes

Operaciones con lenguajes sobre un un alfabeto dado  $\Sigma$

- 1 Unión de lenguajes
- 2 Concatenación de lenguajes
- 3 **Binoide libre**
- 4 Potencia de un lenguaje
- 5 **Clausura o cierre positivo de un lenguaje**
- 6 **Iteración, clausura o cierre de un lenguaje**
- 7 Reflexión de lenguajes

[ 16 ]



# 1 Unión de lenguajes

Sean  $L_1$  y  $L_2$  definidos sobre el mismo alfabeto  $\Sigma$ ,  $L_1, L_2 \subset W(\Sigma)$ , se llama **unión** de dos lenguajes,  $L_1, L_2$  y se representa por  $L_1 \cup L_2$  al lenguaje así definido:

$$L_1 \cup L_2 = \{ x / x \in L_1 \text{ ó } x \in L_2 \}$$

Es el conjunto formado indistintamente por palabras de uno u otro de los dos lenguajes (equivale a la suma)

$$L_1 + L_2 = L_1 \cup L_2$$

[ 17 ]



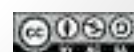
# 2 Concatenación de lenguajes

Sean  $L_1$  y  $L_2$  definidos sobre el mismo alfabeto,  $L_1, L_2 \subset W(\Sigma)$ , se llama **concatenación o producto** de dos lenguajes,  $L_1$  y  $L_2$ , y se representa por  $L_1 \cdot L_2$  al lenguaje así definido:

$$L_1 \cdot L_2 = \{ xy / x \in L_1 \text{ AND } y \in L_2 \}$$

- ✓ Es el conjunto de palabras formado por la concatenación de palabras de  $L_1$  con palabras de  $L_2$
- ✓ Definición válida para lenguajes con algún elemento.
- ✓ Y con el lenguaje vacío:  $\phi \cdot L = L \cdot \phi = \phi$

[ 18 ]



## 2 Concatenación de lenguajes

### Propiedades

- ✓ Operación cerrada
- ✓ Propiedad Asociativa
- ✓ Con elemento neutro
- ✓ Propiedad distributiva respecto a la unión.

[ 19 ]



## 3 Binoide libre

- ✓ La concatenación (monoide) de lenguajes y la unión (monoide) de lenguajes constituyen un **binoide**
- ✓ Los símbolos de  $\Sigma$  se pueden considerar conjuntos de una sola palabra
- ✓ Con  $\Sigma$ , la unión y la concatenación se puede formar cualquier lenguaje sobre dicho  $\Sigma$ . Excepto  $\phi$  y  $\{\lambda\}$ .

El alfabeto  $\Sigma$  es un conjunto de generadores para el conjunto  $L \Rightarrow L$  es el **BINOIDE LIBRE** (operaciones  $\cup$  y  $\bullet$ ) generado por  $\Sigma$

[ 20 ]



## 4 Potencia de un lenguaje

- ✓ Es la reducción de la concatenación a los casos que se refieren a un mismo lenguaje
- ✓ potencia *i-ésima* de un lenguaje al resultado de concatenar ese lenguaje consigo mismo *i* veces
- ✓ concatenación es asociativa  $\Rightarrow$  no especificar el orden
- ✓  $L^i = L \cdot L \cdot L \cdot \dots \cdot L$  ; *i* veces
- ✓ Se define  $L^1 = L$
- ✓ Se cumple:
  - $L^{1+i} = L \cdot L^i = L^i \cdot L$  ( $i > 0$ )
  - $L^{j+i} = L^i \cdot L^j$  ( $i, j > 0$ )
- ✓ Si se define  $L^0 = \{ \lambda \}$ , entonces ( $i \geq 0$ ) ( $j \geq 0$ )

[ 21 ]



## 5 Clausura o cierre positivo

Se representa como  $L^+$  y es el lenguaje

obtenido uniendo el lenguaje  $L$  con  $L^+ = \bigcup_{i=1}^{\infty} L^i$   
todas sus potencias posibles **excepto**  $L^0$

**Ninguna clausura positiva contiene a  $\lambda$ , salvo si  $\lambda \in L$**

Como  $\Sigma$  es un lenguaje sobre  $\Sigma$ , la clausura positiva de  $\Sigma$  será:

$$\Sigma^+ = \bigcup_{i=1}^{\infty} \Sigma^i = W(\Sigma) - \{\lambda\}$$

[ 22 ]



## 6 Iteración, clausura o cierre

Se representa como  $L^*$  y es el lenguaje obtenido uniendo el lenguaje  $L$  con todas sus potencias posibles.

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

Toda clausura contiene a  $\lambda$ .

\* es el operador unario de Kleene

✓ Se cumple:

$$L^* = L^+ \cup \{\lambda\}$$

$$L^+ = L^* \cdot L = L \cdot L^*$$

✓ Como  $\Sigma$  es un lenguaje sobre  $\Sigma$ , se le puede aplicar el \*:

$$\Sigma^* = W(\Sigma) \rightarrow \text{El lenguaje universal es } \Sigma^*$$

( 23 )



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

## 7 Reflexión de lenguaje

Se llama lenguaje reflejo o inverso de  $L$  y se representa por  $L^{-1}$  al lenguaje:

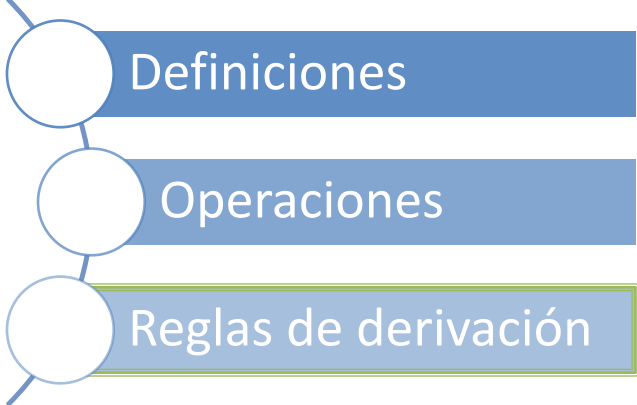
$$L^{-1} = \{x^{-1} \mid x \in L\}$$

es decir, al lenguaje formado por todas las palabras reflejas de  $L$

( 24 )



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso



Definiciones

Operaciones

Reglas de derivación

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

( 25 )

Universidad  
Cádiz (UCA)  
Escuela de Ingeniería

# Producciones

Sea  $\Sigma$  un alfabeto

- ✓ Se llama producción a un **par ordenado**  $(x, y)$  donde  $x, y \in \Sigma^*$ 
  - $x \in \Sigma^*$ , es la parte izquierda de la producción e
  - $y \in \Sigma^*$ , la parte derecha de la producción
- ✓ Se representa como:  $x ::= y$

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

( 26 )

Universidad  
Cádiz (UCA)  
Escuela de Ingeniería



## Derivación directa

- ✓ Sea  $\Sigma$  un alfabeto
- ✓ Sea  $(x, y)$  una producción sobre palabras de  $\Sigma$ ,  $x::=y$
- ✓ Sean  $v$  y  $w$  dos palabras sobre  $\Sigma$  (i.e.  $v, w \in \Sigma^*$ )

Se dice

*"w es derivación directa de v" ó*

*"v produce directamente w"  $v \rightarrow w$*

*"w se reduce directamente a v"*

si  $\exists$  dos palabras  $z, u \in \Sigma^*$  tales que  $v = z \cdot x \cdot u$  y  $w = z \cdot y \cdot u$

### COROLARIO

Si  $x ::= y$  es una producción sobre  $\Sigma \Rightarrow x \rightarrow y$   
(una regla de escritura es una derivación directa)



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 27 ]

## Derivación directa

### Ejemplos

- ✓ Sea  $\Sigma$  el alfabeto castellano de las letras mayúsculas  
y  $ME::=BA$  una producción sobre  $\Sigma$

CABALLO es derivación directa de CAMELLO

(CAMELLO produce directamente CABALLO)

- ✓ Y con la producción  $CA::=PE$  sobre  $\Sigma$

PERA es derivación directa de CARA

En el castellano no son así las cosas, existen las raíces.



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 28 ]

## Derivación directa

En un conjunto de producciones

- ✓ Sea  $\Sigma$  un alfabeto y  $P$  un conjunto de producciones sobre  $\Sigma$
- ✓ Sean  $v$  y  $w$  dos palabras sobre  $\Sigma$ ,  $v, w \in \Sigma^*$

Se dice que

$$\left. \begin{array}{l} \text{"w es derivación directa de v"} \\ \text{"v produce directamente w"} \\ \text{"w se reduce directamente a v"} \end{array} \right\} v \rightarrow w$$

si  $\exists$  dos palabras  $z, u \in \Sigma^*$  tales que  $v = z \cdot x \cdot u$  y  $w = z \cdot y \cdot u$  y se cumple  $(x::=y) \in P$



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

( 29 )

## Derivación

- ✓ Sea  $\Sigma$  un alfabeto y  $P$  un conjunto de producciones sobre  $\Sigma$
- ✓ Sean  $v$  y  $w$  dos palabras sobre  $\Sigma$ ,  $v, w \in \Sigma^*$

Se dice que

$$\left. \begin{array}{l} \text{"w es derivación de v"} \\ \text{"v produce w"} \\ \text{"w se reduce a v"} \end{array} \right\} v \rightarrow^+ w$$

si  $\exists$  una secuencia finita de palabras,  $u_0, u_1, u_2, \dots, u_n$  ( $n > 0$ )  $\in \Sigma^*$

tales que  $v = u_0$

$$\left. \begin{array}{l} u_0 \rightarrow u_1 \\ u_1 \rightarrow u_2 \\ \dots\dots\dots \\ u_{n-1} \rightarrow u_n \\ w = u_n \end{array} \right\}$$

**Derivación de longitud n**



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

( 30 )

## Relación de Thue

- ✓ Sea  $\Sigma$  un alfabeto y  $P$  un conjunto de producciones sobre  $\Sigma$
- ✓ Sean  $v$  y  $w$  dos palabras sobre  $\Sigma$ , es decir  $v, w \in \Sigma^*$

Se dice que existe una relación de Thue entre  $v$  y  $w$  y se representa por  $v \xrightarrow{*} w$  si se verifica que:

$$\left. \begin{array}{l} v \xrightarrow{*} w \\ v = w \end{array} \right\} \begin{array}{l} \text{o} \\ \text{o} \end{array} \begin{array}{l} \exists \text{ una derivación de longitud } n \text{ o} \\ \text{son iguales} \end{array}$$

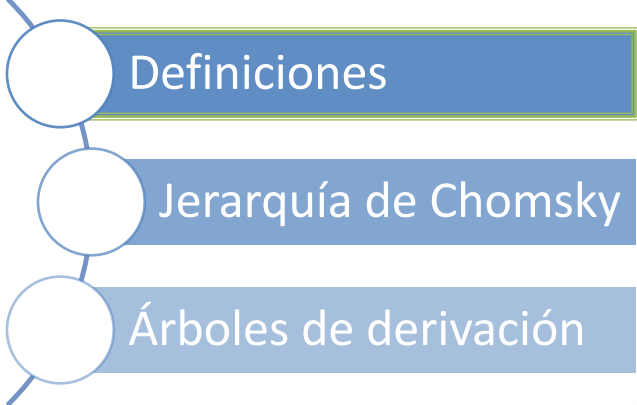
### Propiedades

- ✓ Reflexiva
- ✓ Transitiva
- ✓ Simétrica (en general NO se cumple)



## GRAMÁTICAS FORMALES





Definiciones


Jerarquía de Chomsky

Árboles de derivación

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

( 33 )

Universidad  
Cádiz (UCA)  
Escuela de Ingeniería



## ¿Qué es una gramática?

Una gramática describe la estructura de las frases y de las palabras de un lenguaje y se aplica por igual a:


- ✓ Las lenguas naturales humanas
- ✓ Lenguajes de programación.

Una gramática es un “ente formal” para especificar de manera finita el conjunto de cadenas de símbolos que constituyen un lenguaje.

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

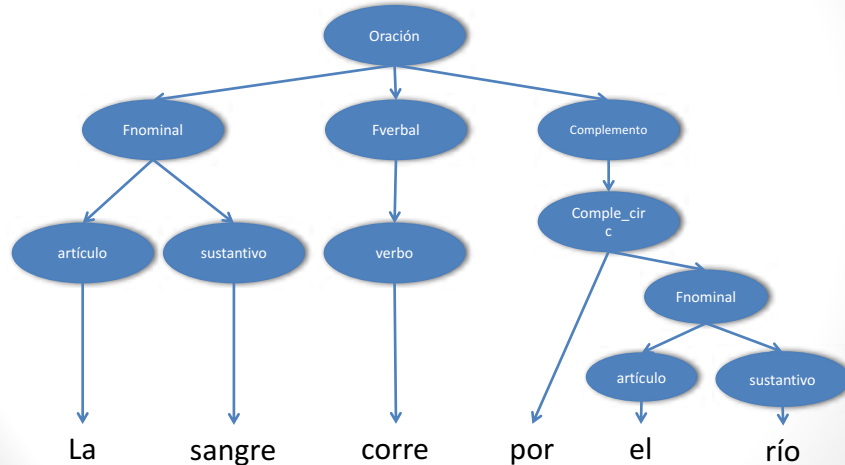
( 34 )

Universidad  
Cádiz (UCA)  
Escuela de Ingeniería



# Gramática del Castellano

Una gramática del castellano como diagrama sintáctico



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 35 ]

# Reglas de producción

Una gramática del castellano en notación de Backus

<Oración> ::= <Fnominal> <Fverbal> | <Fnominal> <Fverbal> <Complemento>

<Fnominal> ::= <Sustantivo> | <NomPr> | <Artículo> <Sustantivo>

| <Artículo> <Sustantivo> <Adjetivo>

| <Artículo> <Sustantivo> <Adjetivo>

| <Fnominal> de <Fnominal>

<Fverbal> ::= <Verb> | <Verb> <Adverbio>

<Complemento> ::= <ComDir> | <ComIn> | <ComCir> | <ComDir> <ComIn> <ComCir>

<ComDir> ::= <Fnominal> , <ComIn> ::= a <Fnominal> | para <Fnominal> | ...

<ComCir> ::= en <Fnominal> | desde <Fnominal> | cuando <Fnominal> | ...

Donde: <Sustantivo>, <Adjetivo>, <Adverbio>, <Artículo>, <NomPr>, <Verbo>, etc toman como valores palabras propias de estas categorías gramaticales



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 36 ]

## Definición

Cómo se determina (define) una gramática:  
Es una cuádrupla:  $(\Sigma_T, \Sigma_N, S, P)$ ,  $\Sigma_T$  y  $\Sigma_N$  son alfabetos:

$\Sigma_T$ : **Alfabeto de símbolos terminales.**  
Todas las cadenas del lenguaje representado por la G ( $L(G)$ ) están formadas con símbolos de este alfabeto.

$\Sigma_N$ : **Alfabeto de símbolos No Terminales.**  
Conjunto de símbolos auxiliares introducidos como elementos auxiliares para la definición de G pero que no figuran en las cadenas de  $L(G)$ .

**S:** **Axioma** o símbolo destacado.  
Es un símbolo NT a partir del que se comienzan a aplicar las reglas de P.

**P:** conjunto de reglas de producción:  $u::=v$  donde  $u \in \Sigma^+$  y  $v \in \Sigma^*$   
 $u = xAy$  tal que  $x, y \in \Sigma^*$  y  $A \in \Sigma_N$ .



## Definición

$\Sigma_T$ : Alfabeto de  
símbolos terminales

$\Sigma_N$ : Alfabeto de símbolos  
NO terminales

Axioma

$G = (\{0,1\}, \{N,C\}, N, P)$

$P = \{N ::= NC,$   
 $N ::= C,$   
 $C ::= 0,$   
 $C ::= 1\}$

Producciones



## Definición

Se cumple entre los alfabetos de  $G$ :

$\Sigma_T \cup \Sigma_N = \Sigma$  Alfabeto o vocabulario de  $G$

$\Sigma_T \cap \Sigma_N = \emptyset$  (son disjuntos)

**Notación de Backus:** Si  $u ::= v$  y  $u ::= w$  son dos reglas de producción de  $P$ , entonces se puede escribir  $u ::= v \mid w$

Se denomina notación **BNF**: *Forma Normal de Backus o Forma Normal de Backus-Naur*

Ejemplo:

sea  $G = (\{0,1,2,3,4,5,6,7,8,9\}, \{N,D\}, N,P)$

$P = \{ N ::= ND \mid D ; D ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \}$



## Forma sentencial

Sea una  $G$ , sea  $x \in \Sigma^*$ , es decir sea  $x \in (\Sigma_T \cup \Sigma_N)^*$ ,

$x$  se denomina **forma sentencial de  $G$**  si se verifica:

$$S^* \rightarrow x$$

es decir, que existe una **relación de Thue** entre el axioma y  $x$ .

Si  $x \in \Sigma_T^*$  se dice que  $x$  es una **Sentencia o instrucción del lenguaje descrito por  $G$**



## Lenguaje asociado a una gramática

Sea la gramática:

$$G_1 = (\Sigma_T, \Sigma_N, S, P)$$

Se llama:

lenguaje asociado a  $G_1$   
 lenguaje generado por  $G_1$   
 o lenguaje descrito por  $G_1$

al **conjunto de todas las sentencias (palabras) generadas** por  $G_1$ , es decir:

$$L(G_1) = \{x \mid S \xrightarrow{*} x, x \in \Sigma_T^*\}$$



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 41 ]

## Recursividad

Sea  $G$ ,

- ✓ Una  $G$  se llama **recursiva en  $U$** ,  $U \in \Sigma_{NT}$ , si se cumple:

$$U \rightarrow x U y$$

- Si  $x = \lambda$  ( $U \rightarrow U y$ ) se dice que  $G$  es **recursiva a izquierdas**
- Si  $y = \lambda$  ( $U \rightarrow x U$ ) se dice que  $G$  es **recursiva a derechas**

- ✓ Una regla de producción es recursiva si tiene la forma:

$$U ::= x U y$$

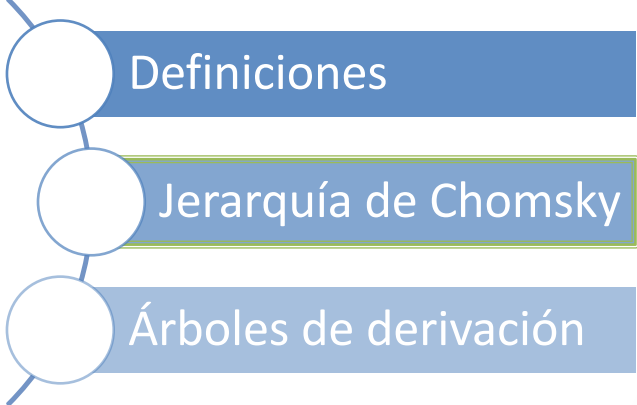
- ✓ Si un lenguaje es infinito, la gramática que lo representa tiene que ser recursiva



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 42 ]





Definiciones

**Jerarquía de Chomsky**

Árboles de derivación

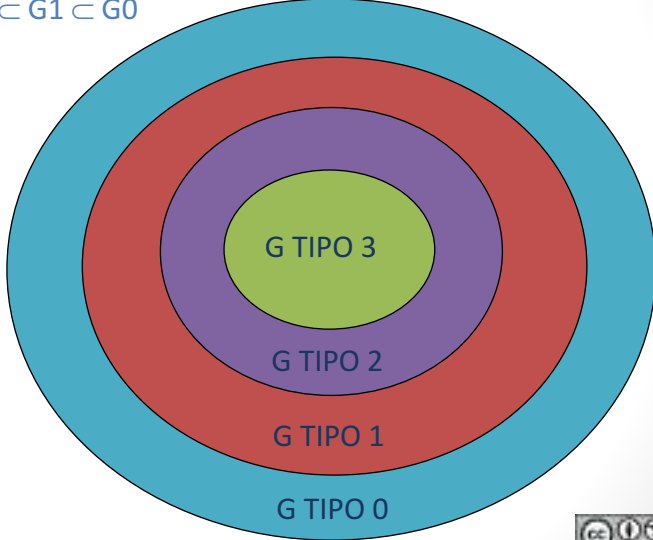
A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 43 ]

Universidad Carlos III de Madrid

## Jerarquía de Chomsky

$G_3 \subset G_2 \subset G_1 \subset G_0$



G TIPO 3

G TIPO 2

G TIPO 1

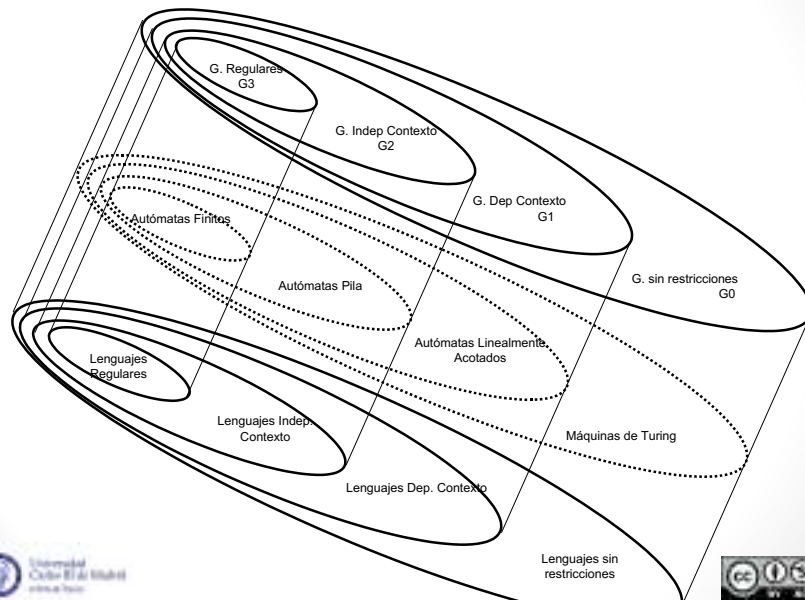
G TIPO 0

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 44 ]

Universidad Carlos III de Madrid

# Máquinas, Lenguajes y Algoritmos



[ 45 ]

## Tipo 0

$G = (\Sigma_T, \Sigma_N, S, P)$   
 $\Sigma_T \cup \Sigma_N = \Sigma$ , alfabeto gramática,  
 $\Sigma_T \cap \Sigma_N = \emptyset$

G0  
 No Restringidas  
 o con Estructura de Frase

$u ::= v$      $u \in \Sigma^+$   
                    $v \in \Sigma^*$

Única restricción:  $\lambda ::= v \notin P$

Forma sentencial:

$u = xAy$ ,  $x, y \in \Sigma^*$ ,  $A \in \Sigma_N$

Ejemplo:

$G = \{(0,1), (S,A), S, P\}$ , donde:  
 $P = \{S \rightarrow 0S0A, SA \rightarrow AS, 0A \rightarrow 1\}$

**Con estructura de frases:**

$(xAy ::= xvy) \in P$ , donde:

$x, y \in \Sigma^*$ ,  $A \in \Sigma_N$ ,  $v \in \Sigma^*$

En  $xAy ::= xvy$  cuando  $v = \lambda$ ,

$xAy ::= xy$  reglas compresoras

[ 46 ]

## Tipo 0

- ✓ Los lenguajes que son representados por G0 se llaman lenguajes sin restricciones
- ✓ Chomsky 1959. Todo  $L(G_0)$  puede ser descrito por una G0 con estructura de frases.

### Ejemplo

$$\begin{array}{lcl}
 G = (\{a,b\}, \{A,B,C\}, A, P) & & G' = \{a,b\}, \{A,B,C,X,Y\}, A, P' \\
 P = \{A ::= aABC / abC\} & & P' = \{A ::= aABC / abC\} \\
 CB ::= BC & \longrightarrow & \left\{ \begin{array}{l} CB ::= XB, XB ::= XY \\ XY ::= BY, BY ::= BC \end{array} \right. \\
 bB ::= bb & & bB ::= bb \\
 bC ::= b & & bC ::= b
 \end{array}$$



## Tipo 1

$$\begin{array}{l}
 G = (\Sigma_T, \Sigma_N, S, P) \\
 \Sigma_T \cup \Sigma_N = \Sigma, \text{ alfabeto gramática,} \\
 \Sigma_T \cap \Sigma_N = \emptyset
 \end{array}$$

**G1**  
Sensibles al Contexto

$$xAy ::= xvy$$

$x, y \in \Sigma^*, A \in \Sigma_N$   
 $v \in \Sigma^+ \rightarrow$  No permite reglas compresoras  
 Excepción:  $(S ::= \lambda) \in P$

Ejemplo:  
 $G = (\{a,b,c\}, \{A,B\}, A, P),$   
 $P = \{A ::= Aaa \mid a,$   
 $\quad aA ::= aB \mid a,$   
 $\quad aCCc ::= aAaCc,$   
 $\quad C ::= c\}$

Los Lenguajes representados por una gramática de Tipo 1 se llaman *lenguajes dependientes del contexto* o lenguajes sensibles al contexto (se puede cambiar A por v, siempre en el contexto x...y)



## Tipo 1

Los lenguajes que son representados por G1 se llaman **Lenguajes sensibles al contexto**

donde

$$\lambda \in L(G1) \text{ Sii } (S ::= \lambda) \in P$$

Ejemplo de G que no es G1

$$G = (\{a,b\}, \{S\}, S, P),$$

$$P = \{ S ::= aaaaSbbbb, \\ aSb ::= ab \}$$

Ejemplo de G que si es G1

$$G = (\{a,b\}, \{S\}, S, P),$$

$$P = \{ S ::= aaaaSbbbb, \\ aSb ::= abb \}$$



( 49 )

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

## Tipo 1. No decrecimiento

**Propiedad de NO DECRECIMIENTO de las G1**

Las cadenas que se obtienen en cualquier derivación de una G1 son de longitud no decreciente, es decir:

$$u \rightarrow v \Rightarrow |v| \geq |u|$$

La longitud de la parte derecha de la producción es mayor o igual a la longitud de la parte izquierda

**Demostración**

$$\alpha A \beta \rightarrow \alpha \gamma \beta$$

donde  $\gamma \in (\Sigma_T \cup \Sigma_{NT})^+$  por definición de G1 (no compresoras)

es decir,  $\gamma \neq \lambda$  siempre, lo que implica  $|\gamma| \geq 1$

y como  $|A| = 1$  como mínimo, queda demostrado



( 50 )

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

## Tipo 2

$G = (\Sigma_T, \Sigma_N, S, P)$   
 $\Sigma_T \cup \Sigma_N = \Sigma$ , alfabeto gramática,  
 $\Sigma_T \cap \Sigma_N = \emptyset$

**G2**  
**De Contexto Libre**

**$A ::= v$**

$v \in \Sigma^*$   
 $A \in \Sigma_N$

$r \in P$  se caracterizan por  
 tener **un sólo símbolo NT**  
 en su parte izquierda

$(S ::= \lambda) \in P$  y  $(A ::= \lambda) \notin P$   
 (algoritmo limpieza reglas NO generativas)

Ejemplo:

$G = (\{0,1,2,3,4,5,6,7,8,9\}, \{N,D\}, N, P)$

$P = \{N \rightarrow DN \mid D,$

$D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9\}$

**Contexto Libre:** Se puede cambiar  $A$  por  $v$ , en cualquier contexto



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

( 51 )

## Tipo 2

Los lenguajes que son representados por G2 se  
 llaman **Lenguajes no sensibles al contexto o de  
 contexto libre**

$\forall L(G2) \exists L(G2')$  sin reglas  $A ::= \lambda$  ( $A \neq S$ )

**Si  $\lambda \in L(G2) \rightarrow$  puede aplicarse el algoritmo de  
 eliminación reglas NO generativas.**



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

( 52 )

## Tipo 3

$G = (\Sigma_T, \Sigma_N, S, P)$   
 $\Sigma_T \cup \Sigma_N = \Sigma$ , alfabeto gramática,  
 $\Sigma_T \cap \Sigma_N = \emptyset$

G3  
 Gramáticas Regulares

**G3 Lineales por la Izda.**

$A ::= a$   
 $A ::= V a$   
 $S ::= \lambda$

**G3 Lineales por la Drcha.**

$A ::= a$   
 $A ::= a V$   
 $S ::= \lambda$

$a \in \Sigma_T$   
 $A, V \in \Sigma_N$   
 $S$  es axioma

$r \in P$  un sólo símbolo NT en su parte izda y su parte dcha comienza por un T seguido o no de NT (al revés en lineal derecha)

Ejemplo:

$G = \{(a,b), (S,A), S, P\}$ ,  
 $P = \{S \rightarrow aS, S \rightarrow aA, A \rightarrow bA, A \rightarrow b\}$



Universidad  
Carlos III de Madrid



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

( 53 )

## Tipo 3

Los lenguajes que son representados por G3 se llaman **lenguajes regulares**

$\forall L(G3) \exists L(G3')$  sin reglas  $A ::= \lambda$  ( $A \neq S$ )

Si  $\lambda \in L(G3)$ .

Ver algoritmo eliminación reglas NO generativas.



Universidad  
Carlos III de Madrid



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

( 54 )

# Jerarquía de Chomsky. Resumen

 $G_3 \subset G_2 \subset G_1 \subset G_0$ 

G0. No Restringidas o con Estructura de Frase

 $u ::= v$ 
 $u \in \Sigma^+$   
 $v \in \Sigma^*$ 

Única restricción:  $\lambda ::= v \notin P$

Con Estructura De Frases:

$(xAy ::= xvy) \in P$ , donde:  $x, y \in \Sigma^*$ ,  $A \in \Sigma_N$ ,  $v \in \Sigma^*$   
 Si  $v = \lambda \rightarrow xAy ::= xy$  reglas compresoras

G1. Sensibles al Contexto

 $xAy ::= xvy$ 
 $x, y \in \Sigma^*$ ,  $A \in \Sigma_N$ 

$v \in \Sigma^+ \rightarrow$  No permite reglas **compresoras**  
 Excepción:  $(S ::= \lambda) \in P$

G2. De Contexto Libre

 $A ::= v$ 
 $v \in \Sigma^*$   
 $A \in \Sigma_N$ 

$(S ::= \lambda) \in P$  y  $(A ::= \lambda) \notin P$

G3. Gramáticas Regulares

G3 Lineales por la Izda.

 $A ::= a$   
 $A ::= V a$   
 $S ::= \lambda$ 

G3 Lineales por la Drcha.

 $A ::= a$   
 $A ::= a V$   
 $S ::= \lambda$ 
 $a \in \Sigma_T$   
 $A, V \in \Sigma_N$ 


A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 55 ]

## Gramáticas equivalentes

Dos gramáticas son equivalentes si representan el mismo lenguaje.

Dada una gramática lineal por la derecha cualquiera, existe otra lineal por la izquierda equivalente y viceversa.



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 56 ]

# Gramáticas equivalentes

ALGORITMO: 3 PASOS.

PASO 1.

Construir una gramática equivalente que no sea recursiva en el axioma (axioma inducido):

1. se añade un nuevo símbolo en el alfabeto  $\Sigma_N$ , B
2.  $\forall S ::= x$ , donde  $x \in \Sigma^+$ , se añade una regla  $B ::= x$
3. Se transforman las reglas  $A ::= a S$  (que desaparecen) en reglas del tipo  $A ::= a B$ .
4. Las reglas tipo  $S ::= \lambda$  no se ven afectadas por este algoritmo.

Nota: las reglas  $S ::= x$ ,  $x \in \Sigma^+$ , no desaparecen

[ 57 ]



Universidad  
Carlos III de Madrid  
Ciencias Exactas



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

# Gramáticas equivalentes

ALGORITMO: 3 PASOS.

PASO 1. Quitar el axioma inducido:

Construir una gramática equivalente que no sea recursiva en el axioma:

$G1 = (\{a,b\}, \{S, A\}, S, P)$

$P = \{ \begin{array}{l} S ::= bA, \\ A ::= aS \mid a \end{array} \}$

PASO 1

$G2 = (\{a,b\}, \{S, A, B\}, S, P)$

$P = \{ \begin{array}{l} S ::= bA, \\ A ::= aB \mid a \\ B ::= bA \end{array} \}$

[ 58 ]



Universidad  
Carlos III de Madrid  
Ciencias Exactas



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso



# Gramáticas equivalentes

ALGORITMO: 3 PASOS.

PASO 2.

Construcción de un grafo dirigido a partir de la gramática.

a) número de nodos =  $C(\Sigma_N) + 1$ , cada nodo etiquetado con símbolos de  $\Sigma_N$  y otro con  $\lambda$

b) cada  $A ::= a B \in P \longrightarrow$  

c) cada  $A ::= a \in P \longrightarrow$  

d) si  $S ::= \lambda \in P \longrightarrow$  



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 59 ]

# Gramáticas equivalentes

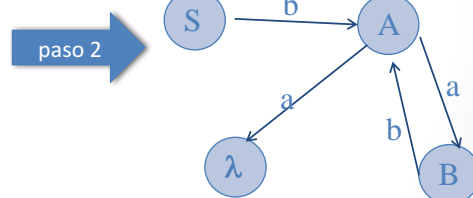
ALGORITMO: 3 PASOS.

PASO 2.

Construcción de un grafo dirigido a partir de la gramática.

$G_2 = (\{a,b\}, \{S, A, B\}, S, P)$

$P = \{ \begin{array}{l} S ::= bA, \\ A ::= aB \mid a \\ B ::= bA \end{array} \}$



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 60 ]

# Gramáticas equivalentes

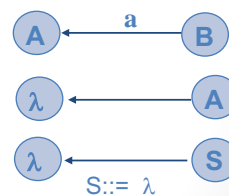
ALGORITMO: 3 PASOS.

PASO 3.

Se intercambian las etiquetas y se construye un nuevo grafo.

Transformación del grafo dirigido anterior:

- a) intercambiar etiquetas de S y  $\lambda$
  - b) invertir sentido de todos arcos
  - c) deshacer el camino y generar las nuevas reglas
- $\Rightarrow$  interpretar el grafo para obtener la G3LI equivalente



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 61 ]



Universidad  
Carlos III de Madrid

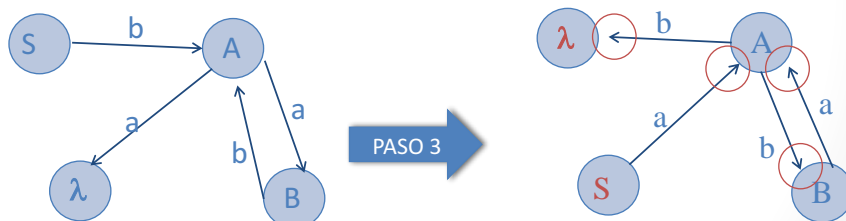


# Gramáticas equivalentes

ALGORITMO: 3 PASOS.

PASO 3.

Se intercambian las etiquetas y se construye un nuevo grafo.



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 62 ]



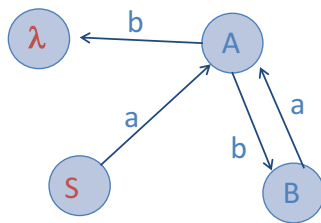
Universidad  
Carlos III de Madrid



# Gramáticas equivalentes

ALGORITMO: 3 PASOS.

A partir del grafo, se obtiene la gramática equivalente a izquierdas.



$$G3 = (\{a,b\}, \{S, A, B\}, S, P)$$

$$P = \{ \begin{array}{l} S ::= Aa, \\ A ::= Bb \mid b \\ B ::= Aa \end{array} \}$$


A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 63 ]

# Gramáticas equivalentes

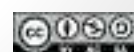
ALGORITMO: 3 PASOS.

Comprobemos que las gramáticas son equivalentes...

$$G1 = (\{a,b\}, \{S, A\}, S, P)$$

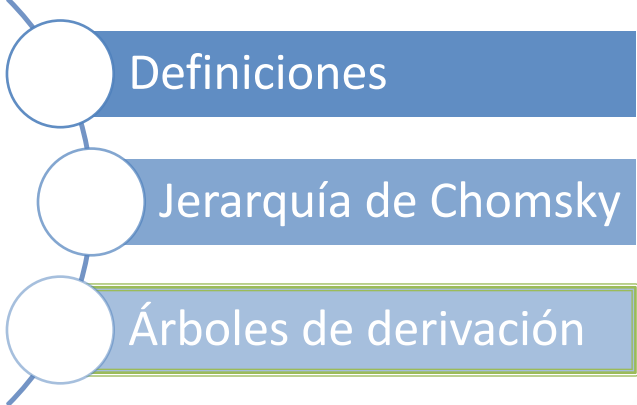
$$P = \{ \begin{array}{l} S ::= bA, \\ A ::= aS \mid a \end{array} \}$$

$$G3 = (\{a,b\}, \{S, A, B\}, S, P)$$

$$P = \{ \begin{array}{l} S ::= Aa, \\ A ::= Bb \mid b \\ B ::= Aa \end{array} \}$$


A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 64 ]



Definiciones

Jerarquía de Chomsky

Árboles de derivación

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 65 ]

Universidad  
Cádiz (UCA)  
Instituto de Investigación en  
Lenguaje y Computación (IILC)

## Árboles de derivación

- ✓ A las derivaciones de las *G* tipo 1, 2 y 3 les corresponde un árbol de derivación equivalente, también llamado  

**“árbol sintáctico” ó  
“parse tree”**
- ✓ Representa las producciones aplicadas durante la generación de **una sentencia**, es decir, su estructura de acuerdo con la *G*

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 66 ]

Universidad  
Cádiz (UCA)  
Instituto de Investigación en  
Lenguaje y Computación (IILC)

# Árboles de derivación

Es un árbol **ordenado** y **etiquetado** que se construye:

- ✓ La raíz se denota por el axioma de G (S habitualmente)
- ✓ Una **derivación directa** se representa por un conjunto de **ramas que salen de un nodo dado (parte izquierda de la P)**
- ✓ Aplicar una regla un símbolo de la parte izq. queda sustituido por una palabra  $u$  de la parte dcha. Por cada uno de los símbolos de  $u$  se dibuja una rama que sale del NT a ese símbolo:



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 67 ]

# Árboles de derivación

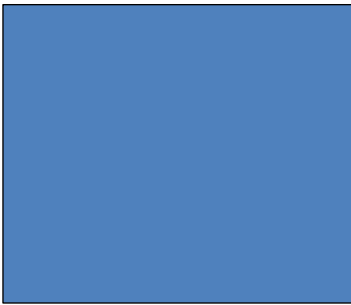
Table Text Size

Start Pause Step Noninverted Tree

Input 1.2  
String accepted: 29 nodes generated.

LHS	RHS
N	→ E
N	→ 0 D
N	→ E D
E	→ CE
E	→ C
D	→ E
C	→ 0
C	→ 1
C	→ 2
C	→ 3
C	→ 4
C	→ 5
C	→ 6
C	→ 7
C	→ 8
C	→ 9

Derived 2 from C. Derivations complete.




A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 68 ]

## Árboles de derivación

En una G1, además, se debe conservar el contexto. Para cada rama:

- ✓ el nodo de partida se llama **padre** del nodo final
- ✓ el nodo final es **hijo** del nodo padre
- ✓ dos nodos hijos del mismo padre se llaman **hermanos**
- ✓ un nodo es **ascendente** de otro si es su padre o ascendiente de su padre
- ✓ un nodo es **descendiente** de otro si es su hijo o descendiente de sus hijos



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 69 ]

## Árboles de derivación

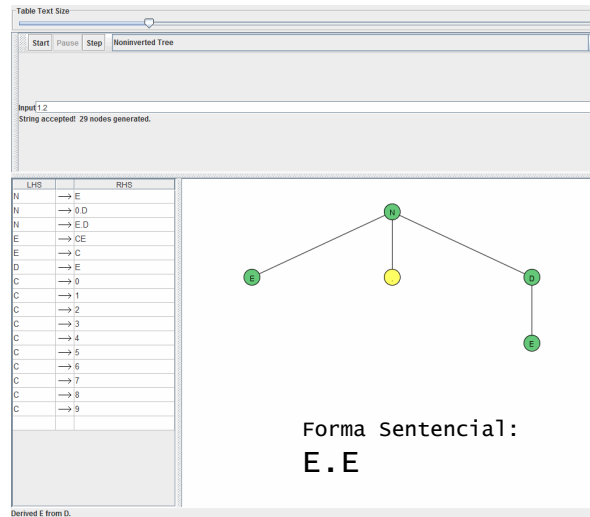
- ✓ A lo largo del proceso de construcción del árbol, los nodos finales de cada paso sucesivo, leídos de izqda. a dcha. dan la **forma sentencial** obtenida por la derivación representada por el árbol.
- ✓ El **conjunto de las hojas del árbol** (nodos denotados por símbolos terminales o  $\lambda$ ) leídos de izqda. a dcha. nos dan la **sentencia** generada por la derivación



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 70 ]

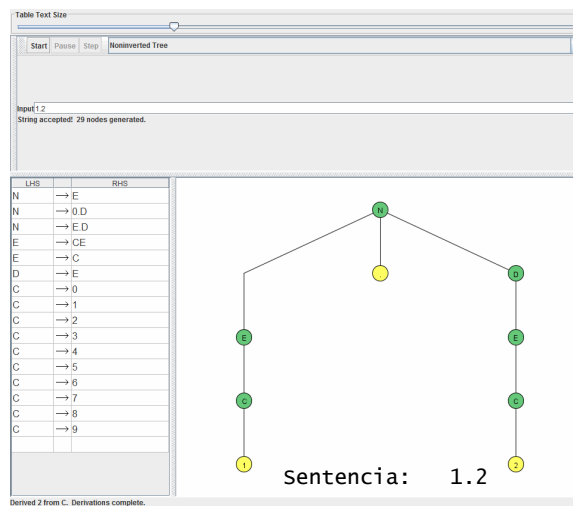
# Árboles de derivación



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 71 ]

# Árboles de derivación



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 72 ]

## Árboles de derivación

Dada la  $G = (\{a,b\}, \{A,S\}, S, \{S ::= aAS \mid a, A ::= SbA \mid SS \mid ba\})$ . Hallar un árbol de derivación para una sentencia de 6 letras.

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 73 ]



Universidad  
Carlos III de Madrid  
eduroad.es

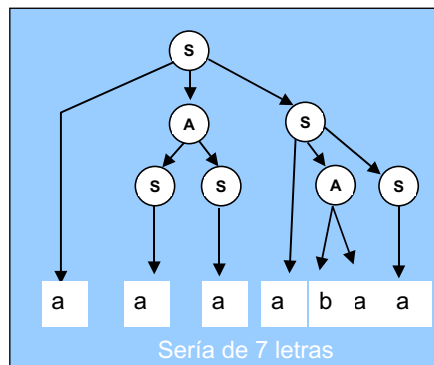
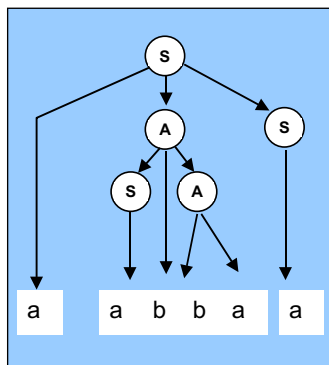


## Árboles de derivación

Dada la  $G = (\{a,b\}, \{A,S\}, S, \{S ::= aAS \mid a, A ::= SbA \mid SS \mid ba\})$ . Hallar un árbol de derivación para una sentencia de 6 letras.

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 74 ]



Sería de 7 letras



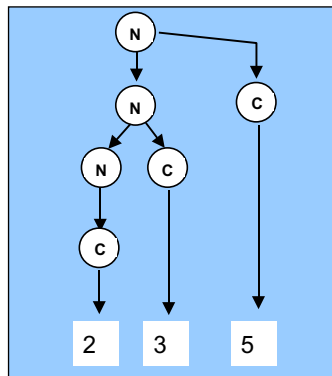
Universidad  
Carlos III de Madrid  
eduroad.es





## Árboles de derivación

Dada la  $G = (\{0,1,2,3,4,5,6,7,8,9\}, \{N,C\}, N, \{N ::= NC \mid C, C ::= 0|1|2|3|4|5|6|7|8|9\})$ . Hallar un árbol de derivación para  $N \rightarrow 235$

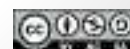
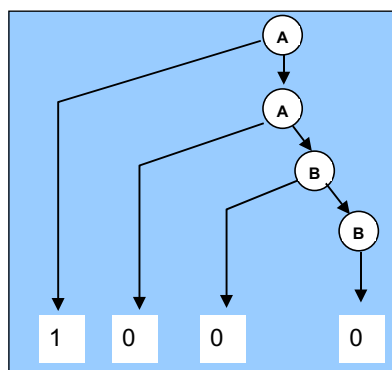


A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 75 ]

## Árboles de derivación

Dada la  $G = (\{0,1\}, \{A,B\}, A, \{A ::= 1A \mid 0B, B ::= 0B \mid 0\})$  una de cuyas derivaciones válidas es:  $A \rightarrow 1A \rightarrow 10B \rightarrow 100B \rightarrow 1000$ . Hallar un árbol de derivación para  $A \rightarrow 1000$



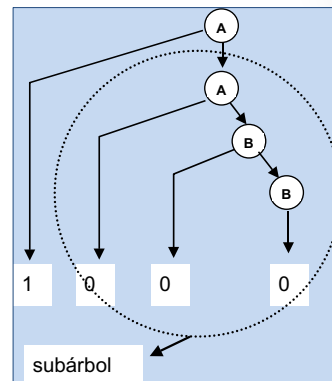
A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 76 ]

## Subárboles de derivación

Dado un árbol A correspondiente a una derivación, se llama **subárbol de A** al árbol

- ✓ cuya raíz es un nodo cualquiera de A,
- ✓ cuyos nodos son todos los descendientes de la raíz del subárbol en A
- ✓ y cuyas ramas son todas las que unen dichos nodos entre si en A.



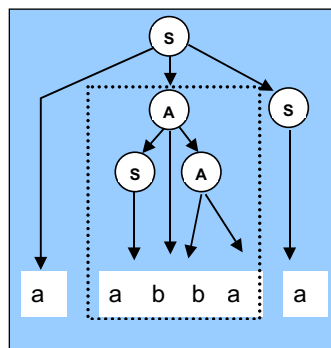
A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 77 ]

## Subárboles de derivación

### Teorema

Las hojas de un subárbol, leídas de izda. a dcha., forman una frase respecto al símbolo NT raíz del subárbol



**abba** es una frase de la forma sentencial **aabbba** respecto del símbolo **A**



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 78 ]

# Ambigüedad

- ✓ Concepto relacionado con el de árbol de derivación:
- ✓ Si una sentencia puede obtenerse en una  $G$  por medio de dos o más árboles de derivación diferentes, **la sentencia es ambigua**
- ✓ Una  $G$  es **ambigua** si contiene **al menos una sentencia ambigua**



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

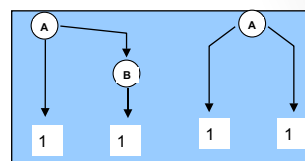
[ 79 ]

# Ambigüedad

Existen 3 niveles de ambigüedad:

- ✓ **Sentencia:** una sentencia es ambigua si puede obtenerse por medio de dos o más árboles de derivación diferentes

ej:  $G = (\{1\}, \{A, B\}, A, \{A ::= 1B \mid 11, B ::= 1\})$



- ✓ **Gramática:** es ambigua si contiene al menos una sentencia ambigua, ej: la  $G$  anterior
- ✓ **Lenguaje inherentemente ambiguo:** si todas las gramáticas que lo generan son ambiguas.



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 80 ]

# Ambigüedad

- ✓ Aunque una G sea ambigua, es posible que el lenguaje que describe no sea ambiguo [Floyd 1962]  $\Rightarrow$  es posible encontrar una G equivalente que no lo sea
- ✓ Existen lenguajes para los que NO es posible encontrar G no ambiguas  $\Rightarrow$  Lenguajes Inherentemente Ambiguos [Gross 1964]
- ✓ La propiedad de ambigüedad es indecidible. Tan solo es posible encontrar condiciones suficientes que aseguren que una G es no ambigua
- ✓ **Indecidible:** no existe un algoritmo que acepte una G y determine con certeza y en un tiempo finito si una G es ambigua o no.



# Ambigüedad

**Lenguajes Inherentemente Ambiguos:** para los que NO es posible encontrar G no ambiguas

## Ejemplo

$$L = \{a^n b^m c^m d^n\} \cup \{a^n b^n c^m d^m\} / m, n \geq 1$$

## Ejemplo

$L = \{11\}$  NO es inherentemente ambiguo

$G = (\{1\}, \{A, B\}, A, \{A ::= 1B / 11, B ::= 1\})$

$G' = (\{1\}, \{A\}, A, \{A ::= 11\}) \rightarrow$  Gramática NO ambigua

$$L(G) = L(G')$$



# Ambigüedad

Dada la  $G = (\{a,b\}, \{A,B,S\}, S, P)$

$P = \{$

$S ::= bA \mid aB$

$A ::= bAA \mid a \mid aS$

$B ::= b \mid BS \mid aBB\}$

demostrar que es una  $G$  ambigua al serlo la sentencia  
"aabbab"

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

( 83 )



Universidad  
Carlos III de Madrid  
cursos de posgrado



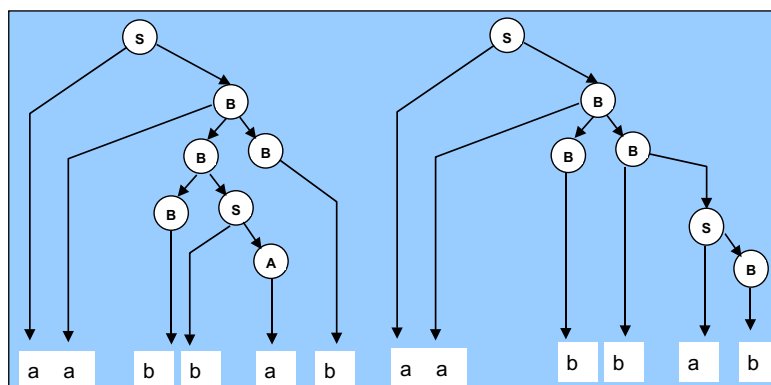
# Ambigüedad

Dada la  $G = (\{a,b\}, \{A,B,S\}, S, P)$

$P = \{$

$S ::= bA \mid aB, A ::= bAA \mid a \mid aS, B ::= b \mid BS \mid aBB\}$

demostrar que es una  $G$  ambigua al serlo la sentencia "aabbab"



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

( 84 )



Universidad  
Carlos III de Madrid  
cursos de posgrado



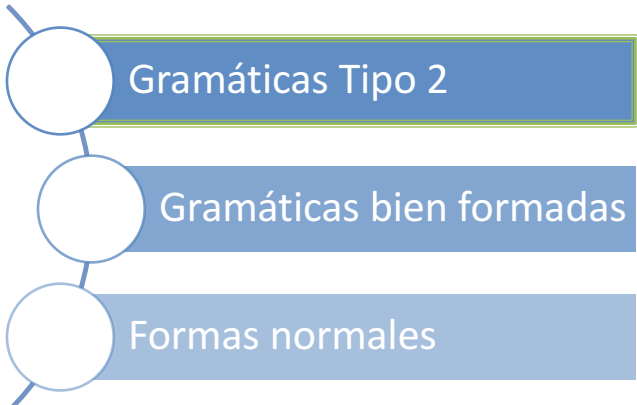
Segunda parte

# GRAMÁTICAS FORMALES

Universidad Carlos III de Madrid

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

( 85 )

Gramáticas Tipo 2


Gramáticas bien formadas

Formas normales

Universidad Carlos III de Madrid

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

( 86 )

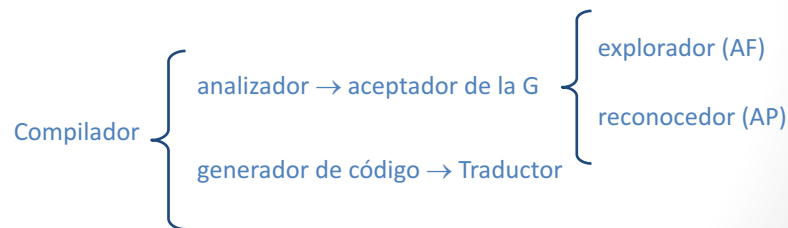


## Gramáticas Independientes del Contexto

Son las gramáticas de tipo 2 en la jerarquía de Chomsky.

### Relevancia

- ✓ son las empleadas en la Definición de Lenguajes de Programación y en la compilación de los mismos



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 87 ]

## Gramáticas Independientes del Contexto

A los lenguajes generados por gramáticas del tipo 2 de la jerarquía de Chomsky se les denomina **Lenguajes independientes del contexto o lenguajes de contexto libre**

- ✓ Se representan como  $L(G_2)$
- ✓ Existen algoritmos que permiten reconocer si un  $L(G_2)$  es vacío, finito o infinito



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 88 ]

## Gramáticas Independientes del Contexto

### 1. Dada una $G$ , el Lenguaje que genera, ¿es vacío o no?:

Sea  $G_2$ ,  $m = C(\Sigma_{NT})$ ,  $L(G_2) \neq \emptyset$  si  $\exists x \in L(G_2)$  tal que  $x$  puede generarse con un árbol de derivación en el que todos los caminos tienen longitud  $\leq m$

Se generan todos los árboles de derivación con caminos  $\leq m = C(\Sigma_{NT})$  mediante el algoritmo:

- conjunto de árboles con longitud 0 (un árbol con  $S$  como raíz y sin ramas)
- a partir del conjunto de árboles de longitud  $n$ , generamos el conjunto de longitud  $n+1 < m+1$  aplicando al conjunto de partida una producción que no haga duplicarse algún NT en el camino considerado
- se aplica el paso b) recursivamente hasta que no puedan generarse más árboles con caminos de longitud  $\leq m$ . Al ser  $m$  y el número de reglas de  $P$  finito  $\Rightarrow$  el algoritmo termina

$L(G_2) = \emptyset$  si ninguno de los árboles genera una sentencia



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 89 ]

## Gramáticas Independientes del Contexto

$m = C(\Sigma_{NT}) = 4$

Ejemplo de  $L(G_2) = \emptyset$

Sea  $G = (\{a,b\}, \{A,B,C,S\}, S, P)$

$P = \{$

$S ::= aB \mid aA$

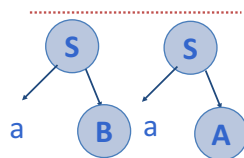
$A ::= B \mid abB$

$B ::= bC\}$

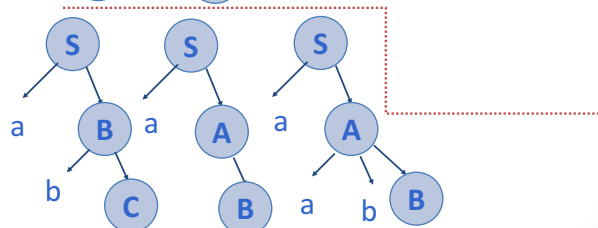
1.  $m=0$



2.  $m=1$



3.  $m=2$



4.  $m=3$

No se generan sentencias y salen NT ya obtenidos  
Lenguaje vacío



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 90 ]



## Gramáticas Independientes del Contexto

### 2. Si $L(G_2)$ es no vacío, comprobar si $L(G_2) = \infty$

- ✓ Se construye un grafo cuyos nodos están etiquetados con los símbolos de  $(\Sigma_{NT})$  mediante el algoritmo:
  - si  $\exists$  una producción  $A ::= \alpha B \beta$ , se crea un arco de A a B donde  $A, B \in \Sigma_{NT}$  y  $\alpha, \beta \in \Sigma^*$
  - si no existen ciclos en el grafo el  $L(G_2) = \text{finito}$
  - $L(G_2) = \infty$  si existen ciclos accesibles desde el axioma que corresponden a derivaciones de la forma  $A \rightarrow^+ \alpha A \beta$ , donde  $|\alpha| + |\beta| > 0$  (que no sean  $\lambda$  las dos a la vez).

**$L(G_2) \neq \infty$  si no hay ciclos en el grafo**



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

( 91 )

## Gramáticas Independientes del Contexto

### Ejemplo de $L(G_2) = \infty$

Sea  $G = (\{a,b,c\}, \{A,B,C,S\}, S, P)$

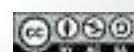
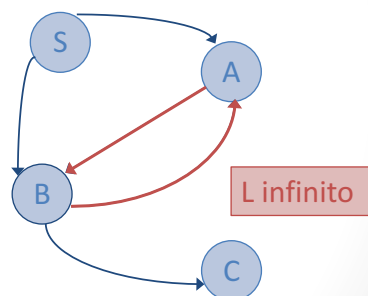
$P = \{$

$S ::= aB / aA$

$A ::= abB$

$B ::= bC / aA$

$C ::= c \}$



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

( 92 )

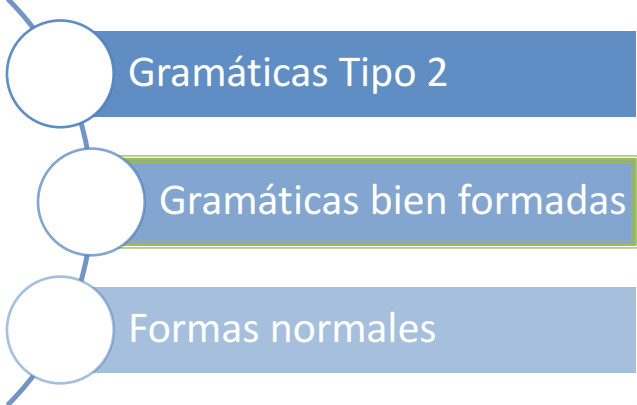


Diagram illustrating the hierarchy of grammar types:

- Gramáticas Tipo 2
- Gramáticas bien formadas** (highlighted)
- Formas normales

Universidad Carlos III de Madrid

93

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

## Gramáticas Bien Formadas

Transformación de una  $G$  dada en otra equivalente cuyas reglas de producción estén en un formato carente de imperfecciones:

- Limpieza de Gramáticas**
  - Reglas Innecesarias
  - Símbolos Inaccesibles
  - Reglas Supérfluas
- Eliminación de símbolos no generativos**
- Eliminación de reglas no generativas**
- Eliminación de reglas de red denominación**

Universidad Carlos III de Madrid

94

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

# Gramáticas Bien Formadas

Transformación de una G dada en otra equivalente cuyas reglas de producción estén en un formato carente de imperfecciones:

## 1. Limpieza de Gramáticas

- a. Reglas Innecesarias
- b. Símbolos Inaccesibles
- c. Reglas Superfluas

Gramática  
Reducida

Gramática  
Limpia

Gramática Bien Formada

## 2. Eliminación de símbolos no generativos

## 3. Eliminación de reglas no generativas

## 4. Eliminación de reglas de red denominación



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

( 95 )

# Gramáticas Bien Formadas

## 1. Limpieza de Gramáticas

- a. **Reglas Innecesarias:** las reglas  $A ::= A \in P$  son innecesarias y hacen que G sea ambigua  $\Rightarrow$  eliminarlas



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

( 96 )

# Gramáticas Bien Formadas

## 1. Limpieza de Gramáticas

- b. **Símbolos Inaccesibles:** sea  $U ::= x \in P$ , donde  $U \in \Sigma_N \neq S$  y no aparece en la parte derecha de ninguna otra regla de producción, se dice que  $U$  es inaccesible.

Todo símbolo  $U \in \Sigma_N$  no inaccesible debe cumplir  $S^* \rightarrow xUy$ .

**Eliminación de símbolos inaccesibles:**

1. Hacer una lista con todos los símbolos de la gramática (T y NT)
2. Marcar el axioma de la gramática.
3. Dado  $xUy ::= xuy \rightarrow$  Marcar todos los símbolos que aparecen en la cadena  $u$  de la parte derecha.
4. Si en el paso anterior se ha marcado algún símbolo, se repite de nuevo dicho paso teniendo en cuenta los símbolos marcados. En caso contrario, fin del algoritmo.



# Gramáticas Bien Formadas

## 1. Limpieza de Gramáticas

- b. **Símbolos Inaccesibles**

### Ejemplo

sea la  $G = (\{a, b, c\}, \{S, A, B, C\}, S, P)$ ,

donde  $P = \{S ::= aA$

$A ::= Bc$

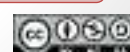
$B ::= bA$

$C ::= c\}$

1.  $\{a, b, c, S, A, B, C\}$
2.  $\{a, b, c, S, A, B, C\}$
3.  $\{a, b, c, S, A, B, C\}$
4.  $\{a, b, c, S, A, B, C\}$
5.  $\{a, b, c, S, A, B, C\}$
6.  $\{a, b, c, S, A, B, C\}$

Sin cambios.  
Fin algoritmo

**C es un símbolo INACCESIBLE!!**



# Gramáticas Bien Formadas

## 1. Limpieza de Gramáticas

- c. **Reglas Superfluas:** son aquellas que no contribuyen a la formación de palabras  $x \in \Sigma_T^*$ .

Estas reglas contiene algún **símbolo No Terminal no generativo**.

Todo símbolo no superfluo debe cumplir  $U \rightarrow t$ , tal que  $t \in \Sigma_T^*$

**Algoritmo** (es un algoritmo recursivo de marcado)

- marcar los NT para los que existe una regla  $U ::= x$  donde  $x \in \Sigma_T^*$  (es una cadena de T o  $\lambda$ , o en pasadas sucesivas contiene NT marcados)
- si todos los NT están marcados  $\Rightarrow$  no existen símbolos superfluos y fin
- si la última vez que se pasó por el paso a) se marcó un NT, volver al paso a).
- todo  $A \in \Sigma_{NT}$  no marcado es superfluo.



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

( 99 )

# Gramáticas Bien Formadas

## 1. Limpieza de Gramáticas

- c. **Reglas Superfluas:**

ejemplo: sea  $G = (\{e, f\}, \{S, A, B, C, D\}, S, P)$

donde  $P = \{$

$S ::= Be$

$A ::= Ae \mid e$

$B ::= Ce \mid Af$

$C ::= Cf$

$D ::= f\}$

1ª pasada:

$D ::= f$  y  $A ::= e$

2ª pasada:

$B ::= Af$

3ª pasada:

$S ::= Be$

Es una cadena de Terminales

Los NoTerminales están marcados.



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

( 100 )

# Gramáticas Bien Formadas

## 1. Limpieza de Gramáticas

### c. Reglas Supérfluas:

ejemplo: sea  $G = (\{e, f\}, \{S, A, B, \cancel{C}, D\}, S, P)$

donde  $P = \{$

$S ::= Be$

$A ::= Ae \mid e$

$B ::= \cancel{C}e \mid Af$

$C ::= \cancel{C}f$

$D ::= f\}$

1ª pasada:

$D ::= f$  y  $A ::= e$

2ª pasada:

$B ::= Af$

3ª pasada:

$S ::= Be$

Sin marcar: C, que se elimina, así como sus reglas

(101)

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

# Gramáticas Bien Formadas

Transformación de una  $G$  dada en otra equivalente cuyas reglas de producción estén en un formato carente de imperfecciones:

## 1. Limpieza de Gramáticas

- Reglas Innecesarias
- Símbolos Inaccesibles
- Reglas Superfluas

Gramática Reducida

Gramática Limpia

Gramática Bien Formada

- Eliminación de símbolos no generativos
- Eliminación de reglas no generativas
- Eliminación de reglas de red denominación

(102)

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

# Gramáticas Bien Formadas

## 2. Eliminación de símbolos no generativos:

Sea  $G_2 = (\Sigma_T, \Sigma_N, S, P)$ ,  $\forall A \in \Sigma_N$  construiremos la gramática  $G(A)$ , donde  $A$  es el axioma. Si  $L(G(A)) = \emptyset \Rightarrow A$  es símbolo **no generativo** y se puede eliminar, así como todas las reglas que lo contengan, obteniéndose otra  $G_2$  equivalente.

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

(103)



# Gramáticas Bien Formadas

## 3. Eliminación de reglas no generativas: son $A ::= \lambda$ ( $A \neq S$ )

### Algoritmo:

1. Eliminar de la Gramática las regla de la forma  $U ::= \lambda$
2. Por cada regla de la gramática donde  $U$  aparezca en la parte dcha.,  $V ::= xUy$ , se añade la regla  $V ::= xy$  (a menos que ya existe).
3. Repetir 2. hasta que no quede ninguna regla de la forma  $U ::= \lambda$ , o que sólo quede  $S ::= \lambda$ .

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

(104)



## Gramáticas Bien Formadas

### 3. Eliminación de reglas no generativas: son $A ::= \lambda$ ( $A \neq S$ )

Ejemplo:

$G = (\{a, b\}, \{S, P, Q\}, S, Pr)$

$Pr = \{ S ::= PQ \mid aSb \mid P$

$P ::= aPQ \mid a$

$Q ::= Qb \mid \lambda \}$

Regla NO generativa:  
 $Q ::= \lambda$

ELIMINARLA



[ 105 ]

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

## Gramáticas Bien Formadas

### 3. Eliminación de reglas no generativas: son $A ::= \lambda$ ( $A \neq S$ )

Ejemplo:

$G = (\{a, b\}, \{S, P, Q\}, S, Pr)$

$Pr = \{ S ::= PQ \mid aSb \mid P$

$P ::= aPQ \mid a$

$Q ::= Qb \mid \lambda \}$

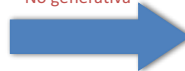
$G' = (\{a, b\}, \{S, P, Q\}, S, Pr)$

$Pr = \{ S ::= PQ \mid aSb \mid P$

$P ::= aPQ \mid a \mid aP$

$Q ::= Qb \mid b \}$

Eliminar la Regla  
No generativa



[ 106 ]

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso



# Gramáticas Bien Formadas

Transformación de una G dada en otra equivalente cuyas reglas de producción estén en un formato carente de imperfecciones:

## 1. Limpieza de Gramáticas

- a. Reglas Innecesarias
- b. Símbolos Inaccesibles
- c. Reglas Supérfluas

## 2. Eliminación de símbolos no generativos

## 3. Eliminación de reglas no generativas

## 4. Eliminación de reglas de red denominación

Gramática  
Reducida

Gramática  
Limpia

Gramática Bien Formada



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

(107)

# Gramáticas Bien Formadas

4. Eliminación de reglas de red denominación: son reglas del tipo  $A ::= B$   
(donde  $A \neq B$ )

## Algoritmo

1. Eliminar de la Gramática las regla de la forma  $U ::= V$
2. Por cada regla de la forma  $V ::= x$ , añadir  $U ::= x$  (si No Existe).
3. Repetir 2. hasta que no quede ninguna regla de red denominación.



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

(108)

## Gramáticas Bien Formadas

4. **Eliminación de reglas de red denominación:** son reglas del tipo  $A ::= B$  (donde  $A \neq B$ )

Ejemplo:

$G = (\{a, b\}, \{S, P, Q\}, S, Pr)$

$Pr = \{ S ::= PQ \mid aSb \mid P$

$P ::= aPQ \mid aP \mid a$

$Q ::= Qb \mid b \}$



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[109]



Universidad  
Carlos III de Madrid  
c3m.es



## Gramáticas Bien Formadas

4. **Eliminación de reglas de red denominación:** son reglas del tipo  $A ::= B$  (donde  $A \neq B$ )

Ejemplo:

$G = (\{a, b\}, \{S, P, Q\}, S, Pr)$

$Pr = \{ S ::= PQ \mid aSb \mid P$

$P ::= aPQ \mid aP \mid a$

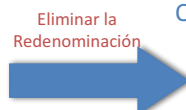
$Q ::= Qb \mid b \}$

$G' = (\{a, b\}, \{S, P, Q\}, S, Pr)$

$Pr = \{ S ::= PQ \mid aSb \mid aPQ \mid aP \mid a$

$P ::= aPQ \mid aP \mid a$

$Q ::= Qb \mid b \}$



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[110]



Universidad  
Carlos III de Madrid  
c3m.es



# Gramáticas Bien Formadas

Transformación de una G dada en otra equivalente cuyas reglas de producción estén en un formato carente de imperfecciones:

## 1. Limpieza de Gramáticas

- a. Reglas Innecesarias
- b. Símbolos Inaccesibles
- c. Reglas Supérfluas

Gramática  
Reducida

Gramática  
Limpia

Gramática Bien Formada

## 2. Eliminación de símbolos no generativos

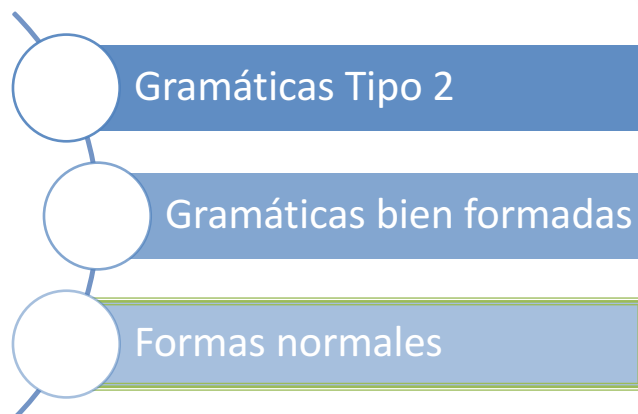
## 3. Eliminación de reglas no generativas

## 4. Eliminación de reglas de red denominación



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

(111)



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

(112)

# Formas Normales

- ✓ Son notaciones que se aplican a las G2:  
Afectan a la forma de las reglas de producción
- ✓ Son dos las que se va a estudiar:  
***Forma Normal de Chomsky***  
***Forma Normal de Greibach***

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

(113)



Universitat  
de València



## Forma Normal de Chomsky

Una gramática bien formada está en Forma Normal de Chomsky (FNC) si las partes derechas de todas sus reglas de producción tienen a lo sumo dos símbolos, y cuando son dos, ambos son No Terminales.

**Ejemplo:** La siguiente gramática está en FNC:

$S ::= \lambda \mid a \mid NN$

$N ::= NT \mid AB$

$A ::= a$

$B ::= b$

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

(114)



Universitat  
de València



# Forma Normal de Chomsky

A partir de cualquier gramática de Tipo 2, se puede construir otra equivalente que está en FNC.

*Para ello, se deben sustituir del siguiente modo las reglas cuya parte derecha tiene más de 2 símbolos por varias reglas que tengan en la parte derecha dos símbolos no terminales o un solo símbolo terminal:*

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

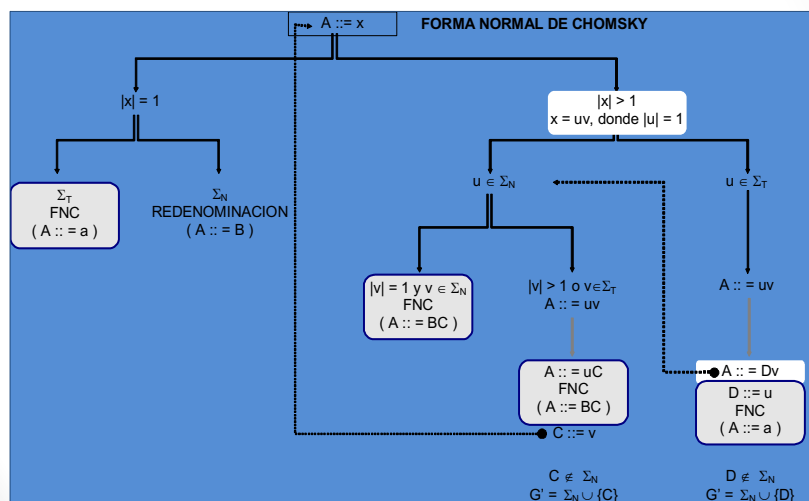
(115)



Universidad  
Carlos III de Madrid  
c3m.es



# Forma Normal de Chomsky



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

(116)



Universidad  
Carlos III de Madrid  
c3m.es



# Forma Normal de Chomsky

## EJEMPLO:

$G = (\{a, b, c\}, \{A, B, C, D, E, F\}, A, P)$   
 $P =$   
 $B ::= B$   
 $C ::= C$   
 $A ::= ABC$   
 $D ::= a$   
 $E ::= FbB$   
 $F ::= EaA$   
 $A ::= bA$   
 $B ::= Ba$   
 $A ::= aBb$   
 $B ::= bAa$   
 $A ::= a$   
 $B ::= a$   
 $C ::= c$   
 $C ::= Cb$



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

(117)

# Forma Normal de Chomsky

## Bien Formar la Gramática

$G = (\{a, b, c\}, \{A, B, C, D, E, F\}, A, P)$   
 $P =$   
 $B ::= B$  ← Innecesaria  
 $C ::= C$  ← Innecesaria  
 $A ::= ABC$   
 $D ::= a$  ← Innecesaria  
 $E ::= FbB$  ← Innecesaria  
 $F ::= EaA$  ← Innecesaria  
 $A ::= bA$   
 $B ::= Ba$   
 $A ::= aBb$   
 $B ::= bAa$   
 $A ::= a$   
 $B ::= a$   
 $C ::= c$   
 $C ::= Cb$

### Gramática Bien Formada:

$G = (\{a, b, c\}, \{A, B, C\}, A, P)$   
 $P =$   
 $A ::= a$   
 $A ::= aBb$   
 $A ::= ABC$   
 $A ::= bA$   
 $B ::= a$   
 $B ::= Ba$   
 $B ::= bAa$   
 $C ::= c$   
 $C ::= Cb$



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

(118)

## Forma Normal de Chomsky

Gramática Bien Formada:

$G = (\{a, b, c\}, \{A, B, C, D, E, F\}, A, P)$

$P =$

$A::=a$	Sí (FNC)
$A::=aBb$	No (FNC)
$A::=ABC$	No
$A::=bA$	No
$B::=a$	Sí
$B::=Ba$	No
$B::=bAa$	No
$C::=c$	Sí
$C::=Cb$	No

Transformación a FNC (quedan tabuladas las nuevas reglas generadas en FNC)

Tratamiento de  $A::=aBb$

$A::=aBb$

$D::=a$

$A::=DBb$

$E::=Bb$

$A::=DE$

$F::=b$

$E::=BF$

Tratamiento de  $A::=ABC$

$A::=ABC$

$G::=BC$

$A::=AG$

Tratamiento de  $A::=bA$

$A::=bA$

$A::=FA$

Tratamiento de  $B::=Ba$

$B::=Ba$

$B::=BD$

Tratamiento de  $B::=bAa$

$B::=bAa$

$B::=FAa$

$H::=Aa$

$B::=FH$

$H::=AD$

Tratamiento de  $C::=Cb$

$C::=Cb$

$C::=CF$



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[119]

## Forma Normal de Greibach

✓ **FNG** es una notación muy interesante para algunos reconocimientos sintácticos. En ella todas las reglas tienen la parte derecha comenzando con un terminal seguido opcionalmente de uno o varios NT

1. TEOREMA: todo **L de contexto libre sin  $\lambda$**  puede ser generado por una G2 en la que todas las reglas sean de la forma:

- $A \rightarrow a\alpha$  donde  $A \in \Sigma_{NT}$ ,  $a \in \Sigma_T$  y  $\alpha \in \Sigma_{NT}^*$
- Si  $\lambda \in L$  habrá que añadir  $S::=\lambda$

2. TEOREMA: toda G2 puede reducirse a otra G2 equivalente sin reglas recursivas a izquierdas



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[120]

## Forma Normal de Greibach

**FNG:** para transformar una G2 en su equivalente en forma normal de Greibach:

1. Limpiar y formar bien. Eliminar la recursividad a izquierdas
2. Aplicar el algoritmo de transformación a FNG, verificando en cada paso que no aparezcan nuevas reglas recursivas a izquierdas y si aparecen, eliminándolas con el paso 1

**EJEMPLO:**

$G = (\{a,b\}, \{S\}, S, P)$ , donde  $P = \{S ::= aSb \mid SS \mid \lambda\}$

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 121 ]



Universidad  
Carlos III de Madrid  
c3m.es



## Forma Normal de Greibach

1. Eliminar la recursividad a izquierdas, resumiendo, sería:

sea  $G = (\{\alpha_1, \alpha_2, \beta_1, \beta_2\}, \{A\}, A, P)$ ,

donde  $P = \{A ::= A\alpha_1 \mid A\alpha_2 \mid \beta_1 \mid \beta_2\}$

Quedaría:

$$A ::= \beta_1 \mid \beta_2 \mid \beta_1 X \mid \beta_2 X$$

$$X ::= \alpha_1 \mid \alpha_2 \mid \alpha_1 X \mid \alpha_2 X$$

**Eliminar la recursividad a izquierdas:**

$S ::= aSb \mid SS \mid \lambda \rightarrow$  (se transforma en)  $\rightarrow S ::= aSb \mid aSbX \mid \lambda$  y  $X ::= SX \mid S$

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 122 ]



Universidad  
Carlos III de Madrid  
c3m.es





# Forma Normal de Greibach

2. Transformación de G2 bien formada sin RI a FNG:

## 2.1 Establecer una relación de orden parcial en $\Sigma_{NT}$

$\Sigma_{NT} = \{A_1, A_2, \dots, A_n\}$  basándose en: si  $A_i \rightarrow A_j \alpha$ ,  $A_i$  precederá a  $A_j$ .

Cuando hay reglas "contradictorias" usar una de ellas para el orden y mirar el resto para ver que conviene más

$A ::= \alpha B \beta$  (A sería nº 1 y B nº 2).  $\alpha$  es una cadena de 0 o más terminales y  $\beta$  una cadena de 0 o más símbolos.

$B ::= \delta C \gamma$  (B sería nº 2 y C nº 3).  $\delta$  es una cadena de 0 o más terminales y  $\gamma$  una cadena de 0 o más símbolos.



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 123 ]

# Forma Normal de Greibach

2. Transformación de G2 bien formada sin RI a FNG:

## 2.2 Se clasifican las reglas en 3 grupos:

**Grupo 1:**  $A_i \rightarrow a \alpha$ , donde  $a \in \Sigma_T$  y  $\alpha \in \Sigma^*$

**Grupo 2:**  $A_i \rightarrow A_j \alpha$  donde  $A_i$  precede a  $A_j$  en el conjunto  $\Sigma_{NT}$  ordenado

**Grupo 3:**  $A_k \rightarrow A_i \alpha$  donde  $A_i$  precede a  $A_k$  en el conjunto  $\Sigma_{NT}$  ordenado

Hacer lo mismo con las de grupo 2

Ordenar los no terminales:  $\{X, S\}$  y clasificar las reglas según este orden:

$S ::= aSb$  (G1, aunque hay que quitar el terminal que está detrás del no terminal)

$S ::= aSbX$  (G1, aunque hay que quitar el terminal que está detrás del no terminal)

$S ::= \lambda$  (G1)

$X ::= SX$  (G2)

$X ::= S$  (G2)



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[ 124 ]

# Forma Normal de Greibach

2. Transformación de G2 bien formada sin RI a FNG:

## 2.3 Se transforman las reglas de grupo 3 → grupo 2 → grupo 1: FNG

$A_k \rightarrow A_i \alpha$  se sustituye  $A_i$  por la parte dcha. de todas las reglas que tienen  $A_i$  como parte izda.

Hacer lo mismo con las de grupo 2

Ordenar los no terminales:  $\{X, S\}$  y clasificar las reglas según este orden:

$S ::= aSb$  (G1, aunque hay que quitar el terminal que está detrás del no terminal)

$S ::= aSbX$  (G1, aunque hay que quitar el terminal que está detrás del no terminal)

$S ::= \lambda$  (G1)

$X ::= SX$  (G2)

$X ::= S$  (G2)



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

(125)

# Forma Normal de Greibach

2. Transformación de G2 bien formada sin RI a FNG:

## 2.3 Se transforman las reglas de grupo 3 → grupo 2 → grupo 1: FNG

$A_k \rightarrow A_i \alpha$  se sustituye  $A_i$  por la parte dcha. de todas las reglas que tienen  $A_i$  como parte izda.

Es decir, Sustituir el primer símbolo NT de la parte dcha. de cada regla del grupo 3 por las partes dchas. de todas las reglas (en cualquier grupo) donde dicho primer símbolo aparezca como parte izquierda. Hacer esto hasta que hayamos conseguido que todas las reglas del grupo 3 hayan sido transformadas en reglas de otros grupos. Si en este proceso aparecen reglas recursivas a izquierdas, transformarlas.

Hacer lo mismo con las de grupo 2



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

(126)

# Forma Normal de Greibach

2. Transformación de G2 bien formada sin RI a FNG:

## 2.3 Se transforman las reglas de grupo 3 $\rightarrow$ grupo 2 $\rightarrow$ grupo 1: FNG

$A_k \rightarrow A_i \alpha$  se sustituye  $A_i$  por la parte dcha. de todas las reglas que tienen  $A_i$  como parte izda.

Hacer lo mismo con las de grupo 2

Transformamos las de grupo G2 en grupo G1:

$X ::= SX \rightarrow S ::= aSb \rightarrow X ::= aSbX$  (G1, aunque hay que quitar el terminal que está detrás del no terminal)

$\rightarrow S ::= aSbX \rightarrow X ::= aSbXX$  (G1, aunque hay que quitar el terminal que está detrás del no terminal)

$X ::= S \rightarrow S ::= aSb \rightarrow X ::= aSb$  (G1, aunque hay que quitar el terminal que está detrás del no terminal)

$\rightarrow S ::= aSbX \rightarrow X ::= aSbX$  (G1, aunque hay que quitar el terminal que está detrás del no terminal)



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[127]

# Forma Normal de Greibach

2. Transformación de G2 bien formada sin RI a FNG:

2. 4. Cuando todas las reglas son de grupo 1, la G está en FNG a falta de eliminar los símbolos terminales no situados en la cabecera de la parte derecha.

$A \rightarrow \alpha a \beta$  donde  $a \in \Sigma_T$  y  $\alpha \neq \lambda$

$A \rightarrow \alpha B \beta$   
 $B \rightarrow a$

Es decir, transformar los símbolos terminales no situados en cabeza de parte derecha, asignándoles nuevos NT que deriven en ellos (o NT que ya existen que sólo deriven en ellos).



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

[128]

## Forma Normal de Greibach

2. Transformación de G2 bien formada sin RI a FNG:

2. 4. Cuando todas las reglas son de grupo 1, la G está en FNG a falta de eliminar los símbolos terminales no situados en la cabecera de la parte derecha.

$A \rightarrow \alpha a \beta$  donde  $a \in \Sigma_T$  y  $\alpha \neq \lambda$

$A \rightarrow \alpha B \beta$   
 $B \rightarrow a$

Introducimos un nuevo símbolo para quitar la a:

$B ::= a$



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

(129)

## Forma Normal de Greibach

### EJEMPLO

$G = (\{a,b\}, \{S\}, S, P)$ , La Gramática en FNG queda:

$P = \{S ::= aSb \mid SS \mid \lambda\}$   $G' = (\{a,b\}, \{S,X,B\}S, P')$

$P' = \{$   
 $S ::= aSB \mid aSBX \mid \lambda$   
 $X ::= aSBX \mid aSBXX \mid aSB$   
 $B ::= b$   
 $\}$



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

(130)

# Bibliografía

- Libro Básico 1 Bibliografía (AAM). Enrique Alfonseca Cubero, Manuel Alfonseca Cubero, Roberto Moriyón Salomón. Teoría de autómatas y lenguajes formales. McGraw-Hill (2007). Capítulo 5
- Libro Básico 2 Bibliografía (HMU). John E. Hopcroft, Rajeev Motwani, Jeffrey D.Ullman. Introducción a la teoría de autómatas, lenguajes y computación (3ª edición). Ed, Pearson Addison Wesley.
- Libro Básico 4 Bibliografía (AAM). Manuel Alfonseca, Justo Sancho, Miguel Martínez Orga. Teoría de lenguajes, gramáticas y autómatas. Publicaciones R.A.E.C. 1997 Capítulo 3



## **Parte V**

### **TEMA 5. Expresiones Regulares**



# 5. Lenguajes Regulares

Grado Ingeniería Informática  
Teoría de Automatas y Lenguajes Formales



AUTÓMATAS FINITOS Y G3



[ 2 ]



## Gramática asociada a un AF

Sea el AF,  $A = (\Sigma, Q, q_0, f, F)$ , existe una G3 LD tal que

$$L(G3LD) = L(A)$$

Es decir, el lenguaje que genera la gramática es el mismo que reconoce el Autómata

Veamos como se obtiene la gramática  $G = \{\Sigma_T, \Sigma_N, S, P\}$  a partir del AF  $= \{Q, \Sigma, q_0, f, F\}$ .

[ 3 ]



Universidad  
Carlos III de Madrid  
educacion



## Gramática asociada a un AF

Se construye la **gramática G3LD** ( $G = \{\Sigma_T, \Sigma_N, S, P\}$ ) de la siguiente forma, **a partir del Autómata** ( $AF = \{\Sigma, Q, q_0, f, F\}$ ):

- $\Sigma_T = \Sigma$  ;  $\Sigma_N = Q$  ;  $S = q_0$
- $P = \{ \dots \}$ 
  1. transición  $f(p, a) = q \rightarrow$  si  $q \notin F \rightarrow p ::= a q$
  2. Si  $q \in F$  y  $f(p, a) = q \rightarrow p ::= a$  y  $p ::= a q$
  3. Si  $q_0 \in F \rightarrow q_0 ::= \lambda$  (es axioma para dar  $\lambda$ )
  4. si  $f(p, \lambda) = q \rightarrow$  si  $q \notin F \rightarrow p ::= q$  (redenominación);
  5.  $q \in F$  y  $f(p, \lambda) = q \rightarrow p ::= q$  y  $p ::= \lambda$  (redenominación y no generativa)

[ 4 ]



Universidad  
Carlos III de Madrid  
educacion



## AF asociado a una G3 (cuando es LD)

### De AF $\rightarrow$ G3: Ejemplo

Sea el AF descrito por la siguiente tabla, hallar la G3LD que genera el lenguaje por ella descrito. Comprobar que los lenguajes son iguales

	0	1
$\rightarrow A$	A	C
B	A	C
$C^*$	C	B

[ 5 ]



## AF asociado a una G3LD

Sea la G3LD,  $G=(\Sigma_T, \Sigma_N, S, P)$ , existe un AF, A, tal que:

$$L(G3LD) = L(A)$$

Cómo se construye **AF a partir de G3LD**:

- $\Sigma = \Sigma_T$
- $Q = \Sigma_N \cup \{F\}$ , con  $F \notin \Sigma_N$
- $q_0 = S$
- $F = \{F\}$
- f:
  - Si  $A ::= a B \rightarrow f(A, a) = B$
  - Si  $A ::= a \rightarrow f(A, a) = F$
  - Si  $S ::= \lambda \rightarrow f(S, \lambda) = F$  (equivalente a hacer  $q_0 = S \in F$ )

[ 6 ]



## AF asociado a una G3 (cuando es LD)

Se ha visto el procedimiento para obtener el AF que aceptaba el lenguaje descrito por una G3LD, sin embargo, ese procedimiento no siempre conduce a un AFD.

Lo habitual es:  $G3LD \rightarrow AFND \rightarrow AFD$

1. Ejemplo: Sea la G3LD hallar el AF correspondiente.

$$G = (\{d, c\}, \{A, S, T\}, A, \{A ::= cS, S ::= d \mid cS \mid dT, T ::= dT \mid d\})$$

[ 7 ]



Universidad  
Carlos III de Madrid  
cursos de grado



## AF asociado a una G3

¿Y si queremos obtener un AF a partir de una G3LI?

$$G3LI \rightarrow G3LD \rightarrow AF$$

¿Y si queremos obtener una G3LI a partir de un AF?

$$AF \rightarrow G3LD \rightarrow G3LI$$

[ 8 ]



Universidad  
Carlos III de Madrid  
cursos de grado



# EXPRESIONES REGULARES

[ 9 ]



Universidad  
Carlos III de Madrid  
cursos de posgrado



[ 10 ]



Universidad  
Carlos III de Madrid  
cursos de posgrado



## Definición de ER (I)

“Metalenguaje para expresar el conjunto de palabras aceptadas por un AF (es decir, para expresar lenguajes de tipo 3 o regulares)”

*Kleene, 1956*

[ 11 ]



## Definición de ER(I)

### Ejemplo

Dado el alfabeto  $\Sigma = \{0,1\}$ ,

La ER  $0^*10^*$  es una palabra del metalenguaje que representa las infinitas palabras del lenguaje regular formado por un 1, precedido y seguido de 0, 1 o infinitos 0s.

El lenguaje  $\Sigma^*$  puede representarse mediante la ER:

$$(0+1)^*$$

El lenguaje  $\{01, 101\}$  puede representarse mediante la ER:

$$01 + 101$$

La ER  $1(1+0)^*$  representa todas las cadenas que empiezan por el símbolo 1.

[ 12 ]



## Definición de ER(II)

Dados los elementos/símbolos:

$\Sigma$ ,  $\emptyset$  (lenguaje vacío),  $\lambda$  (palabra vacía)

y las operaciones:

$+$  (unión),  $\bullet$  (concatenación),  $*$  (cierre o clausura)

se cumple que:

- $\emptyset$  es una ER
- $\lambda$  es una ER
- cualquier  $a \in \Sigma$  es una ER
- si  $\alpha$  y  $\beta$  son EERR entonces  $\alpha + \beta$  y  $\alpha \bullet \beta$  son EERR
- si  $\alpha$  es una ER entonces  $\alpha^*$  es una ER, donde  $\alpha^* = \bigcup_{i=0}^{\infty} \alpha^i$

[ 13 ]



## Definición de ER(III)

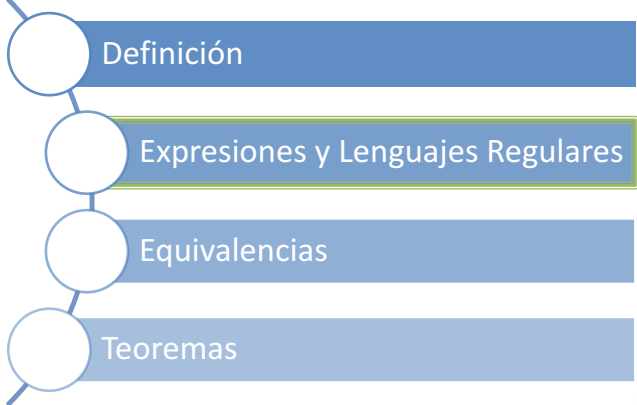
Solo son EERR las que se obtienen de aplicar las reglas anteriores **un número finito de veces** sobre símbolos de  $\Sigma$ ,  $\emptyset$ ,  $\lambda$

La prioridad de las operaciones es la siguiente:

$* > \bullet > +$

[ 14 ]





Definición

Expresiones y Lenguajes Regulares

Equivalencias

Teoremas

[ 15 ]

Universidad  
Cádiz (UCA)  
Cádiz

## EERR y LR

Cada Expresión Regular (ER) describe o expresa un lenguaje regular

A cada ER  $\alpha$ , se le asocia un subconjunto de  $\Sigma^*$ ,  $L(\alpha)$ , que es el LR descrito por  $\alpha$ . Este lenguaje se define con:

- si  $\alpha = \emptyset$ ,  $L(\alpha) = \emptyset$
- si  $\alpha = \lambda$ ,  $L(\alpha) = \{\lambda\}$
- si  $\alpha = a$ ,  $a \in \Sigma$ ,  $L(\alpha) = \{a\}$
- si  $\alpha$  y  $\beta$  son EERR  $\Rightarrow L(\alpha + \beta) = L(\alpha) \cup L(\beta)$
- si  $\alpha$  y  $\beta$  son EERR  $\Rightarrow L(\alpha \cdot \beta) = L(\alpha) L(\beta)$
- si  $\alpha$  es una ER  $\Rightarrow L(\alpha^*) = L(\alpha)^*$

[ 16 ]

Universidad  
Cádiz (UCA)  
Cádiz

Definición

Expresiones y Lenguajes Regulares

Equivalencias

Teoremas

[ 17 ]

Universidad Carlos III de Madrid

## Equivalencia de EERR (I)

Dos EERR son equivalentes,  $\alpha = \beta$ , si describen el mismo lenguaje regular, si  $L(\alpha) = L(\beta)$

Se cumple:

- 1)  $(\alpha + \beta) + \sigma = \alpha + (\beta + \sigma)$  (+ es asociativa)
- 2)  $\alpha + \beta = \beta + \alpha$  (+ es conmutativa)
- 3)  $(\alpha \cdot \beta) \cdot \sigma = \alpha \cdot (\beta \cdot \sigma)$  ( $\cdot$  es asociativa)
- 4)  $\alpha \cdot (\beta + \sigma) = (\alpha \cdot \beta) + (\alpha \cdot \sigma)$  (+ es distributiva)
- 5)  $(\beta + \sigma) \cdot \alpha = (\beta \cdot \alpha) + (\sigma \cdot \alpha)$  respecto de  $\cdot$
- 6)  $\alpha \cdot \lambda = \lambda \cdot \alpha = \alpha$  ( $\cdot$  tiene elemento neutro)
- 7)  $\alpha + \emptyset = \emptyset + \alpha = \alpha$  (+ tiene elemento neutro)
- 8)  $\lambda^* = \lambda$
- 9)  $\alpha \cdot \emptyset = \emptyset \cdot \alpha = \emptyset$

[ 18 ]

Universidad Carlos III de Madrid



## Equivalencia de EERR (II)

9)  $\emptyset^* = \lambda$

10)  $\alpha^* \bullet \alpha^* = \alpha^*$

11)  $\alpha \bullet \alpha^* = \alpha^* \bullet \alpha$

12)  $(\alpha^*)^* = \alpha^*$  (IMPORTANTE)

13)  $\alpha^* = \lambda + \alpha + \alpha^2 + \dots + \alpha^n + \alpha^{n+1} \bullet \alpha^*$

14)  $\alpha^* = \lambda + \alpha \bullet \alpha^*$  (13 con  $n=0$ ) (IMPORTANTE)

15)  $\alpha^* = (\lambda + \alpha)^{n-1} + \alpha^n \bullet \alpha^*$  (de 14, sustituyendo)

16) Sea  $f$  una función,  $f: E_\Sigma^n \rightarrow E_\Sigma$  se verifica:

$$f(\alpha, \beta, \dots, \sigma) + (\alpha + \beta + \dots + \sigma)^* = (\alpha + \beta + \dots + \sigma)^*$$

17) Sea  $f$  una función,  $f: E_\Sigma^n \rightarrow E_\Sigma$  se verifica:

$$(f(\alpha^*, \beta^*, \dots, \sigma^*))^* = (\alpha + \beta + \dots + \sigma)^*$$



[ 19 ]

## Equivalencia de EERR (III)

18)  $(\alpha^* + \beta^*)^* = (\alpha^* \bullet \beta^*)^* = (\alpha + \beta)^*$  (IMPORTANTE)

19)  $(\alpha \bullet \beta)^* \bullet \alpha = \alpha \bullet (\beta \bullet \alpha)^*$

20)  $(\alpha^* \bullet \beta)^* \bullet \alpha^* = (\alpha + \beta)^*$

21)  $(\alpha^* \bullet \beta)^* = \lambda + (\alpha + \beta)^* \bullet \beta$  (de 14 con 20)

22) Reglas de Inferencia:

Dadas tres EERR ( $L$ ,  $A$  y  $B$ ), sea la ecuación

$$L = AL + B,$$

donde  $\lambda \notin A$ , entonces se verifica que

$$L = A^*B$$



[ 20 ]

Definición

Expresiones y Lenguajes Regulares

Equivalencias

**Teoremas**

( 21 )

Universidad  
Cádiz (UCA)  
Escuela de Ingeniería

CC BY-NC-SA

## Teoremas de análisis y síntesis de Kleene

### 1. Teorema de análisis de Kleene

Todo lenguaje aceptado por un AF es un lenguaje regular.

*Solución al problema de análisis:*

Encontrar el lenguaje asociado a un determinado AF: “Dado un AF,  $A$ , encontrar la ER que describe  $L(A)$ ”.

### 1. Teorema de síntesis de Kleene

Todo lenguaje regular es el lenguaje aceptado por un AF.

*Solución al problema de síntesis:*

Encontrar un reconocedor para un lenguaje regular dado: “Dada una ER que representa a un lenguaje regular, construir un AF que acepte ese lenguaje regular”.

( 22 )

# 1. Solución al problema de análisis. Ecuaciones características.

## Problema Análisis: AF -> ER

### Resolución:

Dado un AF, escribir las ecuaciones características de cada uno de sus estados, resolverlas y obtener la ER buscada.

[ 23 ]



Universidad  
Carlos III de Madrid  
c3m.es



# Solución al problema de análisis. Ecuaciones características.

## ECUACIONES CARACTERÍSTICAS:

Describen todas las cadenas que se pueden reconocer desde un estado dado

Se escribe una ecuación  $x_i$  por estado  $q_i$

- Primer miembro:  $x_i$
- El segundo miembro tiene un término por cada rama que salga de  $q_i$ 
  - Las ramas tienen la forma  $a_{ij} \bullet x_j$  donde  $a_{ij}$  es la etiqueta de la rama que une  $q_i$  con  $q_j$ ,  $x_j$  es la variable correspondiente a  $q_j$
  - Se añade un término  $a_{ij}$  por cada rama que une  $q_i$  con un estado final
  - ~~Se añade  $\lambda$  si  $q_i$  es final.~~ SOLO si es final sin ramas o SOLO ramas al sumidero
  - Si de un estado  $q_i$  no sale ninguna rama, el segundo miembro será:
    - si es final:  $x_i = \lambda$
    - si no es final:  $x_i = \emptyset$

[ 24 ]

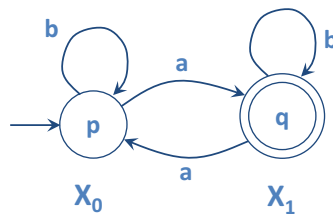


Universidad  
Carlos III de Madrid  
c3m.es



## Solución al problema de análisis. Ecuaciones características.

Ecuaciones Características de un AF – Ejemplo 1:



El AF tiene 2 estados, por lo que tendrá 2 ecuaciones características:

Conjunto de palabras que permiten pasar desde el estado  $p$  a un estado final.

$$X_0 = b X_0 + a X_1 + a$$

Porque  $q$  es un estado final

$$X_1 = b X_1 + a X_0 + b$$

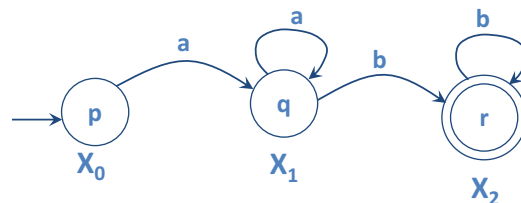
Porque  $q$  es un estado final

( 25 )



## Solución al problema de análisis. Ecuaciones características.

Ejemplo 2:



**Ecuaciones Características:**

$$X_0 = a X_1$$

$$X_1 = b X_2 + a X_1 + b$$

$$X_2 = b X_2 + b$$

( 26 )



## Algoritmo de resolución problema de Análisis.

1. Escribir las ecuaciones características del AF
2. Resolverlas
3. Si el estado inicial es  $q_0$ ,  $X_0$  nos da el conjunto de cadenas que conducen desde  $q_0$  a  $q_f$  y por tanto el lenguaje aceptado por el AF

[ 27 ]



## Solución de las ecuaciones características.

La Ecuación Característica de la forma:  $X = AX + B$ , donde:

$X$ : conjunto de cadenas que permiten pasar de  $q_i$  a  $q_f \in F$

$A$ : conjunto de cadenas que permiten, partiendo de un estado  $q$ , llegar a  $q$ .

$B$ : conjunto de cadenas que permiten llegar al estado final, sin volver a pasar por el  $q_i$  de partida.

⇓ (solución de Arden o reducción al absurdo)

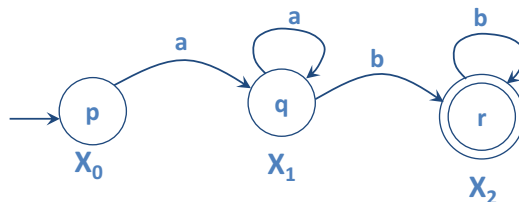
La solución es:  $X = A^* \cdot B$

[ 28 ]



## Solución al problema de análisis. Ecuaciones características.

Ejemplo 1:



**Ecuaciones Características:**

$$X_0 = a X_1$$

$$X_1 = b X_2 + a X_1 + b$$

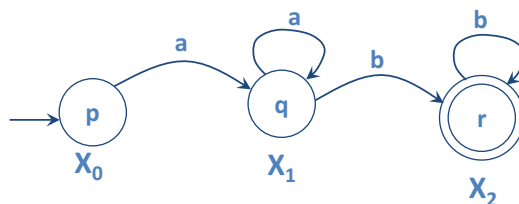
$$X_2 = b X_2 + b$$

[ 29 ]



## Solución al problema de análisis. Ecuaciones características.

Ejemplo 1:



Recuerda:

$$L = AL + B$$

$$L = A^*B$$

**Ecuaciones Características:**

$$X_0 = a X_1$$

$$X_1 = b X_2 + a X_1 + b$$

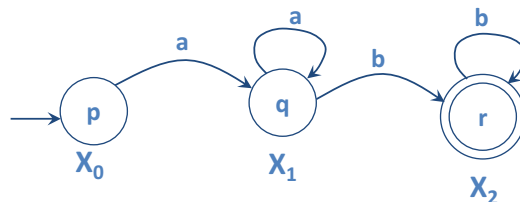
$$X_2 = b X_2 + b + \lambda$$

[ 30 ]



## Solución al problema de análisis. Ecuaciones características.

Ejemplo 1:



Recuerda:

$$L = AL + B$$

$$L = A^*B$$

**Ecuaciones Características:**

$$X_0 = a X_1$$

$$X_1 = b X_2 + a X_1 + b$$

$$X_2 = b X_2 + b$$

$$X_2 = b^* b$$

[ 31 ]

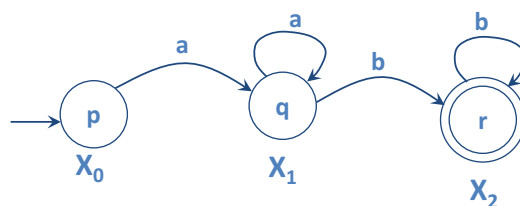


Universidad  
Carlos III de Madrid  
escuela de ingeniería



## Solución al problema de análisis. Ecuaciones características.

Ejemplo 1:



Recuerda:

$$L = AL + B$$

$$L = A^*B$$

**Ecuaciones Características:**

$$X_0 = a X_1$$

$$X_1 = b X_2 + a X_1 + b$$

$$X_2 = b X_2 + b$$

$$X_2 = b^* b$$

$$X_1 = b b^* b + a X_1 + b$$

$$X_1 = a X_1 + b b^* b + b$$

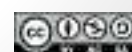
$$X_1 = a^* (b b^* b + b) =$$

$$a^* (b(b^* b + \lambda)) = a^* b b^*$$

[ 32 ]

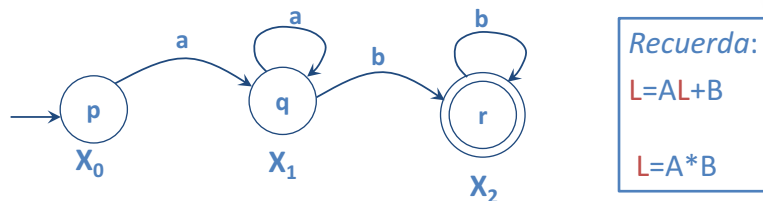


Universidad  
Carlos III de Madrid  
escuela de ingeniería



## Solución al problema de análisis. Ecuaciones características.

Ejemplo 1:



Recuerda:

$$L = AL + B$$

$$L = A^*B$$

**Ecuaciones Características:**

$$X_0 = a X_1$$

$$X_1 = b X_2 + a X_1 + b$$

$$X_2 = b X_2 + b$$

$$X_1 = a^*bb^*$$

$$X_0 = a(X_1) = aa^*bb^*$$

[ 33 ]

## 2. Problema de Síntesis: Algoritmo Recursivo (I)

Dada una ER que representa a un lenguaje regular, construir un AF que acepte ese lenguaje regular.

Sea  $\alpha$  una Expresión Regular

- si  $\alpha = \emptyset$ , el autómata será:
- si  $\alpha = \lambda$ , el autómata será:
- si  $\alpha = a$ ,  $a \in \Sigma$ , el autómata será:

[ 34 ]



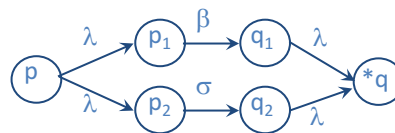
## Problema de Síntesis: Algoritmo Recursivo (II)

Dada una ER que representa a un lenguaje regular, construir un AF que acepte ese lenguaje regular (*cont.*)

- si  $\alpha = \beta + \sigma$ , con los autómatas de  $\beta$  y  $\sigma$ 

$\rightarrow p_1 \xrightarrow{\beta} *q_1$   
 $\rightarrow p_2 \xrightarrow{\sigma} *q_2$

el resultado es:



[ 35 ]

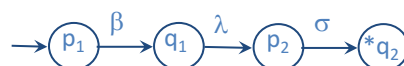
## Problema de Síntesis: Algoritmo Recursivo (III)

Dada una ER que representa a un lenguaje regular, construir un AF que acepte ese lenguaje regular. (*cont.*)

- si  $\alpha = \beta \cdot \sigma$ , con los autómatas de  $\beta$  y  $\sigma$ 

$\rightarrow p_1 \xrightarrow{\beta} *q_1$   
 $\rightarrow p_2 \xrightarrow{\sigma} *q_2$

el resultado es:



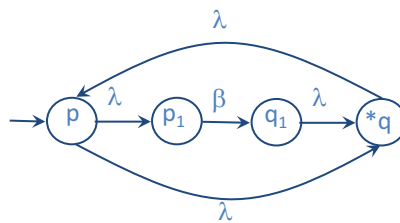
[ 36 ]

## Problema de Síntesis: Algoritmo Recursivo (IV)

Dada una ER que representa a un lenguaje regular, construir un AF que acepte ese lenguaje regular. (*cont.*)

- si  $\alpha = \beta^*$ , con el autómata de  $\beta$   $\rightarrow$  

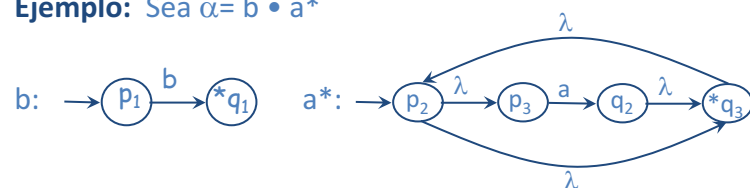
el resultado es:



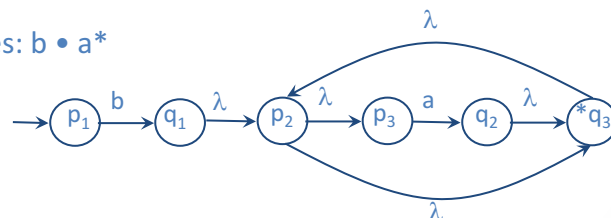
[ 37 ]

## Problema de Síntesis: Algoritmo Recursivo (IV)

**Ejemplo:** Sea  $\alpha = b \cdot a^*$



Entonces:  $b \cdot a^*$

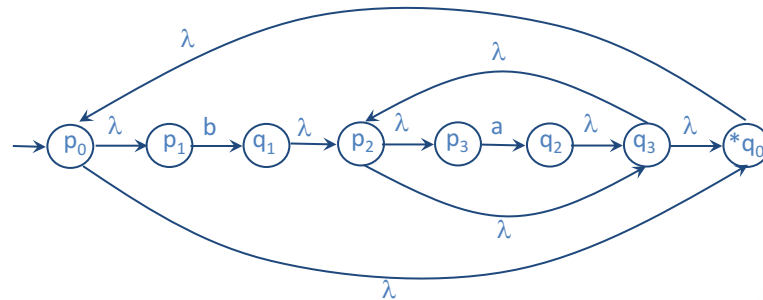


[ 38 ]

## Problema de Síntesis: Algoritmo Recursivo (IV)

**Ejemplo:**

Sea  $\alpha = (b \bullet a^*)^*$



[ 39 ]

## 2. Problema de Síntesis: Derivada de una ER.

Dada una ER, construir un AF que reconozca el lenguaje que la ER describe.

**Solución:** derivar la ER y obtener una G3LD y de ella un AF

Derivada de una ER:  $D_a(R) = \{ x \mid a \bullet x \in R \}$ .

- Derivada de ER R respecto de  $a \in \Sigma$  es el conjunto de colas de todas las palabras representadas por R cuya cabeza es a.

Veamos una definición recursiva

[ 40 ]

## Problema de Síntesis: Derivada de una ER

ER  $\rightarrow$  AF (Derivar la ER  $\rightarrow$  G3LD  $\rightarrow$  AF.  $Da(R) = \{ x \mid a \bullet x \in R \}$ )

Derivada de una ER. **Definición recursiva**

$\forall a, b \in \Sigma$  y  $R, S$  expresiones regulares

- $Da(\emptyset) = \emptyset$
- $Da(\lambda) = \emptyset$
- $Da(a) = \lambda, \quad a \in \Sigma$
- $Da(b) = \emptyset, \quad \forall b \neq a, b \in \Sigma$
- $Da(R+S) = Da(R) + Da(S)$
- $Da(R \bullet S) = Da(R) \bullet S + \delta(R) \bullet Da(S) \quad \forall R$ 
  - $\lambda \in R \Rightarrow \delta(R) = \lambda$
  - $\lambda \notin R \Rightarrow \delta(R) = \emptyset$
- $Da(R^*) = Da(R) \bullet R^*$

[ 41 ]



Universidad  
Carlos III de Madrid  
Instituto de Ingeniería de Software



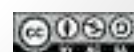
## Solución al problema de Síntesis. Derivada de una ER

- **Definición:**  $Dab(R) = Db(Da(R))$
- A partir de la Derivada de una ER. Se obtendrá la gramática regular lineal derecha:
  - El número de derivadas distintas de una ER es finito.  
Una vez que se han obtenido todas, se puede obtener la G3
  - **Si  $\delta(R) = \lambda$  añadimos  $R \rightarrow \lambda$**
  - Sea  $Da(R) = S$ , con  $S \neq \Phi$ 
    - $S \neq \lambda \Rightarrow R ::= aS \in P$
    - $S = \lambda \Rightarrow R ::= a \in P$
  - Sea  $\delta(Da(R)) = S$ 
    - $\delta(Da(R)) = \lambda \Rightarrow R ::= a \in P$
    - $\delta(Da(R)) = \Phi \Rightarrow$  no se incluye ninguna regla en  $P$
  - El axioma es  $R$  (ER de partida)
    - $\Sigma_T$  = símbolos que formaban la ER de partida
    - $\Sigma_N$  = letras que distinguen cada una de las derivadas distintas

[ 42 ]



Universidad  
Carlos III de Madrid  
Instituto de Ingeniería de Software



## Ejemplos. Derivada Expresiones Regulares

Obtener las G3 LD equivalentes a las ER dadas:

$R = a a^* b b^*$ ,  $\Sigma = \{a, b\}$

$R = a a^* b b^*$  es igual que  
 $R = a \cdot a^* \cdot b \cdot b^*$

- $Da(R) = Da(a) a^* b b^* = a^* b b^*$
- $Db(R) = \emptyset$
- $Daa(R) = Da(a^* b b^*) = Da(a^*) b b^* + \lambda Da(b b^*) = a^* b b^* = Da(R)$
- $Dab(R) = Db(a^* b b^*) = Db(a^*) b b^* + \lambda Db(b b^*) = b^*$
- $Daba(R) = Da(b^*) = \emptyset$
- $Dabb(R) = Db(b^*) = Db(b) b^* = b^* = Dab(R)$
  
- $Da(R) = a^* b b^*$                        $\delta(Da(R)) = \emptyset$
- $Daa(R) = a^* b b^*$                        $\delta(Daa(R)) = \emptyset$
- $Dab(R) = b^*$                                $\delta(Dab(R)) = \lambda$
- $Dabb(R) = b^*$                                $\delta(Dabb(R)) = \lambda$



[ 43 ]

## Ejemplos. Derivada Expresiones Regulares

- Obtención de la G3 LD que genera el Lenguaje representado por la ER:

- $R0 = aa^*bb^*$                        $R1 = a^*bb^*$                        $R2 = b^*$ 
  - $Da(R0) = R1$                        $\delta(Da(R0)) = \emptyset$
  - $Da(R1) = R1$                        $\delta(Da(R1)) = \emptyset$
  - $Db(R1) = R2$                        $\delta(Db(R1)) = \lambda$
  - $Db(R2) = R2$                        $\delta(Db(R2)) = \lambda$

- $Da(R) = S \Rightarrow R \rightarrow aS$                        $\delta(Da(R)) = \lambda \Rightarrow R \rightarrow a$ 
  - $R0 \rightarrow aR1$                       -----
  - $R1 \rightarrow aR1$                       -----
  - $R1 \rightarrow bR2$                        $R1 \rightarrow b$
  - $R2 \rightarrow bR2$                        $R2 \rightarrow b$



[ 44 ]

# Bibliografía

- Libro Básico 1 Bibliografía (AAM). Enrique Alfonseca Cubero, Manuel Alfonseca Cubero, Roberto Moriyón Salomón. Teoría de autómatas y lenguajes formales. McGraw-Hill (2007).  
Apartado 7.2
- Libro Básico 2 Bibliografía (HMU). John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman. Introducción a la teoría de autómatas, lenguajes y computación (3ª edición). Ed, Pearson Addison Wesley.  
Tema 3
- Libro Básico 4 Bibliografía (AAM). Manuel Alfonseca, Justo Sancho, Miguel Martínez Orga. Teoría de lenguajes, gramáticas y autómatas. Publicaciones R.A.E.C. 1997  
Tema 7

[ 45 ]



# 5. Lenguajes Regulares

Grado Ingeniería Informática  
Teoría de Autómatas y Lenguajes Formales



AUTÓMATAS FINITOS Y G3



[ 2 ]

## Gramática asociada a un AF

Sea el AF,  $A = (\Sigma, Q, q_0, f, F)$ , existe una G3 LD tal que

$$L(G3LD) = L(A)$$

Es decir, el lenguaje que genera la gramática es el mismo que reconoce el Autómata

Veamos como se obtiene la gramática  $G = \{\Sigma_T, \Sigma_N, S, P\}$  a partir del AF  $= \{Q, \Sigma, q_0, f, F\}$ .

[ 3 ]



## Gramática asociada a un AF

Se construye la **gramática G3LD** ( $G = \{\Sigma_T, \Sigma_N, S, P\}$ ) de la siguiente forma, **a partir del Autómata** ( $AF = \{\Sigma, Q, q_0, f, F\}$ ):

- $\Sigma_T = \Sigma$  ;  $\Sigma_N = Q$  ;  $S = q_0$
- $P = \{ \dots \}$ 
  1. transición  $f(p, a) = q \rightarrow$  si  $q \notin F \rightarrow p ::= a q$
  2. Si  $q \in F$  y  $f(p, a) = q \rightarrow p ::= a$  y  $p ::= a q$
  3. Si  $q_0 \in F \rightarrow q_0 ::= \lambda$  (es axioma para dar  $\lambda$ )
  4. si  $f(p, \lambda) = q \rightarrow$  si  $q \notin F \rightarrow p ::= q$  (redenominación);
  5.  $q \in F$  y  $f(p, \lambda) = q \rightarrow p ::= q$  y  $p ::= \lambda$  (redenominación y no generativa)

[ 4 ]





## AF asociado a una G3 (cuando es LD)

### De AF $\rightarrow$ G3: Ejemplo

Sea el AF descrito por la siguiente tabla, hallar la G3LD que genera el lenguaje por ella descrito. Comprobar que los lenguajes son iguales

	0	1
$\rightarrow A$	A	C
B	A	C
$C^*$	C	B

[ 5 ]



## AF asociado a una G3LD

Sea la G3LD,  $G=(\Sigma_T, \Sigma_N, S, P)$ , existe un AF, A, tal que:

$$L(G3LD) = L(A)$$

Cómo se construye **AF a partir de G3LD**:

- $\Sigma = \Sigma_T$
- $Q = \Sigma_N \cup \{F\}$ , con  $F \notin \Sigma_N$
- $q_0 = S$
- $F = \{F\}$
- f:
  - Si  $A ::= a B \rightarrow f(A, a) = B$
  - Si  $A ::= a \rightarrow f(A, a) = F$
  - Si  $S ::= \lambda \rightarrow f(S, \lambda) = F$  (equivalente a hacer  $q_0 = S \in F$ )

[ 6 ]



## AF asociado a una G3 (cuando es LD)

Se ha visto el procedimiento para obtener el AF que aceptaba el lenguaje descrito por una G3LD, sin embargo, ese procedimiento no siempre conduce a un AFD.

Lo habitual es:  $G3LD \rightarrow AFND \rightarrow AFD$

1. Ejemplo: Sea la G3LD hallar el AF correspondiente.

$$G = (\{d,c\}, \{A,S,T\}, A, \{A ::= cS, S ::= d|cS|dT, T ::= dT|d\})$$

[ 7 ]



## AF asociado a una G3

¿Y si queremos obtener un AF a partir de una G3LI?

$$G3LI \rightarrow G3LD \rightarrow AF$$

¿Y si queremos obtener una G3LI a partir de un AF?

$$AF \rightarrow G3LD \rightarrow G3LI$$

[ 8 ]



## EXPRESIONES REGULARES

[ 9 ]



[ 10 ]



## Definición de ER (I)

“Metalenguaje para expresar el conjunto de palabras aceptadas por un AF (es decir, para expresar lenguajes de tipo 3 o regulares)”

*Kleene, 1956*

[ 11 ]



## Definición de ER(I)

### Ejemplo

Dado el alfabeto  $\Sigma = \{0,1\}$ ,

La ER  $0^*10^*$  es una palabra del metalenguaje que representa las infinitas palabras del lenguaje regular formado por un 1, precedido y seguido de 0, 1 o infinitos 0s.

El lenguaje  $\Sigma^*$  puede representarse mediante la ER:

$$(0+1)^*$$

El lenguaje  $\{01, 101\}$  puede representarse mediante la ER:

$$01 + 101$$

La ER  $1(1+0)^*$  representa todas las cadenas que empiezan por el símbolo 1.

[ 12 ]



## Definición de ER(II)

Dados los elementos/símbolos:

$\Sigma$ ,  $\emptyset$  (lenguaje vacío),  $\lambda$  (palabra vacía)

y las operaciones:

$+$  (unión),  $\bullet$  (concatenación),  $*$  (cierre o clausura)

se cumple que:

- $\emptyset$  es una ER
- $\lambda$  es una ER
- cualquier  $a \in \Sigma$  es una ER
- si  $\alpha$  y  $\beta$  son EERR entonces  $\alpha + \beta$  y  $\alpha \bullet \beta$  son EERR
- si  $\alpha$  es una ER entonces  $\alpha^*$  es una ER, donde  $\alpha^* = \bigcup_{i=0}^{\infty} \alpha^i$

[ 13 ]



## Definición de ER(III)

Solo son EERR las que se obtienen de aplicar las reglas anteriores **un número finito de veces** sobre símbolos de  $\Sigma$ ,  $\emptyset$ ,  $\lambda$

La prioridad de las operaciones es la siguiente:

$$* > \bullet > +$$

[ 14 ]





Definición

Expresiones y Lenguajes Regulares

Equivalencias

Teoremas

[ 15 ]

Universidad Carlos III de Madrid

CC BY-NC-SA

## EERR y LR

Cada Expresión Regular (ER) describe o expresa un lenguaje regular

A cada ER  $\alpha$ , se le asocia un subconjunto de  $\Sigma^*$ ,  $L(\alpha)$ , que es el LR descrito por  $\alpha$ . Este lenguaje se define con:

- si  $\alpha = \emptyset$ ,  $L(\alpha) = \emptyset$
- si  $\alpha = \lambda$ ,  $L(\alpha) = \{\lambda\}$
- si  $\alpha = a$ ,  $a \in \Sigma$ ,  $L(\alpha) = \{a\}$
- si  $\alpha$  y  $\beta$  son EERR  $\Rightarrow L(\alpha + \beta) = L(\alpha) \cup L(\beta)$
- si  $\alpha$  y  $\beta$  son EERR  $\Rightarrow L(\alpha \cdot \beta) = L(\alpha) L(\beta)$
- si  $\alpha^*$  es una ER  $\Rightarrow L(\alpha^*) = L(\alpha)^*$

[ 16 ]

Universidad Carlos III de Madrid

CC BY-NC-SA

Definición

Expresiones y Lenguajes Regulares

Equivalencias

Teoremas

[ 17 ]

Universidad Carlos III de Madrid

CC BY-NC-SA

## Equivalencia de EERR (I)

Dos EERR son equivalentes,  $\alpha = \beta$ , si describen el mismo lenguaje regular, si  $L(\alpha) = L(\beta)$

Se cumple:

- 1)  $(\alpha + \beta) + \sigma = \alpha + (\beta + \sigma)$  (+ es asociativa)
- 2)  $\alpha + \beta = \beta + \alpha$  (+ es conmutativa)
- 3)  $(\alpha \cdot \beta) \cdot \sigma = \alpha \cdot (\beta \cdot \sigma)$  ( $\cdot$  es asociativa)
- 4)  $\alpha \cdot (\beta + \sigma) = (\alpha \cdot \beta) + (\alpha \cdot \sigma)$  (+ es distributiva)
- 5)  $(\beta + \sigma) \cdot \alpha = (\beta \cdot \alpha) + (\sigma \cdot \alpha)$  respecto de  $\cdot$
- 6)  $\alpha \cdot \lambda = \lambda \cdot \alpha = \alpha$  ( $\cdot$  tiene elemento neutro)
- 7)  $\alpha + \emptyset = \emptyset + \alpha = \alpha$  (+ tiene elemento neutro)
- 8)  $\lambda^* = \lambda$
- 9)  $\alpha \cdot \emptyset = \emptyset \cdot \alpha = \emptyset$

[ 18 ]

Universidad Carlos III de Madrid

CC BY-NC-SA

## Equivalencia de EERR (II)

9)  $\emptyset^* = \lambda$

10)  $\alpha^* \bullet \alpha^* = \alpha^*$

11)  $\alpha \bullet \alpha^* = \alpha^* \bullet \alpha$

12)  $(\alpha^*)^* = \alpha^*$  (IMPORTANTE)

13)  $\alpha^* = \lambda + \alpha + \alpha^2 + \dots + \alpha^n + \alpha^{n+1} \bullet \alpha^*$

14)  $\alpha^* = \lambda + \alpha \bullet \alpha^*$  (13 con  $n=0$ ) (IMPORTANTE)

15)  $\alpha^* = (\lambda + \alpha)^{n-1} + \alpha^n \bullet \alpha^*$  (de 14, sustituyendo)

16) Sea  $f$  una función,  $f: E_\Sigma^n \rightarrow E_\Sigma$  se verifica:

$$f(\alpha, \beta, \dots, \sigma) + (\alpha + \beta + \dots + \sigma)^* = (\alpha + \beta + \dots + \sigma)^*$$

17) Sea  $f$  una función,  $f: E_\Sigma^n \rightarrow E_\Sigma$  se verifica:

$$(f(\alpha^*, \beta^*, \dots, \sigma^*))^* = (\alpha + \beta + \dots + \sigma)^*$$



[ 19 ]

## Equivalencia de EERR (III)

18)  $(\alpha^* + \beta^*)^* = (\alpha^* \bullet \beta^*)^* = (\alpha + \beta)^*$  (IMPORTANTE)

19)  $(\alpha \bullet \beta)^* \bullet \alpha = \alpha \bullet (\beta \bullet \alpha)^*$

20)  $(\alpha^* \bullet \beta)^* \bullet \alpha^* = (\alpha + \beta)^*$

21)  $(\alpha^* \bullet \beta)^* = \lambda + (\alpha + \beta)^* \bullet \beta$  (de 14 con 20)

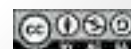
22) Reglas de Inferencia:

Dadas tres EERR ( $L$ ,  $A$  y  $B$ ), sea la ecuación

$$L = AL + B,$$

donde  $\lambda \notin A$ , entonces se verifica que

$$L = A^*B$$



[ 20 ]





Definición

Expresiones y Lenguajes Regulares

Equivalencias

**Teoremas**

[ 21 ]

Universidad Carlos III de Madrid  
Centro de Investigación en Matemáticas

CC BY-NC-SA

## Teoremas de análisis y síntesis de Kleene

### 1. Teorema de análisis de Kleene

Todo lenguaje aceptado por un AF es un lenguaje regular.

*Solución al problema de análisis:*

Encontrar el lenguaje asociado a un determinado AF: “Dado un AF,  $A$ , encontrar la ER que describe  $L(A)$ ”.

### 1. Teorema de síntesis de Kleene

Todo lenguaje regular es el lenguaje aceptado por un AF.

*Solución al problema de síntesis:*

Encontrar un reconocedor para un lenguaje regular dado: “Dada una ER que representa a un lenguaje regular, construir un AF que acepte ese lenguaje regular”.

[ 22 ]

Universidad Carlos III de Madrid  
Centro de Investigación en Matemáticas

CC BY-NC-SA

## 1. Solución al problema de análisis. Ecuaciones características.

### Problema Análisis: AF $\rightarrow$ ER

#### Resolución:

Dado un AF, escribir las ecuaciones características de cada uno de sus estados, resolverlas y obtener la ER buscada.

[ 23 ]



## Solución al problema de análisis. Ecuaciones características.

### ECUACIONES CARACTERÍSTICAS:

Describen todas las cadenas que se pueden reconocer desde un estado dado

Se escribe una ecuación  $x_i$  por estado  $q_i$

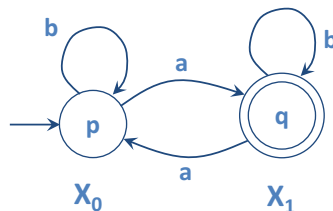
- Primer miembro:  $x_i$
- El segundo miembro tiene un término por cada rama que salga de  $q_i$ 
  - Las ramas tienen la forma  $a_{ij} \bullet x_j$  donde  $a_{ij}$  es la etiqueta de la rama que une  $q_i$  con  $q_j$ ,  $x_j$  es la variable correspondiente a  $q_j$
  - Se añade un término  $a_{ij}$  por cada rama que une  $q_i$  con un estado final
  - Se añade  $\lambda$  si  $q_i$  es final.
  - Si de un estado  $q_i$  no sale ninguna rama, el segundo miembro será:
    - si es final:  $x_i = \lambda$
    - si no es final:  $x_i = \emptyset$

[ 24 ]



## Solución al problema de análisis. Ecuaciones características.

Ecuaciones Características de un AF – Ejemplo 1:



El AF tiene 2 estados, por lo que tendrá 2 ecuaciones características:

Conjunto de palabras que permiten pasar desde el estado  $p$  a un estado final.

$$X_0 = b X_0 + a X_1 + a$$

Porque  $q$  es un estado final

$$X_1 = b X_1 + a X_0 + b$$

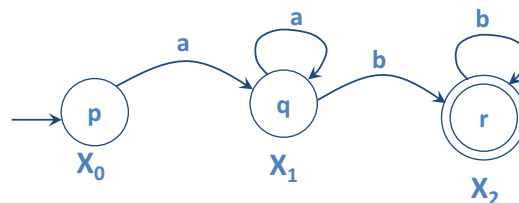
Porque  $q$  es un estado final

[ 25 ]



## Solución al problema de análisis. Ecuaciones características.

Ejemplo 2:



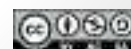
**Ecuaciones Características:**

$$X_0 = a X_1$$

$$X_1 = b X_2 + a X_1 + b$$

$$X_2 = b X_2 + b$$

[ 26 ]



## Algoritmo de resolución problema de Análisis.

1. Escribir las ecuaciones características del AF
2. Resolverlas
3. Si el estado inicial es  $q_0$ ,  $X_0$  nos da el conjunto de cadenas que conducen desde  $q_0$  a  $q_f$  y por tanto el lenguaje aceptado por el AF

[ 27 ]



## Solución de las ecuaciones características.

La Ecuación Característica de la forma:  $X = AX + B$ , donde:

$X$ : conjunto de cadenas que permiten pasar de  $q_i$  a  $q_f \in F$

$A$ : conjunto de cadenas que permiten, partiendo de un estado  $q$ , llegar a  $q$ .

$B$ : conjunto de cadenas que permiten llegar al estado final, sin volver a pasar por el  $q_i$  de partida.

⇓ (solución de Arden o reducción al absurdo)

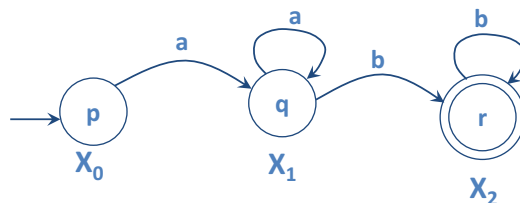
La solución es:  $X = A^* \cdot B$

[ 28 ]



## Solución al problema de análisis. Ecuaciones características.

Ejemplo 1:



**Ecuaciones Características:**

$$X_0 = a X_1$$

$$X_1 = b X_2 + a X_1 + b$$

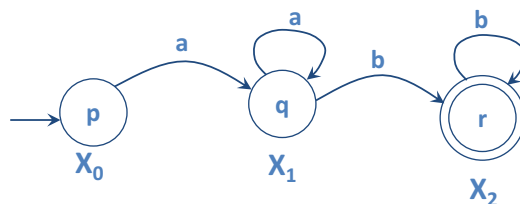
$$X_2 = b X_2 + b + \lambda$$

[ 29 ]



## Solución al problema de análisis. Ecuaciones características.

Ejemplo 1:



Recuerda:

$$L = AL + B$$

$$L = A^*B$$

**Ecuaciones Características:**

$$X_0 = a X_1$$

$$X_1 = b X_2 + a X_1 + b$$

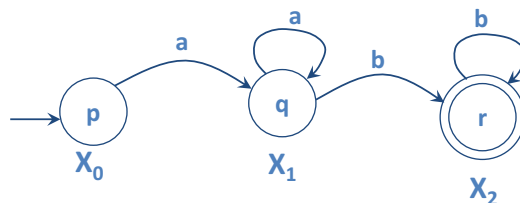
$$X_2 = b X_2 + b + \lambda$$

[ 30 ]



## Solución al problema de análisis. Ecuaciones características.

Ejemplo 1:



Recuerda:

$$L = AL + B$$

$$L = A^*B$$

**Ecuaciones Características:**

$$X_0 = a X_1$$

$$X_1 = b X_2 + a X_1 + b$$

$$X_2 = b X_2 + b + \lambda$$

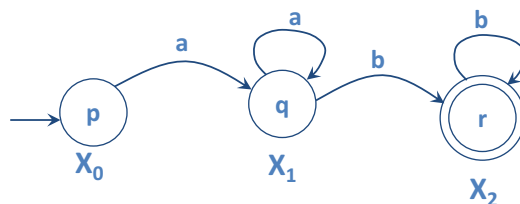
$$X_2 = b^* (b + \lambda)$$

[ 31 ]



## Solución al problema de análisis. Ecuaciones características.

Ejemplo 1:



Recuerda:

$$L = AL + B$$

$$L = A^*B$$

**Ecuaciones Características:**

$$X_0 = a X_1$$

$$X_1 = b X_2 + a X_1 + b$$

$$X_2 = b X_2 + b + \lambda$$

$$X_2 = b^* (b + \lambda)$$

$$X_1 = b b^* (b + \lambda) + a X_1 + b$$

$$X_1 = a X_1 + b b^* (b + \lambda) + b$$

$$X_1 = a^* (b b^* (b + \lambda) + b) =$$

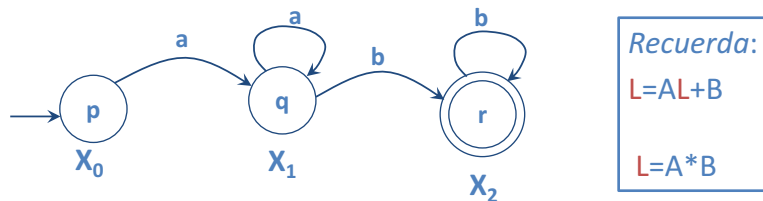
$$a^* b (b^* b + b^* + \lambda) = a^* b b^*$$

[ 32 ]



## Solución al problema de análisis. Ecuaciones características.

Ejemplo 1:



Recuerda:

$$L = AL + B$$

$$L = A^*B$$

**Ecuaciones Características:**

$$X_0 = a X_1$$

$$X_1 = b X_2 + a X_1 + b$$

$$X_2 = b X_2 + b + \lambda$$

$$X_2 = b^*(b + \lambda) = b^* + b^* = b^*$$

$$X_1 = b b^* + a X_1 + b$$

$$X_1 = a X_1 + b b^* + b$$

$$X_1 = a^*(b b^* + b) = a^* b b^*$$

$$X_0 = a a^* b b^*$$

[ 33 ]

## 2. Problema de Síntesis: Algoritmo Recursivo (I)

Dada una ER que representa a un lenguaje regular,  
construir un AF que acepte ese lenguaje regular.

Sea  $\alpha$  una Expresión Regular

- si  $\alpha = \emptyset$ , el autómata será:
- si  $\alpha = \lambda$ , el autómata será:
- si  $\alpha = a$ ,  $a \in \Sigma$ , el autómata será:

[ 34 ]

## Problema de Síntesis: Algoritmo Recursivo (II)

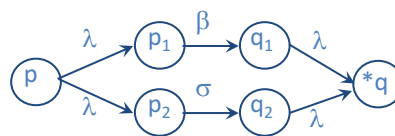
Dada una ER que representa a un lenguaje regular, construir un AF que acepte ese lenguaje regular (*cont.*)

- si  $\alpha = \beta + \sigma$ , con los autómatas de  $\beta$  y  $\sigma$ 

$\rightarrow p_1 \xrightarrow{\beta} *q_1$

$\rightarrow p_2 \xrightarrow{\sigma} *q_2$

el resultado es:



[ 35 ]

## Problema de Síntesis: Algoritmo Recursivo (III)

Dada una ER que representa a un lenguaje regular, construir un AF que acepte ese lenguaje regular. (*cont.*)

- si  $\alpha = \beta \cdot \sigma$ , con los autómatas de  $\beta$  y  $\sigma$ 

$\rightarrow p_1 \xrightarrow{\beta} *q_1$

$\rightarrow p_2 \xrightarrow{\sigma} *q_2$

el resultado es:



[ 36 ]

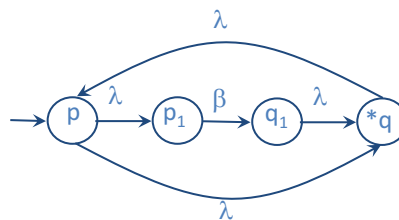


## Problema de Síntesis: Algoritmo Recursivo (IV)

Dada una ER que representa a un lenguaje regular, construir un AF que acepte ese lenguaje regular. (*cont.*)

- si  $\alpha = \beta^*$ , con el autómata de  $\beta$   $\rightarrow$  

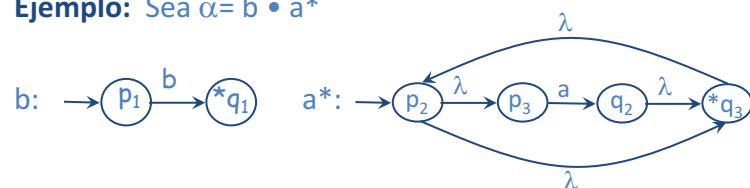
el resultado es:



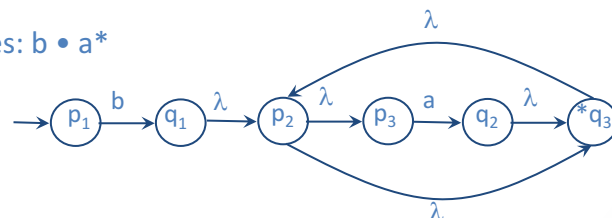
[ 37 ]

## Problema de Síntesis: Algoritmo Recursivo (IV)

**Ejemplo:** Sea  $\alpha = b \cdot a^*$



Entonces:  $b \cdot a^*$

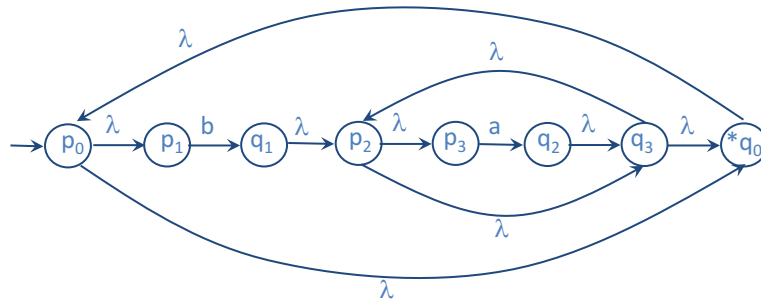


[ 38 ]

## Problema de Síntesis: Algoritmo Recursivo (IV)

**Ejemplo:**

Sea  $\alpha = (b \bullet a^*)^*$



[ 39 ]

## 2. Problema de Síntesis: Derivada de una ER.

Dada una ER, construir un AF que reconozca el lenguaje que la ER describe.

**Solución:** derivar la ER y obtener una G3LD y de ella un AF

Derivada de una ER:  $Da(R) = \{ x \mid a \bullet x \in R \}$ .

- Derivada de ER R respecto de  $a \in \Sigma$  es el conjunto de colas de todas las palabras representadas por R cuya cabeza es a.

Veamos una definición recursiva

[ 40 ]

## Problema de Síntesis: Derivada de una ER

ER  $\rightarrow$  AF (Derivar la ER  $\rightarrow$  G3LD  $\rightarrow$  AF.  $Da(R) = \{ x \mid a \bullet x \in R \}$ )

Derivada de una ER. **Definición recursiva**

$\forall a, b \in \Sigma$  y R,S expresiones regulares

- $Da(\emptyset) = \emptyset$
- $Da(\lambda) = \emptyset$
- $Da(a) = \lambda, \quad a \in \Sigma$
- $Da(b) = \emptyset, \quad \forall b \neq a, b \in \Sigma$
- $Da(R+S) = Da(R) + Da(S)$
- $Da(R \bullet S) = Da(R) \bullet S + \delta(R) \bullet Da(S) \quad \forall R$ 
  - $\lambda \in R \Rightarrow \delta(R) = \lambda$
  - $\lambda \notin R \Rightarrow \delta(R) = \emptyset$
- $Da(R^*) = Da(R) \bullet R^*$

[ 41 ]



Universidad  
Carlos III de Madrid  
Instituto de Ingeniería de Software



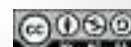
## Solución al problema de Síntesis. Derivada de una ER

- **Definición:**  $Dab(R) = Db(Da(R))$
- A partir de la Derivada de una ER. Se obtendrá la gramática regular lineal derecha:
  - El número de derivadas distintas de una ER es finito.  
Una vez que se han obtenido todas, se puede obtener la G3
  - Sea  $Da(R) = S$ , con  $S \neq \Phi$ 
    - $S \neq \lambda \Rightarrow R ::= aS \in P$
    - $S = \lambda \Rightarrow R ::= a \in P$
  - Sea  $\delta(Da(R)) = S$ 
    - $\delta(Da(R)) = \lambda \Rightarrow R ::= a \in P$
    - $\delta(Da(R)) = \Phi \Rightarrow$  no se incluye ninguna regla en P
  - El axioma es R (ER de partida)
    - $\Sigma_T$  = símbolos que formaban la ER de partida
    - $\Sigma_N$  = letras que distinguen cada una de las derivadas distintas

[ 42 ]



Universidad  
Carlos III de Madrid  
Instituto de Ingeniería de Software



## Ejemplos. Derivada Expresiones Regulares

Obtener las G3 LD equivalentes a las ER dadas:

$R = a a^* b b^*$ ,  $\Sigma = \{a, b\}$

$R = a a^* b b^*$  es igual que  
 $R = a \cdot a^* \cdot b \cdot b^*$

- $Da(R) = Da(a) a^* b b^* = a^* b b^*$
- $Db(R) = \emptyset$
- $Daa(R) = Da(a^* b b^*) = Da(a^*) b b^* + \lambda Da(b b^*) = a^* b b^* = Da(R)$
- $Dab(R) = Db(a^* b b^*) = Db(a^*) b b^* + \lambda Db(b b^*) = b^*$
- $Daba(R) = Da(b^*) = \emptyset$
- $Dabb(R) = Db(b^*) = Db(b) b^* = b^* = Dab(R)$

- |                        |                              |
|------------------------|------------------------------|
| - $Da(R) = a^* b b^*$  | $\delta(Da(R)) = \emptyset$  |
| - $Daa(R) = a^* b b^*$ | $\delta(Daa(R)) = \emptyset$ |
| - $Dab(R) = b^*$       | $\delta(Dab(R)) = \lambda$   |
| - $Dabb(R) = b^*$      | $\delta(Dabb(R)) = \lambda$  |



[ 43 ]

## Ejemplos. Derivada Expresiones Regulares

- Obtención de la G3 LD que genera el Lenguaje representado por la ER:

- |                    |                              |            |
|--------------------|------------------------------|------------|
| • $R0 = aa^* bb^*$ | $R1 = a^* bb^*$              | $R2 = b^*$ |
| • $Da(R0) = R1$    | $\delta(Da(R0)) = \emptyset$ |            |
| • $Da(R1) = R1$    | $\delta(Da(R1)) = \emptyset$ |            |
| • $Db(R1) = R2$    | $\delta(Db(R1)) = \lambda$   |            |
| • $Db(R2) = R2$    | $\delta(Db(R2)) = \lambda$   |            |

- |  |   |
|--|---|
| • $Da(R) = S \Rightarrow R \rightarrow aS$ | $\delta(Da(R)) = \lambda \Rightarrow R \rightarrow a$ |
| • $R0 \rightarrow aR1$                     | -----   |
| • $R1 \rightarrow aR1$                     | -----   |
| • $R1 \rightarrow bR2$                     | $R1 \rightarrow b$                                    |
| • $R2 \rightarrow bR2$                     | $R2 \rightarrow b$                                    |



[ 44 ]

# Bibliografía

- Libro Básico 1 Bibliografía (AAM). Enrique Alfonseca Cubero, Manuel Alfonseca Cubero, Roberto Moriyón Salomón. Teoría de autómatas y lenguajes formales. McGraw-Hill (2007).  
Apartado 7.2
- Libro Básico 2 Bibliografía (HMU). John E. Hopcroft, Rajeev Motwani, Jeffrey D.Ullman. Introducción a la teoría de autómatas, lenguajes y computación (3ª edición). Ed, Pearson Addison Wesley.  
Tema 3
- Libro Básico 4 Bibliografía (AAM). Manuel Alfonseca, Justo Sancho, Miguel Martínez Orga. Teoría de lenguajes, gramáticas y autómatas. Publicaciones R.A.E.C. 1997  
Tema 7

[ 45 ]



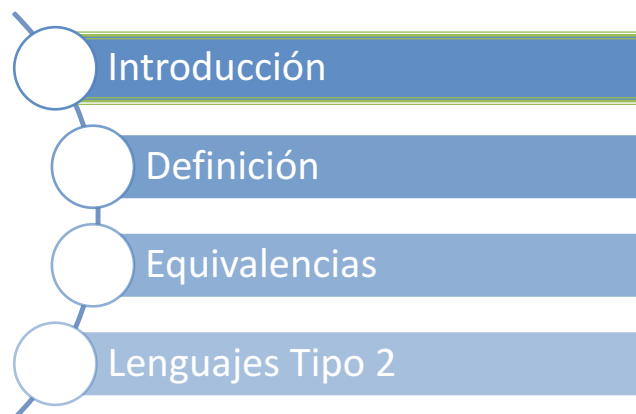
## **Parte VI**

### **TEMA 6. Autómatas a Pila**



# 6. Autómatas a Pila

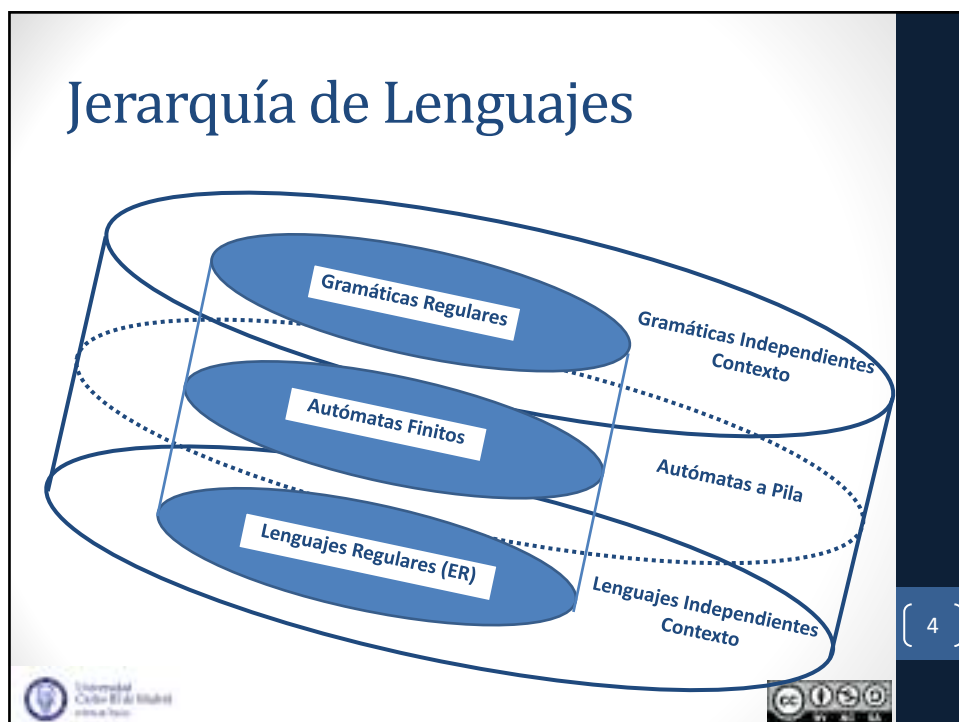
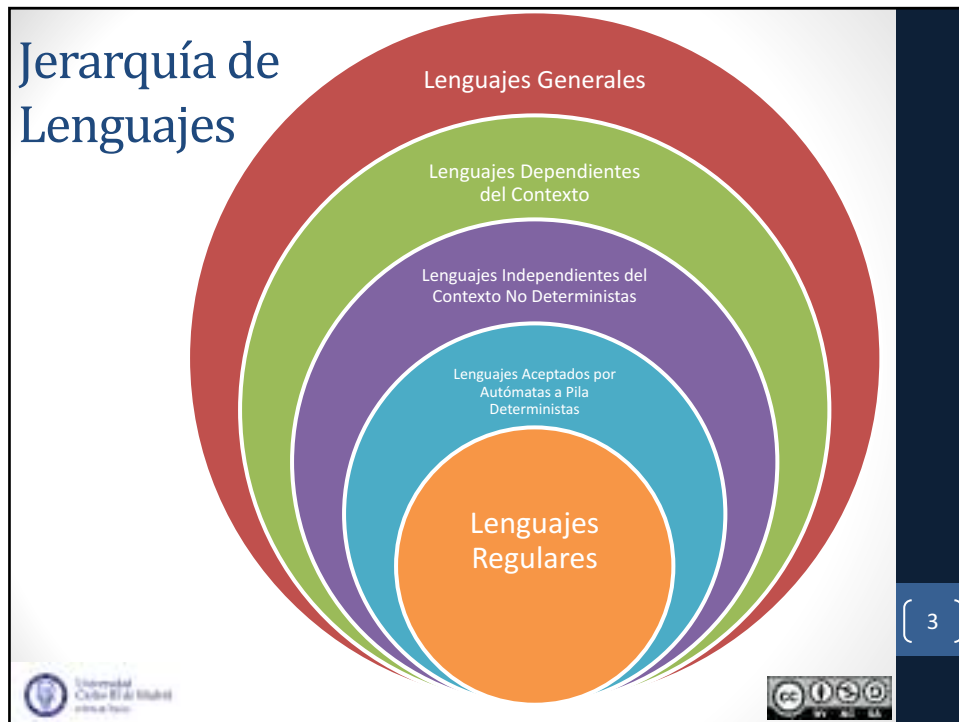
Grado Ingeniería Informática  
Teoría de Autómatas y Lenguajes Formales



[ 2 ]





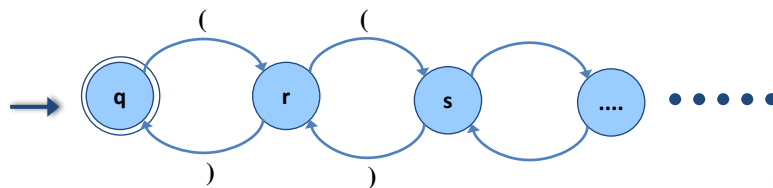


## Limitaciones de los AF

Falta de memoria



No pueden reconocer expresiones matemáticas,  
pej.  $(2x + (2+n/25))$ , mas general el lenguaje  $X^nY^n$



[ 5 ]



## Teoremas

- 1 Para cada gramática  $G$  independiente del contexto, existe un autómata de pila  $M$  tal que

$$L(G) = L(M)$$

- 2 Para cada autómata de pila  $M$ , existe una gramática  $G$  independiente del contexto tal que

$$L(M) = L(G)$$

- 3 Existe un lenguaje independiente del contexto que no es el lenguaje aceptado por ningún autómata de pila determinista

[ 6 ]



A vertical navigation menu with four items, each preceded by a circular icon with a pointer. The items are: 'Introducción', 'Definición' (highlighted with a green border), 'Equivalencias', and 'Lenguajes Tipo 2'. The slide number '[ 7 ]' is in the bottom right corner.

- Introducción
- Definición
- Equivalencias
- Lenguajes Tipo 2

[ 7 ]

## Definición de AP

The diagram illustrates the components of a Pushdown Automaton (AP). At the top, a horizontal 'CINTA' (tape) contains the symbols '/', '2', '5', ')', and ')'. Below the tape, a box labeled 'Q' represents the 'CONTROL DE ESTADOS' (state control). To the right of the state control is a vertical 'PILA' (stack) containing the symbols 'B', '...', '...', and 'A<sub>0</sub>'. A blue arrow labeled 'Movimiento de la cinta' points from the stack area back to the tape, indicating the movement of the tape head.

CINTA

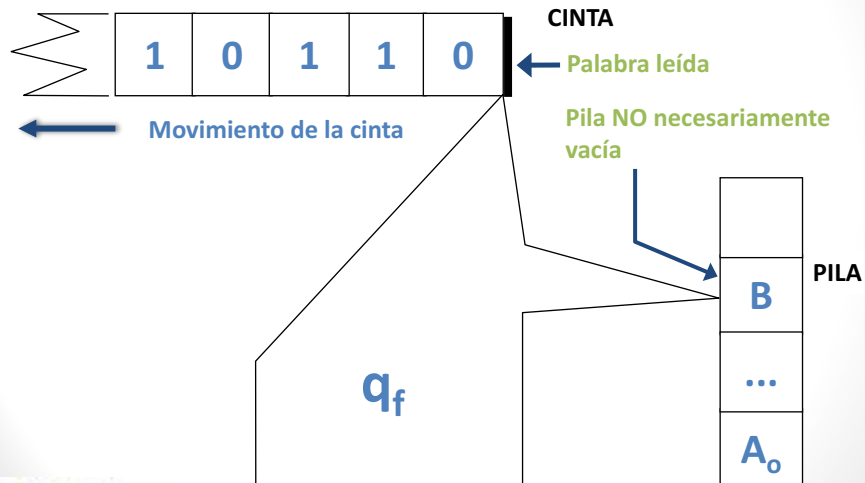
Movimiento de la cinta

Q  
CONTROL  
DE  
ESTADOS

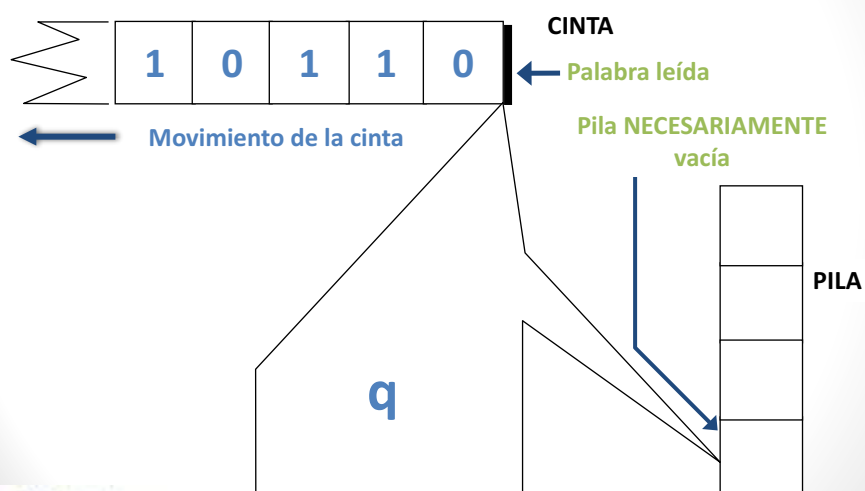
PILA

[ 8 ]

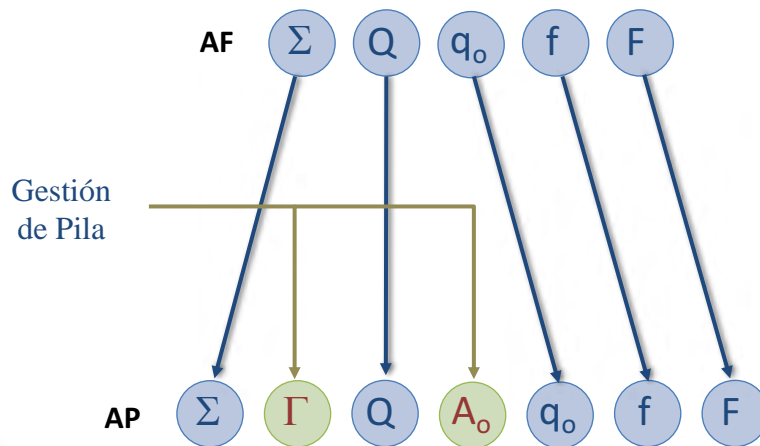
## Aceptación por estados finales



## Aceptación por vaciado de pila



## Definición formal



[ 11 ]

**AP:  $(\Sigma, \Gamma, Q, A_0, q_0, f, F)$**

$\Sigma$  : alfabeto de entrada

Palabras: x, y, z, ax, ay...

$\Gamma$  : alfabeto de pila

Palabras: X, Y, Z, AX, AY...

$Q$  : conjunto finito de estados

$Q = \{p, q, r, \dots\}$

$A_0 \in \Gamma$  : símbolo inicial de la pila

$q_0 \in Q$  : estado inicial del autómata

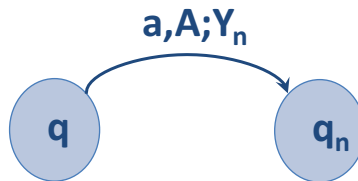
$f$  : función de transición

$F \subset Q$  : conjunto de estados finales

[ 12 ]

## Transición

$$f(q,a,A)=\{(q_1,Z_1),(q_2,Z_2),\dots,(q_n,Z_n)\}$$



$$(q,a,A;q_n,Y_n)$$

1. Leer un símbolo de la entrada
2. Extraer un símbolo de la pila
3. Insertar una palabra en la pila
4. Pasar a un nuevo estado

[ 13 ]



Universidad  
Carlos III de Madrid  
eduroad.es



## Función de Transición

$$f : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow P(Q \times \Gamma^*)$$

Transiciones dependientes  
de la entrada

$$Q \times \Sigma \times \Gamma$$

Transiciones independientes  
de la entrada

$$Q \times \lambda \times \Gamma$$

AP  
Deterministas

$$Q \times \Gamma^*$$

AP No  
Deterministas

$$P(Q \times \Gamma^*)$$

[ 14 ]



Universidad  
Carlos III de Madrid  
eduroad.es



## T. independientes de la entrada

Sea la transición:

$$f(q, \lambda, A) = \{(q_1, Z_1), (q_2, Z_2), \dots, (q_n, Z_n)\}$$

donde:

$$q, q_i \in Q$$

$$A \in \Gamma$$

$$Z_i \in \Gamma^*$$

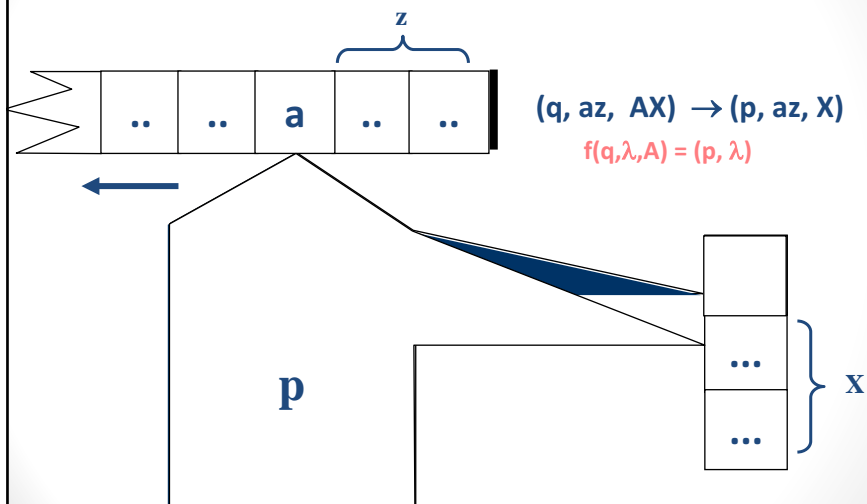
[ 15 ]



Universidad  
Carlos III de Madrid  
estudio de la



## T. independientes de la entrada



[ 16 ]



Universidad  
Carlos III de Madrid  
estudio de la



## T. dependientes de la entrada

Sea la transición:

$$f(q,a,A) = \{(q_1,Z_1), (q_2,Z_2), \dots, (q_n,Z_n)\}$$

donde:

$$q, q_i \in Q$$

$$a \in \Sigma$$

$$A \in \Gamma$$

$$Z_i \in \Gamma^*$$

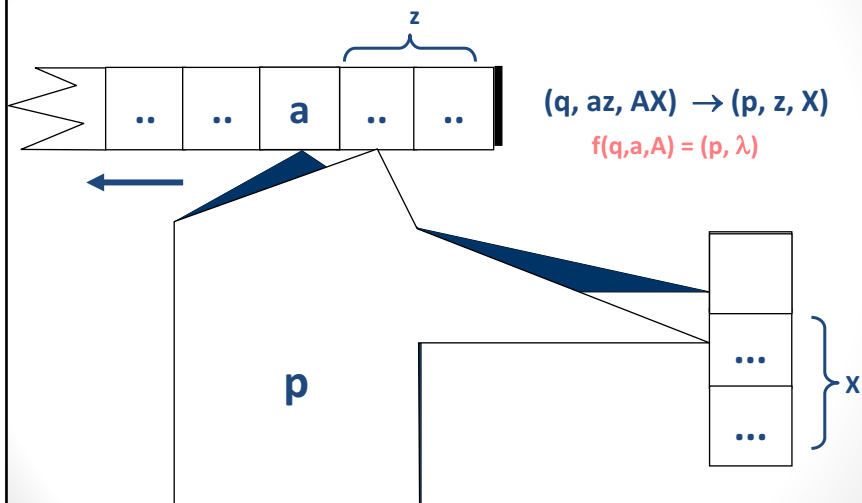
[ 17 ]



Universidad  
Carlos III de Madrid  
edmonar@cc.ics.uclm.es



## T. dependientes de la entrada



[ 18 ]



Universidad  
Carlos III de Madrid  
edmonar@cc.ics.uclm.es





## Descripción Instantánea

Permite describir sencillamente la configuración del AP en cada momento

Terna  $(q, x, z)$  donde:

$$q \in Q, x \in \Sigma^*, z \in \Gamma^*$$

Contiene:

- el estado actual ( $q$ )
- lo que queda por leer de la entrada ( $x$ ) y
- el contenido de la pila ( $z$ ) en un momento dado

[ 19 ]



Universidad  
Carlos III de Madrid  
cursos de posgrado



## Descripción Instantánea

**Movimiento:**  $(q, ay, AX) \vdash (p, y, YX)$

describe el paso de una descripción instantánea a otra

**Sucesión de movimientos:**

$(q, ay, AX) \vdash^* (p, y, YX)$

representa que desde la primera descripción instantánea se puede alcanzar la segunda

[ 20 ]



Universidad  
Carlos III de Madrid  
cursos de posgrado



## Autómatas a Pila Deterministas

$(\Sigma, \Gamma, Q, A_0, q_0, f, F)$  es determinista si verifica:

$$\forall q \in Q, A \in \Gamma, |f(q, \lambda, A)| > 0 \Rightarrow f(q, a, A) = \emptyset \quad \forall a \in \Sigma$$

$$\forall q \in Q, A \in \Gamma, \forall a \in \Sigma \cup \{\lambda\}, |f(q, a, A)| < 2$$

$$\text{Si } \exists f(q, \lambda, A) \nexists f(q, a, A) \text{ o}$$

$$\text{Si } \exists f(q, a, A) \nexists f(q, \lambda, A)$$

si  $(p, x, y; q, z)$  y  $(p, x, y; r, w)$  son transiciones de un autómata a pila determinista entonces

$$q = r, z = w$$

[ 21 ]



Universidad  
Carlos III de Madrid  
c3m.es



## Lenguaje aceptado por un AP

Por vaciado de pila

$$LV_{AP} = \{x \mid (q_0, x, A_0) \vdash^* (p, \lambda, \lambda), p \in Q, x \in \Sigma^*\}$$

Por estado final

$$LF_{AP} = \{x \mid (q_0, x, A_0) \vdash^* (p, \lambda, X), p \in F, x \in \Sigma^*, X \in \Gamma^*\}$$

[ 22 ]

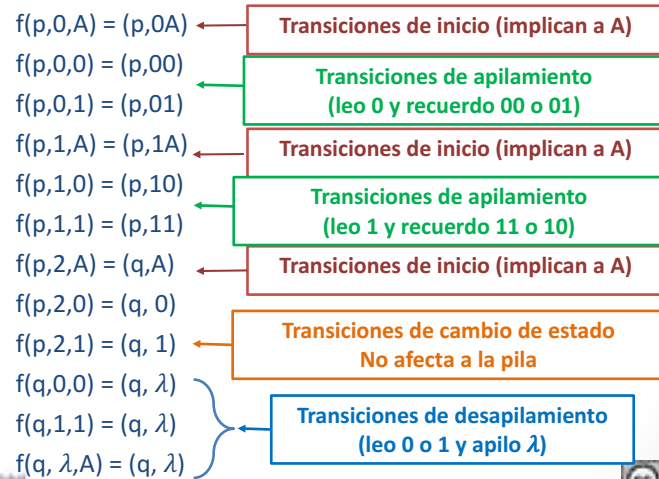


Universidad  
Carlos III de Madrid  
c3m.es



## Ejemplo Alfonseca (viejo pp 224-225)

Dado el  $AP_V = (\{0,1,2\}, \{A,0,1\}, \{p,q\}, A, p, f, \phi)$



[ 23 ]

## Ejemplo Alfonseca (viejo pp 224-225)

Es un AP por estado final

$$LF_{AP} = \{w \sqsupset w^{-1}, w \in (0+1)^*\}$$

[ 24 ]

## Ejemplo

LENGUAJE: algunas instrucciones  
 $\text{var} ::= \text{num};$  (asignación)

if cond  
 then  
 BLOQUE (asignación ó if)

if cond  
 then  
 BLOQUE (asignación ó if)  
 else  
 BLOQUE (asignación ó if)

[ 25 ]



## Ejemplo

AP= ({if, then, else, ::=, var, num, cond, ;},  
 $\{S, B, C, F, N, P, T, E\}, \{q\}, q, S, f, \phi$ )

$f(q, \text{var}, S) = \{(q, \text{FNP})\}$   
 $f(q, \text{if}, S) = \{(q, \text{CTBP}), (q, \text{CTBEBP})\}$   
 $f(q, \text{if}, B) = \{(q, \text{CTB}), (q, \text{CTBEB})\}$   
 $f(q, \text{var}, B) = \{(q, \text{FN})\}$   
 $f(q, \text{cond}, C) = \{(q, \lambda)\}$   
 $f(q, ::=, F) = \{(q, \lambda)\}$   
 $f(q, \text{num}, N) = \{(q, \lambda)\}$   
 $f(q, ,, P) = \{(q, \lambda)\}$   
 $f(q, \text{then}, T) = \{(q, \lambda)\}$   
 $f(q, \text{else}, E) = \{(q, \lambda)\}$

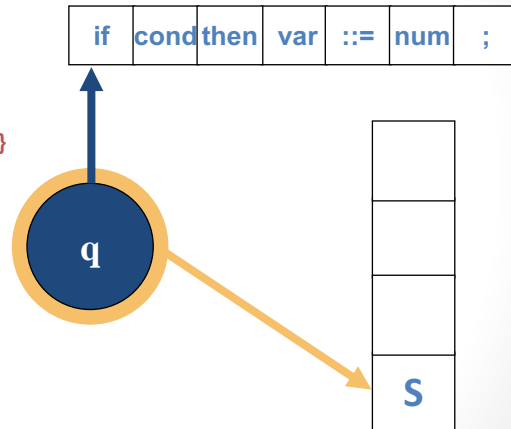
ELEMENTOS DE  $\Gamma$   
 $S \rightarrow$  Símbolo inicial  
 $F \rightarrow ::=$   
 $N \rightarrow$  Numero  
 $P \rightarrow ;$   
 $C \rightarrow$  Condición  
 $T \rightarrow$  Then  
 $B \rightarrow$  Bloque  
 $E \rightarrow$  Else

[ 26 ]



## Ejemplo

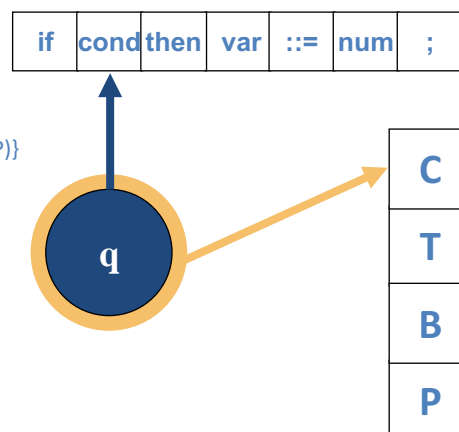
$f(q, \text{var}, S) = \{(q, \text{FNP})\}$   
 $f(q, \text{if}, S) = \{(q, \text{CTBP}), (q, \text{CTBEBP})\}$   
 $f(q, \text{if}, B) = \{(q, \text{CTB}), (q, \text{CTBEB})\}$   
 $f(q, \text{var}, B) = \{(q, \text{FN})\}$   
 $f(q, \text{cond}, C) = \{(q, \lambda)\}$   
 $f(q, ::=, F) = \{(q, \lambda)\}$   
 $f(q, \text{num}, N) = \{(q, \lambda)\}$   
 $f(q, ,, P) = \{(q, \lambda)\}$   
 $f(q, \text{then}, T) = \{(q, \lambda)\}$   
 $f(q, \text{else}, E) = \{(q, \lambda)\}$



[ 27 ]

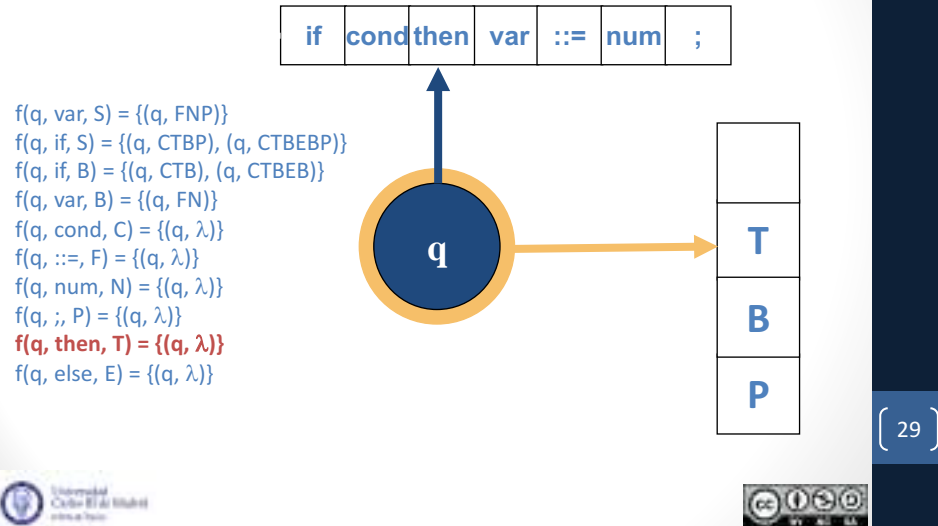
## Ejemplo

$f(q, \text{var}, S) = \{(q, \text{FNP})\}$   
 $f(q, \text{if}, S) = \{(q, \text{CTBP}), (q, \text{CTBEBP})\}$   
 $f(q, \text{if}, B) = \{(q, \text{CTB}), (q, \text{CTBEB})\}$   
 $f(q, \text{var}, B) = \{(q, \text{FN})\}$   
 $f(q, \text{cond}, C) = \{(q, \lambda)\}$   
 $f(q, ::=, F) = \{(q, \lambda)\}$   
 $f(q, \text{num}, N) = \{(q, \lambda)\}$   
 $f(q, ,, P) = \{(q, \lambda)\}$   
 $f(q, \text{then}, T) = \{(q, \lambda)\}$   
 $f(q, \text{else}, E) = \{(q, \lambda)\}$

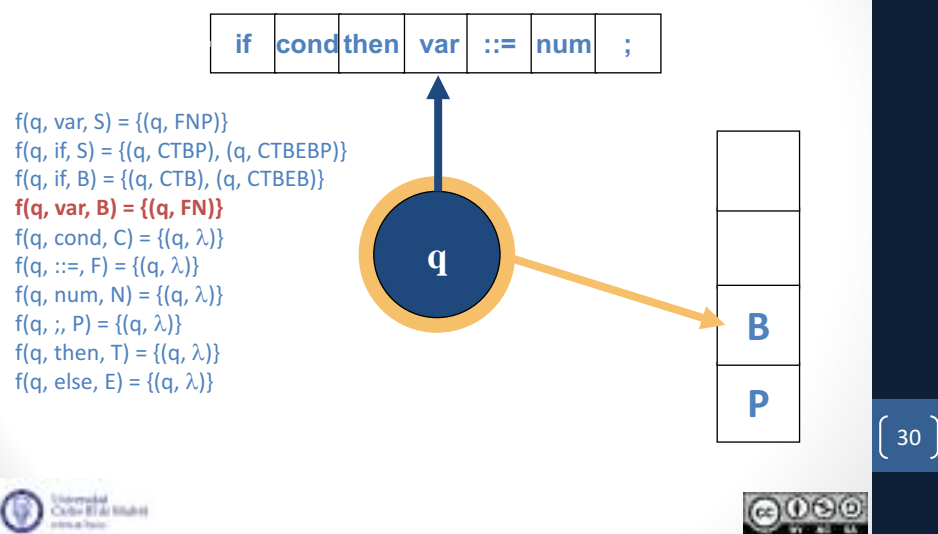


[ 28 ]

## Ejemplo



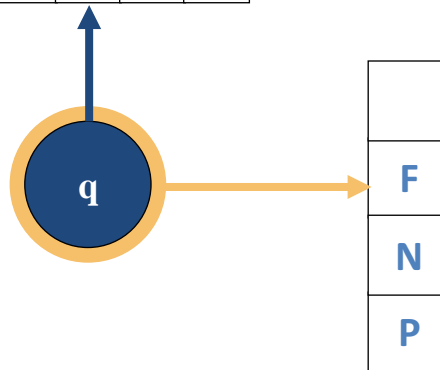
## Ejemplo



## Ejemplo

if	cond	then	var	::=	num	;
----	------	------	-----	-----	-----	---

$f(q, \text{var}, S) = \{(q, \text{FNP})\}$   
 $f(q, \text{if}, S) = \{(q, \text{CTBP}), (q, \text{CTBEBP})\}$   
 $f(q, \text{if}, B) = \{(q, \text{CTB}), (q, \text{CTBEB})\}$   
 $f(q, \text{var}, B) = \{(q, \text{FN})\}$   
 $f(q, \text{cond}, C) = \{(q, \lambda)\}$   
 $f(q, \text{::=}, F) = \{(q, \lambda)\}$   
 $f(q, \text{num}, N) = \{(q, \lambda)\}$   
 $f(q, \text{;, P}) = \{(q, \lambda)\}$   
 $f(q, \text{then}, T) = \{(q, \lambda)\}$   
 $f(q, \text{else}, E) = \{(q, \lambda)\}$

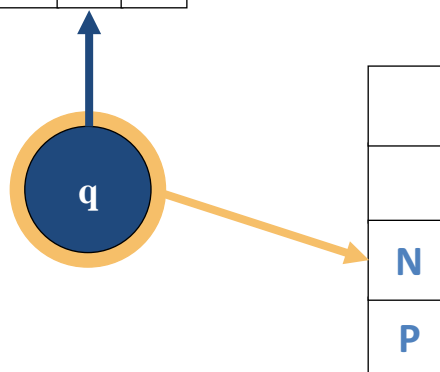


[ 31 ]

## Ejemplo

if	cond	then	var	::=	num	;
----	------	------	-----	-----	-----	---

$f(q, \text{var}, S) = \{(q, \text{FNP})\}$   
 $f(q, \text{if}, S) = \{(q, \text{CTBP}), (q, \text{CTBEBP})\}$   
 $f(q, \text{if}, B) = \{(q, \text{CTB}), (q, \text{CTBEB})\}$   
 $f(q, \text{var}, B) = \{(q, \text{FN})\}$   
 $f(q, \text{cond}, C) = \{(q, \lambda)\}$   
 $f(q, \text{::=}, F) = \{(q, \lambda)\}$   
 $f(q, \text{num}, N) = \{(q, \lambda)\}$   
 $f(q, \text{;, P}) = \{(q, \lambda)\}$   
 $f(q, \text{then}, T) = \{(q, \lambda)\}$   
 $f(q, \text{else}, E) = \{(q, \lambda)\}$

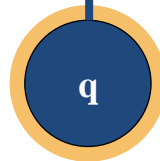


[ 32 ]

## Ejemplo

if	cond	then	var	::=	num	;
----	------	------	-----	-----	-----	---

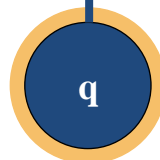
$f(q, \text{var}, S) = \{(q, \text{FNP})\}$   
 $f(q, \text{if}, S) = \{(q, \text{CTBP}), (q, \text{CTBEBP})\}$   
 $f(q, \text{if}, B) = \{(q, \text{CTB}), (q, \text{CTBEB})\}$   
 $f(q, \text{var}, B) = \{(q, \text{FN})\}$   
 $f(q, \text{cond}, C) = \{(q, \lambda)\}$   
 $f(q, ::=, F) = \{(q, \lambda)\}$   
 $f(q, \text{num}, N) = \{(q, \lambda)\}$   
 **$f(q, ;, P) = \{(q, \lambda)\}$**   
 $f(q, \text{then}, T) = \{(q, \lambda)\}$   
 $f(q, \text{else}, E) = \{(q, \lambda)\}$



[ 33 ]

## Ejemplo

if	cond	then	var	::=	num	;
----	------	------	-----	-----	-----	---

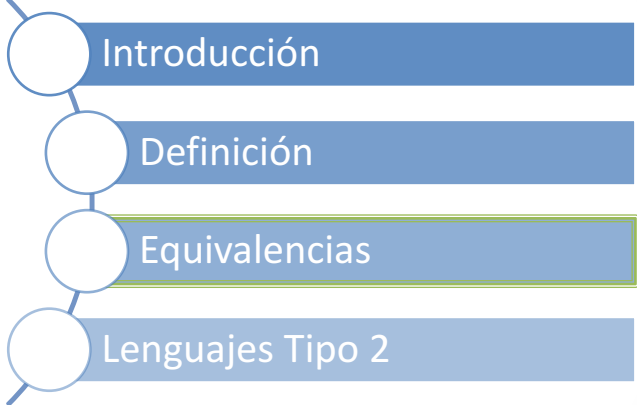


**Pila vacía**  
**Sentencia reconocida**



[ 34 ]





A vertical list of four items, each preceded by a white circle with a blue outline. The items are: 'Introducción', 'Definición', 'Equivalencias', and 'Lenguajes Tipo 2'. The 'Equivalencias' item is highlighted with a green border.

Introducción

Definición

Equivalencias

Lenguajes Tipo 2

[ 35 ]

Universidad Carlos III de Madrid

CC BY-NC-SA

# Equivalencias

**Teorema**

Para cada autómatas de pila que acepte cadenas sin vaciar su pila, existe un autómatas equivalente pero que vacía su pila antes de llegar a un estado de aceptación.

[ 36 ]

Universidad Carlos III de Madrid

CC BY-NC-SA

## Paso de $AP_F$ a $AP_V$

$$AP_F = (\Sigma, \Gamma, Q, A_0, q_0, f, F)$$

$$AP_V = (\Sigma, \Gamma \cup \{B\}, Q \cup \{p, r\}, B, p, f', \phi)$$

1. Nuevo símbolo para la pila
2. Dos estados nuevos
3. Valor inicial de la pila
4. Nuevo estado inicial
5. SIN estados finales

[ 37 ]



Universidad  
Carlos III de Madrid

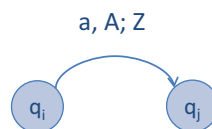
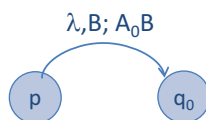


## Paso de $AP_F$ a $AP_V$

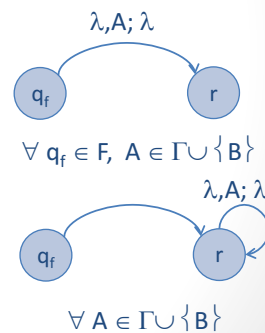
$$AP_F = (\Sigma, \Gamma, Q, A_0, q_0, f, F)$$

$$AP_V = (\Sigma, \Gamma \cup \{B\}, Q \cup \{p, r\}, B, p, f', \phi)$$

$f'$  se define así:



$q_i, q_j \in Q, a \in \Sigma \cup \{\lambda\}, A \in \Gamma, Z \in \Gamma^*$



[ 38 ]



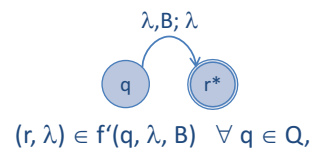
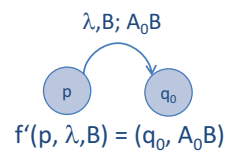
Universidad  
Carlos III de Madrid



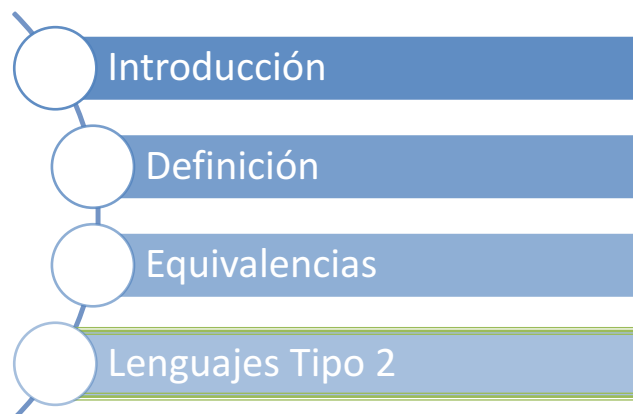
## Paso de $AP_V$ a $AP_F$

$$AP_V = (\Sigma, \Gamma, Q, A_0, q_0, f, \phi) \rightarrow AP_F = (\Sigma, \Gamma \cup \{B\}, Q \cup \{p, r\}, B, p, f', \{r\})$$

$f'$  se define así:



$$f(q, a, A) = f'(q, a, A) \quad \forall q \in Q, a \in \Sigma \cup \{\lambda\}, A \in \Gamma$$

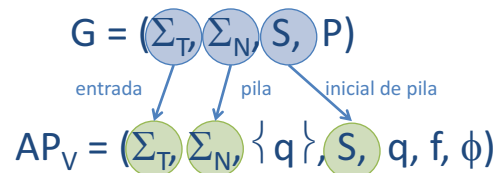


[ 40 ]



## De Gramática Tipo 2 a $AP_V$

Dada una G2 en FNG, construir un  $AP_V$ :



Se obtiene un  $AP_V$  con **un solo** estado

[ 41 ]



Universidad  
Carlos III de Madrid  
escuela de ingenieros



## De Gramática Tipo 2 a $AP_V$

**f** se define como:

$$(q, Z) \in f(q, a, A)$$

es decir:

$f(q, a, A) = (q, Z)$  si existe una producción del tipo  $A ::= aZ$

$f(q, a, A) = (q, \lambda)$  si existe una producción del tipo  $A ::= a$

$$f(q, a, A) = \{(q, Z), (q, \lambda)\}$$

dada una producción:

$$A ::= aZ \mid aD \mid b \Rightarrow f(q, a, A) = \{(q, Z), (q, D)\}$$

$$f(q, b, A) = (q, \lambda)$$

$$\text{Si } S ::= \lambda \Rightarrow (q, \lambda) \in f(q, \lambda, S)$$

[ 42 ]



Universidad  
Carlos III de Madrid  
escuela de ingenieros



## De $AP_V$ a Gramática Tipo 2

Dado un  $AP_V$ , construir una  $G_2$  tal que  $L(G_2) = L(AP_V)$

$$AP_V = (\Sigma, \Gamma, Q, A_0, q_0, f, \phi)$$

$$G = (\Sigma_T, \Sigma_N, S, P)$$

$$\{S\} \cup \{(pAq) \mid p, q \in Q, A \in \Gamma\}$$

Para construir P:

1.  $S ::= (q_0, A_0, q) \forall q \in Q$  (se eligen las que empiezan por  $q_0 A_0$ )
2. De cada transición  $f(p, a, A) = (q, BB'B''...B''')$   
donde:  
 $A, B, B', B'', ..., B''' \in \Gamma; a \in \Sigma \cup \{\lambda\}$   
se obtiene:  
 $(p A z) ::= a (q B r) (r B' s) s ... y (y B''' z)$
3. De cada transición  $f(p, a, A) = (q, \lambda)$  se obtienen:  $(p, A, q) ::= a$

[ 45 ]



## Bibliografía

- Libro Básico 1 Bibliografía. Enrique Alfonseca Cubero, Manuel Alfonseca Cubero, Roberto Moriyón Salomón. Teoría de autómatas y lenguajes formales. McGraw-Hill (2007).  
[Capítulo 4 y Apartado 8.1](#)
- Libro Básico 2 Bibliografía. John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman. Introducción a la teoría de autómatas, lenguajes y computación (3ª edición). Ed, Pearson Addison Wesley.  
[Capítulo 6](#)
- Libro Básico 4 Bibliografía. Manuel Alfonseca, Justo Sancho, Miguel Martínez Orga. Teoría de lenguajes, gramáticas y autómatas. Publicaciones R.A.E.C. 1997  
[Capítulo 10](#)

[ 46 ]



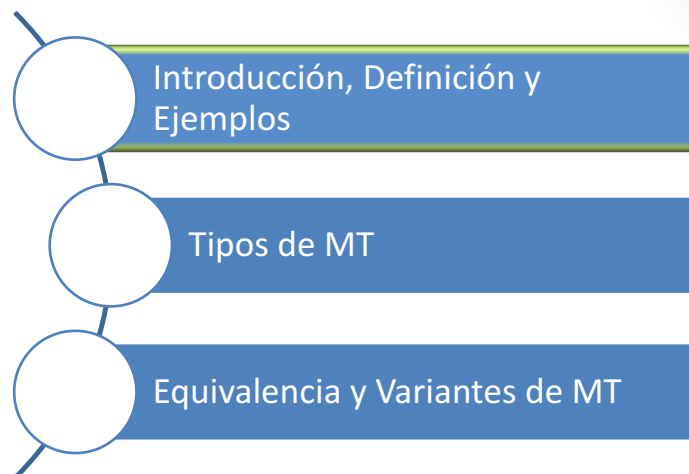
## **Parte VII**

### **TEMA 7. Maquina de Turing**



# 7. Máquinas de Turing

Grado Ingeniería Informática  
Teoría de Autómatas y Lenguajes Formales



[ 2 ]



# Introducción

## Origen:

- La Máquina de Turing (MT) fue descrita por Alan Turing en 1936.



AlanTuring.net

Alan Turing (Inglés: 1912 - 1956)

Fue un científico inglés que hizo grandes aportaciones en: matemáticas, criptoanálisis, lógica, filosofía, biología, ciencias de la computación, inteligencia artificial y vida artificial.

Es considerado uno de los padres de la ciencia de la computación. Es el precursor de la informática moderna.



[ 3 ]

# Introducción

- “Propongo considerar la siguiente cuestión: ¿**pueden pensar las máquinas?**”. Ha nacido la IA.
- Test de Turing

Turing: [http://www.youtube.com/watch?feature=player\\_embedded&v=ChJVatqU2So](http://www.youtube.com/watch?feature=player_embedded&v=ChJVatqU2So)



[ 4 ]

# Introducción

## Test de Turing:

- 1990, auspiciado por el filántropo Hugh Loebner, se celebra el **Premio Loebner** para honrar la memoria de Alan Turing.
- Una vez al año, robots de todo el mundo compiten para tratar de superar la prueba, aunque nadie ha podido hacerlo todavía.
- Si bien existen premios menores de consolación, los 100.000 dólares del premio gordo permanecen desiertos.
- Cuando un androide sea capaz de hacerse pasar por un humano antes los jueces, el concurso de desconvocará para siempre.
- Muchos han estado cerca, pero no ha llegado el día en que una máquina haya **imitado** la inteligencia humana.

[ 5 ]



# Introducción

## • Test de Turing:

- 2010, cuando [Suzette](#), de Bruce Wilcox, que volvió a ganar el certamen el año siguiente, logró engañar al juez del Premio Loebner. Una conversación sobre política en la que el *bot* sembró la confusión con una **imitación** casi perfecta de un humano.
- En la palabra **imitar** está la clave. Los críticos del test de Turing esgrimen el argumento de que la mayoría de los robots que se enfrentan a él no están basados en una auténtica inteligencia artificial, sino en una especialmente dirigida para superar la prueba. Además, todo está permitido, desde las respuestas absurdas hasta las mentiras.
- Wilcox es especialista en programación de *chatbots*, y su aplicación [Tom Loves Angela](#), es ya un clásico de los programas de conversación basados en inteligencia artificial.

[ 6 ]



## Introducción

- **Test de Turing:**

- Según los expertos, el Premio Loebner es la prueba más fiel al test original, aunque existen una serie de concursos en la misma línea. Entre ellos, **Turing 100**, donde hace unos años se produjo una sorpresa protagonizada por un robot que imitaba a un adolescente de trece años.
- El programa [Eugene Goostman](#), también obtuvo los mejores resultados en varias ediciones del Premio Loebner, obtuvo un 29% de respuestas consideradas humanas, cuando el límite para superar el test de Turing se encuentra en el 30%.



[ 7 ]

## Introducción

- **Test de Turing: Fecha límite**

- Como ocurre con la ley de Moore, sobre el test de Turing existen especulaciones sobre cuándo los robots superarán el límite de su inteligencia.
- Ray Kurzweil, ha predicho que un ordenador pasará la prueba de Turing en el año 2029, basándose en el concepto de la singularidad tecnológica.
- En la misma línea, Luis Von Ahn, uno de los mayores expertos en inteligencia artificial de nuestra época: *“es difícil saberlo, pero quizás hasta dentro de 10 o 20 años no sea posible. Por ejemplo, un ordenador capaz de escribir noticias correctamente estaría muy cerca de superar el test de Turing”*, explica el experto en ciencia computacional.



[ 8 ]

## Definición de una MT

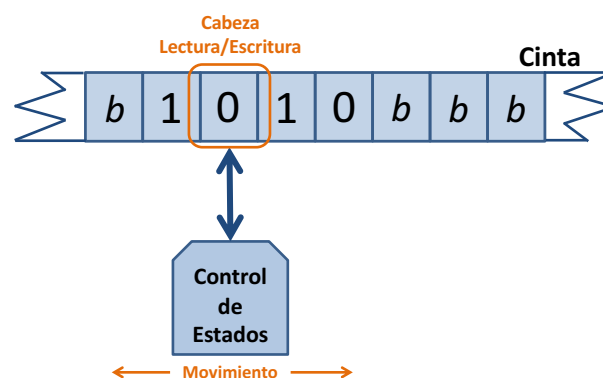


- Dispositivo hipotético capaz de manipular símbolos en una tira de cinta considerando ciertas reglas. A pesar de su simplicidad, pueden simular la lógica de cualquier algoritmo de un computador.
- Una MT está formado por:
  - Cinta infinita dividida en celdas
  - Cabezal de lectura/escritura capaz de moverse sobre dicha cinta.



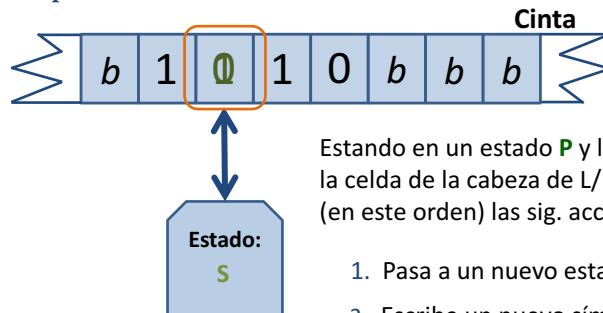
## Definición de una MT

Representación:



## Definición de una MT

Operaciones:



Estando en un estado **P** y leyendo el símbolo de la celda de la cabeza de L/E (Ej: 0), se realizan (en este orden) las sig. acciones:

1. Pasa a un nuevo estado. (Ej: S)
2. Escribe un nuevo símbolo en la cinta (reemplazando el existente). (Ej: 0 → 1)
3. Mueve el cabezal de L/E hacia:  
Dcha, Izqda, o no se mueve (Ej: Izqda)

[ 11 ]



## Definición de una MT

Definición Formal:

Una MT se define como una 7-tupla:

$$MT = (\Sigma, \Gamma, b, Q, q_0, f, F)$$

Donde:

Símbolo	
$\Sigma \subset \Gamma$	Alfabeto de entrada.
$\Gamma$	Alfabeto de símbolos de la cinta.
$b \in \Gamma$	Símbolo especial - espacio en blanco ( $b \notin \Sigma$ ). También se representa como: □
$Q$	Conjunto finito de estados.
$q_0 \in Q$	Estado inicial.
$f$	Función $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L(-), R(+), S(=)\}$ (donde L: Left, R: Right y S: Stay).
$F \subseteq Q$	Conjunto de estados finales o de aceptación.

[ 12 ]



## Definición de una MT

### Características:

- La cinta se supone infinita por ambos lados.
- Inicialmente la cinta contiene un número finito de símbolos consecutivos (de  $\Sigma$ ) precedidos y seguidos por el símbolo  $b$  (o  $\square$ ).
- La cabecera de L/E está situada inicialmente sobre el elemento más a la izquierda de la palabra.
- Toda MT se representa por una tabla de transición (como el resto de Autómatas). **Si la transición No es posible  $\rightarrow$  La MT se detiene.**

$f$ (Estados)	Símbolo	Símbolo	...
Estado	(Estado, Símbolo, Movim.)	(Estado, Símbolo, Movim.)	...
Estado	(Estado, Símbolo, Movim.)	(Estado, Símbolo, Movim.)	...
...	...	...	...

[ 13 ]



## Ejemplo de una MT

$MT\_1 = (\Sigma = \{0, 1\}, \Gamma = \{0, 1, b\}, b, Q = \{q_0, q_1, q_F\}, q_0, f, F = \{q_F\})$

donde  $f$ :

$f$	0	1	b
$\rightarrow q_0$	$(q_0, 0, R)$	$(q_1, 1, R)$	$(q_F, 0, S)$
$q_1$	$(q_1, 0, R)$	$(q_0, 1, R)$	$(q_F, 1, S)$
$*q_F$			

[ 14 ]



## Ejemplo de una MT

$$MT_1 = (\Sigma=\{0,1\}, \Gamma=\{0,1,b\}, b, Q=\{q_0, q_1, q_F\}, q_0, f, F=\{q_F\})$$

donde f:

f	0	1	b
$\rightarrow q_0$	$(q_0, 0, R)$	$(q_1, 1, R)$	$(q_F, 0, S)$
$q_1$	$(q_1, 0, R)$	$(q_0, 1, R)$	$(q_F, 1, S)$
$*q_F$			

Estado Inicial ( $\rightarrow$ )

Estado Final (\*)

También puede representarse como: □

Representación: (Estado al que transita,  
Símbolo que se escribe en la cinta,  
Movimiento que realiza el cabezal de L/E)

Desplazamiento:  
R  $\rightarrow$  Derecha  
L  $\rightarrow$  Izquierda  
S  $\rightarrow$  No se mueve

También puede representarse como: +, -, =

( 15 )



## Ejemplo de una MT

$$MT_1 = (\Sigma=\{0,1\}, \Gamma=\{0,1,b\}, b, Q=\{q_0, q_1, q_F\}, q_0, f, F=\{q_F\})$$

donde f:

f	0	1	b
$\rightarrow q_0$	$(q_0, 0, R)$	$(q_1, 1, R)$	$(q_F, 0, S)$
$q_1$	$(q_1, 0, R)$	$(q_0, 1, R)$	$(q_F, 1, S)$
$*q_F$			

( 16 )



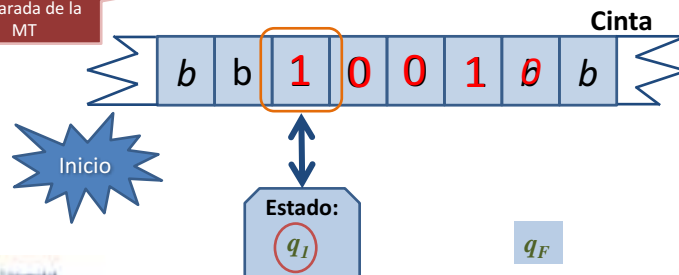
## Ejemplo de una MT

$$MT\_1 = (\Sigma=\{0,1\}, \Gamma=\{0,1,b\}, b, Q=\{q_0, q_1, q_F\}, q_0, f, F=\{q_F\})$$

donde f:

f	0	1	b
$\rightarrow q_0$	$(q_0, 0, R)$	$(q_1, 1, R)$	$(q_F, 0, S)$
$q_1$	$(q_1, 0, R)$	$(q_0, 1, R)$	$(q_F, 1, S)$
$*q_F$			

Sin transiciones  
→ Parada de la  
MT



( 17 )

## Ejemplo de una MT

$$MT\_1 = (\Sigma=\{0,1\}, \Gamma=\{0,1,b\}, b, Q=\{q_0, q_1, q_F\}, q_0, f, F=\{q_F\})$$

donde f:

f	0	1	b
$\rightarrow q_0$	$(q_0, 0, R)$	$(q_1, 1, R)$	$(q_F, 0, S)$
$q_1$	$(q_1, 0, R)$	$(q_0, 1, R)$	$(q_F, 1, S)$
$*q_F$			



¿Cómo funciona esta MT?

Al final de la palabra (en el primer b), escribe:  
0 → Si el número de 1's de la palabra leída es Par  
1 → Si el número de 1's de la palabra leída es Impar

( 18 )



# Definición de una MT

## Diagrama de Estados:

La función de transición también puede describirse en forma de diagrama de estados:

- Los nodos representan estados.
- Los arcos representan transiciones de estados.
- Cada arco es etiquetado con los prerequisites y los efectos de cada transición:
  - Símbolo inicial,
  - Símbolo que se escribe,
  - Dirección del movimiento del cabezal.

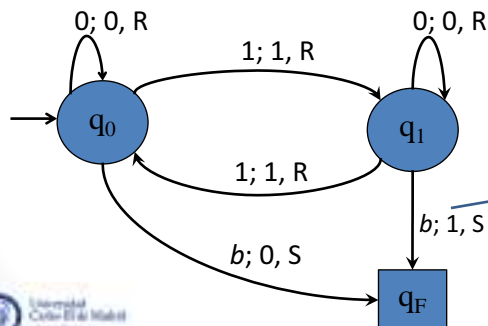
[ 19 ]



# Definición de una MT

## Diagrama de Estados - Ejemplo:

$f$	0	1	b
$\rightarrow q_0$	$(q_0, 0, R)$	$(q_1, 1, R)$	$(q_F, 0, S)$
$q_1$	$(q_1, 0, R)$	$(q_0, 1, R)$	$(q_F, 1, S)$
$*q_F$			



### Nomenclatura:

- 1º) Símbolo que se lee de la cinta (b)
- 2º) Símbolo que se escribe en la cinta (1)
- 3º) Movimiento que realiza el cabezal (S)

[ 20 ]





## Tipos de MT

**MT que actúa como TRANSDUCTOR:**

- Modifica el contenido de la cinta realizando cierta función.

**MT que actúa como RECONOCEDOR:**

- MT capaz de reconocer un lenguaje  $L$ .
- MT capaz de aceptar un lenguaje  $L$ .

[ 22 ]

Universidad Carlos III de Madrid  
www.uc3m.es

## Tipos de MT

### MT que actúa como TRANSDUCTOR:

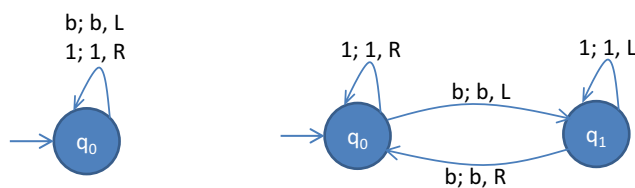
- Modifica el contenido de la cinta realizando cierta función.  
*Ejs: MT que sustituye los dígitos por cero,*  
*MT que añade un bit de paridad a la entrada,*  
*MT que duplica el número de 1's que hay en la cinta*  
 ...
- Si la Entrada está bien formada:  
 debe terminar en un Estado Final.
- Si la Entrada No está bien formada:  
 debe terminar en un Estado No Final.

[ 23 ]

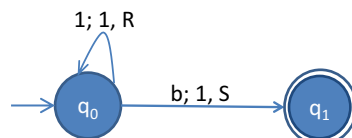


## Ejemplos de MT

Diferentes MT que no se detienen ( $\Gamma = \{1, b\}$ ):



MT que calcula  $n+1$  considerando el  $n$  ( $n \geq 0$ ) como una sucesión de 1's:



[ 24 ]



## Tipos de MT

### MT que actúa como TRANSDUCTOR:

- Modifica el contenido de la cinta realizando cierta función.

### MT que actúa como RECONOCEDOR:

- MT capaz de reconocer un lenguaje  $L$ .
- MT capaz de aceptar un lenguaje  $L$ .

[ 25 ]



## Tipos de MT

### MT que actúa como RECONOCEDOR:

- MT capaz de RECONOCER o ACEPTAR un lenguaje  $L$ .
- Una MT RECONOCE un lenguaje  $L$ , si dada una entrada ( $w$ ) en la cinta, la MT SIEMPRE se para, y lo hace en un Estado Final si y sólo si:  $w \in L$
- Una MT ACEPTA un lenguaje  $L$ , si dada una entrada ( $w$ ) en la cinta, la MT se para en un Estado Final si y sólo si:  $w \in L$ 
  - Así, en este caso, si  $w \notin L$ , la MT podría no parar.

Ejs: MT que reconoce el lenguaje  $a^*b^*$ ,

MT que acepta el lenguaje  $a^n b^n c^n$

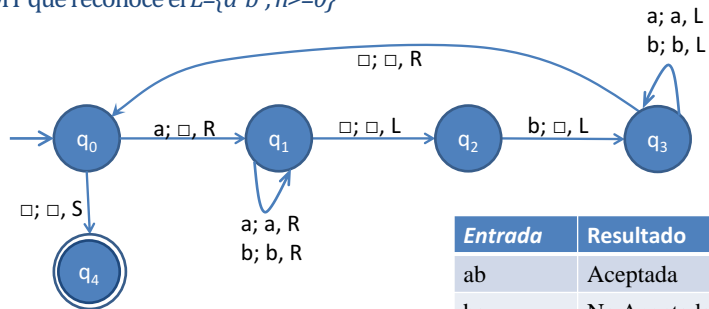
...

[ 26 ]



## Ejemplos de MT

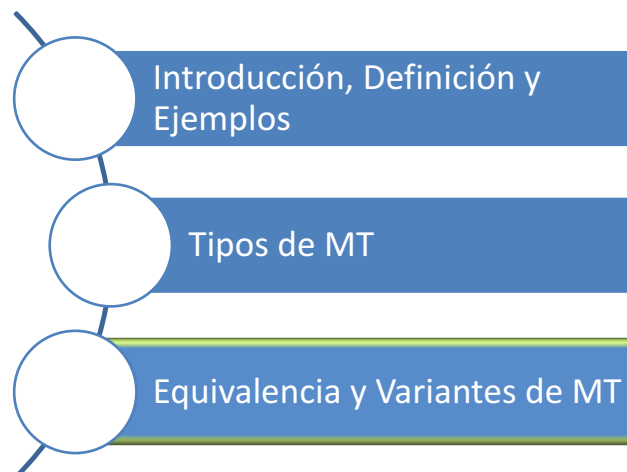
MT que reconoce el  $L = \{a^n b^n, n \geq 0\}$



En este caso, el símbolo especial- espacio en blanco ( $b \notin \Sigma$ ) se representa como:  $\square$ , porque  $b \in \Gamma$

Entrada	Resultado
ab	Aceptada
ba	No Aceptada
aabb	Aceptada
aab	No Aceptada
abb	No Aceptada
aaaabbbb	Aceptada

[ 27 ]



[ 28 ]

## Equivalencia de MT

### Dos MT son equivalentes si:

Ambas realizan la misma acción sobre TODAS sus entradas. Además, si una MT no se parara para alguna entrada, la otra tampoco podrá pararse.

- Si las MT actúan como **Transductor**:
  - Para cada entrada posible, los contenidos de la cinta al final del proceso deben ser iguales.
- Si las MT actúan como **Reconocedor**:
  - Ambas deben Aceptar y/o Reconocer las mismas palabras.

[ 29 ]



## Variantes de MT

- Existen numerosas variantes de MT obtenidas al restringir algún aspecto de las mismas.
- Consideremos algunos ejemplos:
  - MT con alfabeto binario ( $\Gamma = \{0, 1, b\}$ ).
  - MT limitada por un extremo.
  - MT con restricciones en el movimiento de L/E.

[ 30 ]



# La Máquina de Turing



- Breaking the Code: Biography of Alan Turing (Derek Jacobi, BBC, 1996)  
<http://www.youtube.com/watch?v=S23yie-779k>
- <https://www.youtube.com/watch?v=8fgIRhM9pkU>
- <http://www.youtube.com/watch?v=6k2OUZdA7vQ>

[ 31 ]



# MT Universal (MTU)

- MT capaz de simular el comportamiento de cualquier MT.
- Una MTU contiene en su cinta:
  1. La descripción de otra MT,
  2. El contenido de la cinta de dicha MT,
 y produce como resultado de su ejecución, el mismo resultado que produciría la MT sobre su cinta.
- Es una MT “programable”

[ 32 ]



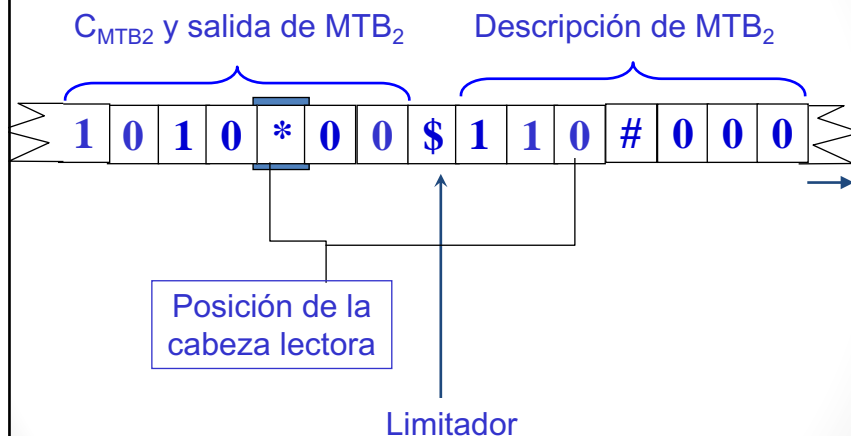
## MT Universal (I)

- Es una MT “programable”
  - Dependiendo del programa, puede simular a cualquier otra MT.
  - Lee y ejecuta programas almacenados en su cinta
  - El programa de una MTU es una versión codificada de una MT que lleva a cabo la tarea que se desea que ejecute la MTU.
- Para construir una MTU que desempeñe una tarea:
  - Hay que diseñar una MT genérica para esa tarea y
  - codificar dicha MT genérica en la cinta de la MTU.
  - Se tratará de una MT con alfabeto binario y cinta limitada en un sentido.
  - codificar la cadena de entrada.

[ 33 ]



## MT Universal



En torno a la figura de Alan Turing

[ 34 ]





## MT Universal. Cinta

- Información sobre:
  - función de transición
  - estado inicial
  - posición inicial de la cabeza
  - contenido inicial de la cinta
- Dividida en dos partes por el \$:
  - izda del \$: CMTB2
  - dcha del \$: codificación de MTB2
- Cabeza lectora: ¿tiene que leer en ambos lados del \$?
  - Donde está la cabeza: \*
  - contenido reemplazado por \*: en la posición "1" a la dcha del \$.
- A la dcha del \$, estructura análoga a MT Transcriptor de Información , con:
  - longitud de la dirección y etiquetas, " $l$ " =  $E(\log 2 / Q) + 1$
  - longitud de los registros, " $m$ " =  $2 \times l + 1$



[ 35 ]

## MT y Computabilidad

- Turing demostró con su MT (y sus extensiones) que todo problema computable (resoluble), lo es en una MTU.
- Complejidad computacional:
  - Se basa en tratar de dar respuesta a la siguiente pregunta:
    - ¿Qué hace a algunos problemas computacionalmente difíciles y a otros sencillos?
  - Estudia el orden de complejidad de un algoritmo que resuelve un problema *decidible*.
  - Para ello, considera los 2 tipos de recursos requeridos durante el cómputo para resolver un problema:
    - **Tiempo:** Número de pasos base de ejecución de un algoritmo para resolver un problema.
    - **Espacio:** Cantidad de memoria utilizada para resolver un problema.



[ 36 ]

## Funciones NO computables (I)

- ¿Cuáles son los problemas/funciones no computables?
- Equivalente a los “no-decidibles”:
  - **Función de Rado:**
  - **Secuencias aleatorias**
  - **El teorema de Fermat:**  $a^n + b^n = c^n$  /  $a, b, c \neq 0$  y  $n$  entero  $> 2$
  - **Los primos pares:**

[ 37 ]



## Funciones NO computables (II)

- ¿Cuáles son los problemas/funciones no computables?
- Equivalente a los “no-decidibles”:
  - **Función de Rado:** El Problema del Castor Afanoso: ¿Cuál es el número máximo de 1's que pueden ser escritos por una máquina de Turing de  $N$  estados (donde  $N$  no incluye el estado final) que termina en parada, y que comienza con una cinta inicialmente en blanco? Este número, que varía en función del número de estados de la máquina, se denota  $\sigma(N)$ . Una máquina que produce  $\sigma(N)$  celdas no en blanco se denomina Castor Afanoso.
  - **Secuencias aleatorias**
  - **El teorema de Fermat**
  - **Los primos pares**

[ 38 ]



## Funciones NO computables (III)

- Función de Rado:

Num. estados	Num max. 1's impresos	Cota inferior para el valor de $\sigma$
3	$\sigma(3)$	6
4	$\sigma(4)$	12
5	$\sigma(5)$	17
6	$\sigma(6)$	35
7	$\sigma(7)$	22.961
8	$\sigma(8)$	$3^{92}$
9	$\sigma(9)$	$3^{92}+1$
10	$\sigma(10)$	$((a^a)^a)\dots^a = a^a$

[ 39 ]



## Funciones NO computables (IV)

- ¿Cuáles son los problemas/funciones no computables?
- Diferentes de los “no-decidibles”:
  - **Función de Rado:** El Problema del Castor Afanoso: ¿Cuál es el número máximo de 1's que pueden ser escritos por una máquina de Turing de  $N$  estados (donde  $N$  no incluye el estado final) que termina en parada, y que comienza con una cinta inicialmente en blanco? Este número, que varía en función del número de estados de la máquina, se denota  $\Sigma(N)$ . Una máquina que produce  $\Sigma(N)$  celdas no en blanco se denomina Castor Afanoso.
  - **Secuencias aleatorias**
  - **El teorema de Fermat**
  - **Los primos pares:**

[ 40 ]



## Funciones NO computables (V)

- ¿Cuáles son los problemas/funciones no computables?
- Diferentes de los “no-decidibles”:
  - **Función de Rado:**
  - **Secuencias aleatorias**
  - **El teorema de Fermat:**  $a^n + b^n = c^n$  /  $a, b, c \neq 0$  y  $n$  entero  $> 2$
  - **Los primos pares:**

[ 41 ]



## Funciones NO computables (VI)

- ¿Cuáles son los problemas/funciones no computables?
- Diferentes de los “no-decidibles”:
  - **Función de Rado:**
  - **Secuencias aleatorias**
  - **El teorema de Fermat:**
  - **Los primos pares:** Existe un número infinito de primos  $p$  tales que  $p + 2$  también es primo. Ejs: 3 y 5 son primos pares, 11 y 13 ó 29 y 31. según los primos son más grandes la frecuencia de aparición de pares va disminuyendo, pero siguen surgiendo pares de primos gemelos aun entre números de tamaños enormes.

[ 42 ]



## Paradojas

- La **paradoja de Russell** o **paradoja del barbero**, 1901,
  - En un lejano poblado de un antiguo [emirato](#) había un barbero llamado As-Samet *diestro en afeitar cabezas y barbas, maestro en escamondar pies y en poner sanguijuelas*. Un día el emir se dio cuenta de la falta de barberos en el emirato, y ordenó que **los barberos sólo afeitaran a aquellas personas que no pudieran hacerlo por sí mismas**. Cierta día el emir llamó a As-Samet para que lo afeitara y él le contó sus angustias:
    - En mi pueblo soy el único barbero. *No puedo afeitar al barbero de mi pueblo, ¡que soy yo!, ya que si lo hago, entonces puedo afeitarme por mí mismo, por lo tanto ¡no debería afeitarme! Pero, si por el contrario no me afeito, entonces algún barbero debería afeitarme, ¡pero yo soy el único barbero de allí!*
  - El emir pensó que sus pensamientos eran tan profundos, que lo premió con la mano de la más virtuosa de sus hijas. Así, el barbero As-Samet vivió para siempre feliz y barbón.

[ 43 ]



## El problema de la Parada (I)

NO existe una Máquina de Turing que pueda decidir si una Máquina de Turing se va a parar: problema de la parada.

- Prueba por reducción al absurdo:
  - Supongamos que si existe esa MT y llamémosla **A**
  - **A** tiene codificado en su cinta (parecido a como ocurre en MTU) una MT (**MT<sub>p</sub>**) y su cinta (**C**). A la salida hará:

$$A(MT_p, C) = \begin{cases} 1 & \text{si } MT_p \text{ con } C \text{ se para} \\ 0 & \text{si } MT_p \text{ con } C \text{ no se para} \end{cases}$$

- La MT **A** cuando escribe un 1, entra en un bucle  $\infty$  (no se para)

[ 44 ]



## El problema de la Parada (II)

- $MT_p$ : es la  $p$ -ésima MT
  - La cinta  $C$  puede ser binaria, de forma que  $C$  se corresponderá con un número binario ( $q$ ).
- Supongamos que escribimos una  $MT_1$  que:
  - Recibe el número en binario  $k$ ,
  - lo duplica y
  - aplica al resultado  $A(MT_k k)$ .
- Dicha máquina será una MT determinada, por ejemplo la  $n$ -ésima.

[ 45 ]



## El problema de la Parada (III)

- ¿Qué ocurrirá con  $A(MT_n, n)$ ?:
  - Si  $A(MT_n, n)$  se para  $\rightarrow MT_n(n)$  no acaba
    - Si  $MT_n(n)$  no se para  $\rightarrow A(n, n)$  acaba
    - Si  $A(n, n)$  no se para  $\rightarrow MT_n(n)$  acaba
  - Es decir, si  $MT_n(n)$  no se para  $\rightarrow MT_n(n)$  se para, ABSURDO!
- Por otra parte:
  - Si  $A(MT_n, n)$  no se para  $\rightarrow MT_n(n)$  acaba
    - Si  $MT_n(n)$  se para  $\rightarrow A(n, n)$  no acaba
    - Si  $A(n, n)$  se para  $\rightarrow MT_n(n)$  no acaba
  - Es decir, si  $MT_n(n)$  se para  $\rightarrow MT_n(n)$  no se para, ABSURDO!
- Como hemos llegado a un absurdo, la hipótesis de partida es falsa.

[ 46 ]



## El problema de la Parada (IV)

### PRIMERO:

Proponemos la existencia de un programa que:

Dado el número de Gödel de un programa

**Programa propuesto**

Se detendrá con la variable  $x = 1$  si la entrada representa un programa autoterminante o  $x = 0$  si no es así.

...

**ENTONCES:** si existe tal programa, podemos modificarlo.

Añadiéndole una estructura mientras / fin

**Programa propuesto**

mientras X no 0, hacer;  
fin;

Para producir un programa nuevo, con número de Gödel g

[ 47 ]

## El problema de la Parada (V)

...

**SIN EMBARGO:** si ese nuevo programa no fuera autoterminante ( $X=0$ ) y

Lo arrancamos con entrada g

La ejecución llegaría a este punto con  $X=0$

Pasaría por alto esta parte del Programa

**Programa propuesto**  
mientras X no 0 hacer;  
Fin;

y se detendrá la ejecución

Es decir, si el nuevo programa no es autoterminante, si es autoterminante.

**AHORA BIEN:** si ese nuevo programa fuera autoterminante ( $X=1$ ) y

Lo arrancamos con entrada g

La ejecución llegaría a este punto con  $X=1$

Y la ejecución quedaría atrapada en este bucle eternamente

**Programa propuesto**  
mientras X no 0 hacer;  
fin

es decir, si el nuevo programa es autoterminante, no es autoterminante

[ 48 ]

## El problema de la Parada (V)

...

...

### En CONSECUENCIA:

La existencia del  
Programa  
propuesto

la existencia de un  
nuevo programa

**Programa  
propuesto**  
autoterminante

Conduciría a

**Programa  
propuesto**  
mientras X no  
0 hacer;  
fin

que no es

ni no  
autoterminante.

Así que la existencia del  
programa propuesto es imposible.

49

## 7. Máquinas de Turing

Grado Ingeniería Informática  
Teoría de Autómatas y Lenguajes Formales





## **Parte VIII**

### **TEMA 8. Complejidad Computacional**



# Tema 8. Complejidad Computacional.



## Tema 8. Complejidad Computacional

### Índice:

- Autómatas, Complejidad y Computabilidad.
- Clasificación de los Problemas de Decisión.



## Autómatas, Complejidad y Computabilidad

- En la **Teoría de la Computación**, los tres siguientes áreas:
  - Autómata,
  - Complejidad y
  - Computación
 están relacionados por la siguiente pregunta:
  - *¿Cuáles son las capacidades y limitaciones de los ordenadores?*
- Sin embargo, esta pregunta se interpreta de forma diferente en cada una de las 3 áreas.



## Autómatas, Complejidad y Computabilidad

### Teoría de Autómatas:

- Se encarga de las definiciones y propiedades de los modelos matemáticos de computación (esenciales en áreas aplicadas de la informática).
- Uno de estos modelos son los **Autómatas Finitos**, utilizados en:
  - Procesamiento de textos
  - Compiladores
  - Diseño Hardware.
- Otro modelo son las **Gramáticas Libres de Contexto**, usadas en:
  - Lenguajes de programación
  - Inteligencia Artificial.



## Autómatas, Complejidad y Computabilidad

### Teoría de la Complejidad:

- Se basa en tratar de dar respuesta a la siguiente pregunta:
  - *¿Qué hace a algunos problemas computacionalmente difíciles y a otros sencillos?*
- Tiene como finalidad la creación de mecanismos y herramientas capaces de describir y analizar la complejidad de un algoritmo y la complejidad intrínseca de un problema.



## Autómatas, Complejidad y Computabilidad

### Teoría de la Computabilidad:

- Está muy relacionado con la teoría de la Complejidad, ya que introduce varios de los conceptos que esta área utiliza.
- Su finalidad principal es la clasificación de diferentes problemas, así como formalizar el concepto de *computar*.
- Así, estudia qué lenguajes son *decidibles* con diferentes tipos de “*máquinas*” y diferentes modelos formales de computación.



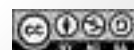
## Complejidad Computacional

- Estudia el orden de complejidad de un algoritmo que resuelve un problema *decidable*.
- Para ello, considera los 2 tipos de recursos requeridos durante el cómputo para resolver un problema:
  - **Tiempo:** Número de pasos base de ejecución de un algoritmo para resolver un problema.
  - **Espacio:** Cantidad de memoria utilizada para resolver un problema.



## Complejidad Computacional

- La complejidad de un algoritmo se expresa como función del tamaño de la entrada del problema,  $n$ .
- Se refiere al ratio de crecimiento de los recursos con respecto a  $n$ :
  - Ratio del Tiempo de ejecución (Temporal):  $T(n)$ .
  - Ratio del espacio de almacenamiento necesario (Espacial):  $S(n)$ .



## Clasificación de Problemas de decisión

- En base a dos criterios:
  - *Teoría de la Computabilidad*
    - *Decidible.*
    - *Parcialmente Decidible (reconocible).*
    - *No Decidible.*
  - *Teoría de la Complejidad Computacional*
    - *Conjuntos de Clase de Complejidad (Clase L, NL, P, P-Completo, NP, NP-Completo, NP-Duro...).*

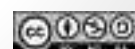
*Un problema de decisión es aquel en el que las respuestas posibles son Si o No*



## Clasificación de Problemas de decisión

Considerando la **Teoría de la Computabilidad** un problema de decisión podrá ser:

- **Decidible** (o resoluble algorítmicamente):
  - Si existe un procedimiento mecánico (MT) que lo resuelva.
  - Además, la MT debe detenerse para cualquier entrada.
- **Parcialmente Decidible** (Reconocible):
  - Si existe un procedimiento mecánico (MT) que lo resuelva.
  - Además, la MT debe detenerse para aquellas entradas que son una solución correcta al problema.
- **No Decidible**
  - Si NO es decidible





## Clasificación de Problemas de decisión

Considerando la **Teoría de la Complejidad Computacional** un problema de decisión podrá ser:

- **Conjuntos de Clase de Complejidad** (Clase L, NL, P, P-Completo, NP, NP-Completo, NP-Duro...).

En este caso, nos

centraremos únicamente en los problemas denominados Clase P, NP y NP-Completo.

Sin embargo, para esta distinción es necesario considerar el modelo teórico de las **Máquinas de Turing**.

Además, debemos distinguir entre:

- **MT Determinista** (Para cada par (*estado*, *símbolo*), existe como máximo una transición a otro estado).
- **MT No Determinista** (Existe al menos un par (*estado*, *símbolo*), con más de una transición a estados diferentes).



## Clasificación de Problemas de decisión

Considerando la **Teoría de la Complejidad Computacional** un problema de decisión podrá ser:

- **Clase P** (Polynomial-time)
  - Contiene aquellos problemas de decisión que una **MT Determinista** puede resolver en **tiempo polinómico**.
    - Los problemas de complejidad polinómica son tratables, es decir en la práctica se pueden resolver en tiempo *razonable*.
    - La mayoría de los problemas *corrientes* (ordenación, búsqueda...) pertenecen a esta clase.



## Clasificación de Problemas de decisión

Considerando la **Teoría de la Complejidad Computacional** un problema de decisión podrá ser:

- **Clase NP** (Non-Deterministic Polynomial-time)
  - Contiene aquellos problemas de decisión que una **MT No Determinista** puede resolver en **tiempo polinómico**.

Como toda MTD es un caso particular de una MTND:  
 $P \subseteq NP$

Clase NP

Clase P

Saber si  $P=NP$  o  $P \neq NP$  es todavía un problema abierto en computación teórica!!  
 Tan importante es demostrar que estas clases son distintas, que es uno de los *problemas* premiados con 1.000.000 \$.



## Clasificación de Problemas de decisión

Considerando la **Teoría de la Complejidad Computacional** un problema de decisión podrá ser:

- **Clase NP-Completo**
  - Un problemas de decisión es NP-Completo sii:
    - Es NP
    - Todos los demás problemas de NP se pueden se pueden reducir a él en tiempo polinómico.

Reducir de un problema:

Es una manera de convertir un problema en otro de tal forma que la solución al segundo problema se puede utilizar para resolver el primero.



# Tema 8. Complejidad Computacional.



# 1. RECURSOS

## Limpiar y bien formar siempre

### 1.1. Tema 3

#### 1.1.1. $AFD \rightarrow AFD \text{ mínimo}$

1. Buscar  $Q/E_0$ , que divide los estados en finales y no finales.
2. Hacer  $Q/E_1, Q/E_2 \dots$  pueden separarse, pero nunca juntarse de nuevo.
3. Cuando se repitan las particiones hemos terminado. Como máximo tendremos que hacer  $Q/E_{(n-2)}$  iteraciones.

#### 1.1.2. $AFND \rightarrow AFD$

1. Calcular  $T^*$ .
2. Quitar la columna  $\lambda$  y añadir la columna  $\lambda^*$ .
3. Sustituir la columna a, b, ... por  $\lambda^*a\lambda^*, \lambda^*b\lambda^* \dots$
4. El nuevo estado inicial será  $p \lambda^*$ .
5. Transformar los caminos múltiples en estados combinados (finales si alguna de las letras es final) y las transiciones no definidas en transiciones al sumidero.

### 1.2. Tema 4

#### 1.2.1. $G3 LD \rightarrow G3 LI$

1. Quitar el axioma inducido ( $A \rightarrow aS$ ) introduciendo un nuevo símbolo (que haga lo mismo que el axioma, pero no se copia lambda)
2. Construir un grafo dirigido en el que los nodos son los  $\Sigma_{NT}$  y las flechas son los  $\Sigma_T$ .
3. Intercambiar las etiquetas de  $\lambda$  y S, y dar la vuelta a las flechas.
4. Interpretar el grafo

Nota: las que iban de S van a  $\lambda$ , y de  $\lambda$  no puede salir nada.

### 1.2.2. Lenguaje vacío (G2)

Generar el árbol de derivación hasta llegar a  $n$  (número de estados). Si no genera sentencias y se repiten los  $\Sigma_{NT}$ , es un lenguaje vacío.

### 1.2.3. Lenguaje infinito (G2)

Construir un grafo cuyos nodos están etiquetados con los  $\Sigma_{NT}$ . Si existen ciclos accesibles desde el axioma, entonces es un lenguaje infinito.

### 1.2.4. Limpieza y bien-formación de gramáticas

1. Eliminar reglas innecesarias ( $A \rightarrow A$ )
2. Eliminar símbolos inaccesibles (para ello se construye un vector con los símbolos T y NT)  
Ir marcando desde el axioma los que vaya produciendo.
3. Eliminar reglas superfluas con el algoritmo de marcado
  - a) Se marcan los  $\Sigma_{NT} \rightarrow \Sigma_T$  y los  $\Sigma_{NT} \rightarrow \lambda$
  - b) Se marcan los que contengan un  $\Sigma_{NT}$  marcado en la derecha
  - c) Se repite hasta que no se pueden marcar más
  - d) Se eliminan todas las reglas no marcadas
4. Eliminar los símbolos no generativos, es decir, aquellos que solo aparecen en reglas superfluas.
5. Eliminar las reglas no generativas, las de tipo  $A \rightarrow \lambda$ . Cada vez que aparezca A en la  $\lambda$ . Se admite parte derecha, se añade la posibilidad de que sea *Axioma*  $\rightarrow \lambda$ , OJO si cuando eliminamos solo quedaba un símbolo ( $C \rightarrow M$  y eliminamos M), ponemos lambda ( $C \rightarrow \lambda$ ) y repetimos el proceso de eliminación (para C).
6. Eliminar las reglas de red denominación, las de tipo  $A \rightarrow B$ . Por cada regla de la forma  $B \rightarrow x$ , se añade  $A \rightarrow x$

### 1.2.5. $G2 \rightarrow FNC$

Se separa el primer símbolo de la derecha del resto, por ejemplo, de  $A \rightarrow aBb$  sacamos  $A \rightarrow DE$  y  $D \rightarrow a$ ,  $E \rightarrow Bb \rightarrow BC$ ,  $C \rightarrow b$  (previo paso debe estar limpia y bien formada)

### 1.2.6. $G2 \rightarrow FNG$

1. Limpiar y bien formar la gramática.
2. Quitar recursividad a izquierdas si las reglas.  $\lambda$  no se toca en este paso
3. Ordenar el alfabeto  $\Sigma_{NT}$  (A, B) y clasificar las reglas en  $G2(AB)$  o  $G3(BA)$
4. Pasar las de  $G3$  a  $G2$ . Se hace por sustitución, no sustituir con las reglas que dan  $\lambda$ 
  - a) Quitar recursividad a izquierdas si apareciese.
5. Pasar de  $G2$  a  $G1$ , empezando por la que me deje meter un  $\Sigma_T$  en la cabeza.
6. Si hay un  $\Sigma_T$  que no esté en la cabeza, sustituirlo por un  $\Sigma_{NT}$  que dé ese  $\Sigma_T$ .

### 1.2.7. Quitar recursividad a izquierdas

Dada una regla de tipo  $A \rightarrow A\alpha \mid \beta$  (donde  $\alpha$  y  $\beta$  son cualquier cosa...)

Se transforma en:

- $A \rightarrow \beta \mid \beta X$
- $X \rightarrow \alpha X \mid \alpha$

Si tuviéramos varias ( $A \rightarrow A\alpha \mid \beta_1 \mid \beta_2$ ), entonces se transforma en:

- $A \rightarrow \beta \mid \beta X$
- $X \rightarrow \alpha X \mid \alpha$

### 1.2.8. Paso de $G3LD$ (FNG) $\rightarrow$ AF y viceversa

Por cada regla  $A \rightarrow aB$ :

- A y B son estados del autómata y se realiza la transición de A a B con 'a'.

Si tenemos una regla del tipo  $A \rightarrow a$ :

- Se realiza la transición de A a un estado final con 'a'.

Para pasar de AF a  $G3LD$  se hace exactamente igual.

### 1.3. Tema 5

#### 1.3.1. Teoría de síntesis

Nota:  $D_{ab}(\alpha) = D_b(D_a(\alpha))$

1. Derivar la expresión respecto de todos los símbolos y todas las que me vayan saliendo.

Las voy llamando  $R_x$ , donde x es un Número

2. Si  $R_x$  puede ser  $\lambda$ , se añade una regla,  $R_x \rightarrow \lambda$
3. Si  $D_y(R_{x_1}) = R_{x_2}$ , se añade una regla  $R_{x_1} \rightarrow yR_{x_2}$
4. Luego se aplica el paso de G3LD (FNG) a AF para obtener el AF correspondiente

$$D_a(a) = \lambda$$

$$D_a(b) = \emptyset$$

$$D_a(RS) = D_a(R)S + d(R)D_a(S)$$

$$D_a(R + S) = D_a(R) + D_a(S)$$

$$D_a(R^*) = D_a(R)R^*$$

$$d(a) = \emptyset$$

$$d(a^*) = \lambda$$

$$d(a^* + a) = \lambda$$

Si puede ser  $\lambda$ , es  $\lambda$ , si no  $\emptyset$

### 1.4. Tema 6

#### 1.4.1. $APF \rightarrow APV$

1. Añadir un estado inicial nuevo con una transición y un nuevo «chivato» que nos indique cuando se vacía la pila
2. Añadir un estado «final» que desapile todo lo que pudiera quedar y el chivato.

#### 1.4.2. $APV \rightarrow APF$

1. Añadir un estado inicial nuevo con su chivato
2. Añadir un estado final al que se transita desapilando el chivato

### 1.4.3. $G2 (FNG) \rightarrow APV$

1. Tres tipos de reglas:

a)  $A \rightarrow aBCD : f(q, a, A) = (q, BCD)$

b)  $A \rightarrow a : f(q, a, A) = (q, \lambda)$

c)  $S \rightarrow \lambda : f(q, \lambda, S) = (q, \lambda)$

2. El automata tendrá un único estado, q.

### 1.4.4. $APV \rightarrow G2$

1. Se pone una regla del tipo  $S \rightarrow (q_0, A_0, pqr...)$

2. Se ponen reglas de tipo:

a) Tipo 1A :  $f(q, a, B) = (p, DEF)$

El molde será:  $(qB\_ ) \rightarrow a(pD\_ )(\_E\_ )(\_F\_ )$

b) Tipo 1B :  $f(p, \lambda, B) = (q, A)$

El molde será:  $(pB\_ ) \rightarrow X(qA\_ )$

c) Tipo 2A :  $f(p, a, B) = (q, \lambda)$

El molde será:  $(pBq) \rightarrow a$

d) Tipo 2B :  $f(p, \lambda, B) = (q, \lambda)$

El molde será:  $(pBq) \rightarrow \lambda$



### 1.4.5. Equivalencia de EERR

1.  $(\alpha + \beta) + \sigma = \alpha + (\beta + \sigma)$
2.  $\alpha + \beta = \beta + \alpha$
3.  $(\alpha \cdot \beta) \cdot \sigma = \alpha \cdot (\beta \cdot \sigma)$
4.  $\alpha \cdot (\beta + \sigma) = (\alpha \cdot \beta) + (\alpha \cdot \sigma)$   
 $(\beta + \sigma) \cdot \alpha = (\beta \cdot \alpha) + (\sigma \cdot \alpha)$
5.  $\alpha \cdot \lambda = \lambda \cdot \alpha = \alpha$
6.  $\alpha + \phi = \phi + \alpha = \alpha$
7.  $\lambda^* = \lambda$
8.  $\alpha \cdot \phi = \phi \cdot \alpha = \phi$
9.  $\phi^* = \lambda$
10.  $\alpha^* \cdot \alpha^* = \alpha^*$
11.  $\alpha \cdot \alpha^* = \alpha^* \cdot \alpha$
12.  $(\alpha^*)^* = \alpha^*$
13.  $\alpha^* = \lambda + \alpha + \alpha^2 + \dots + \alpha^n + \alpha^{n+1} \cdot \alpha^*$
14.  $\alpha^* = \lambda + \alpha \cdot \alpha^*$
15.  $\alpha^* = (\lambda + \alpha)n - 1 + \alpha n \cdot \alpha^*$
16. Sea  $f$  una función,  $f : E_{\Sigma^n} \rightarrow E_{\Sigma}$  se verifica:  
 $f(\alpha, \beta, \dots, \sigma) + (\alpha + \beta + \dots + \sigma)^* = (\alpha + \beta + \dots + \sigma)^*$
17. Sea  $f$  una función,  $f : E_{\Sigma^n} \rightarrow E_{\Sigma}$  se verifica:  
 $(f(\alpha^*, \beta^*, \dots, \sigma^*))^* = (\alpha + \beta + \dots + \sigma)^*$
18.  $(\alpha^* + \beta^*)^* = (\alpha^* \cdot \beta^*)^* = (\alpha + \beta)^*$
19.  $(\alpha \cdot \beta)^* \cdot \alpha = \alpha \cdot (\beta \cdot \alpha)^*$
20.  $(\alpha^* \cdot \beta)^* \cdot \alpha^* = (\alpha + \beta)^*$
21.  $(\alpha^* \cdot \beta)^* \cdot \alpha^* = \lambda + (\alpha + \beta)^* \cdot \beta$
22. R. Inferencia:  $X = Ax + B \rightarrow X = A^* \cdot B$

### 1.4.6. Analisis

1. Hacer las ecuaciones del Automata Finito

De  $X_0$  a  $X_1$  con una 'a':  $X_0 = aX_1$

De  $X_0$  a  $X_2$  que es final con una 'b':  $X_0 = bX_2 + b$

Si hay varias transiciones se ponen en la misma ecuación sumándose

Si el estado final solo va al sumidero o no tiene ramas se le añade  $\lambda$

2. Utilizar las equivalencias de EERR, empezando por las más lejanas al inicial. Esencialmente se usa la regla de inferencia

$$\begin{array}{ll} X_0 = aX_0 & X_0 = \emptyset \\ X_1 = bX_1 + c & X_1 = b^*c \\ X_2 = c & X_2 = c \end{array}$$

### 1.4.7. Formatos

AFD = (Alfabeto,  $Q$ ,  $q_0$ ,  $f$ ,  $F$ )  $F$  = Estado finales  $f$ =Función transición

AFND = (Alfabeto,  $Q$ ,  $q_0$ ,  $f$ ,  $F$ ,  $T$ )  $T$  = Transiciones con  $\lambda$

$G$  = (Terminales, No Terminales,  $S$ ,  $P$ )  $S$  = Axioma  $P$  = Transiciones

AP = (Alfabeto cinta, Alfabeto pila,  $Q$ ,  $A_0$ ,  $q_0$ ,  $f$ ,  $F$ )  $A_0$  = Fondo pila

MT = (Alfabeto de entrada, Alfabeto cinta,  $b$ ,  $Q$ ,  $q_0$ ,  $f$ ,  $F$ )  $b$  = Símbolos especiales

### 1.4.8. Jeraquia de Chomsky

Todos aceptan axioma para dar  $\lambda$

**G0** Lenguaje sin restricciones, puede ser cualquier cosa, se caracteriza por reglas compresoras ( $aVs \rightarrow d$  OJO también si no es axioma  $B \rightarrow \lambda$ ) y estructura de frases ( $AS \rightarrow SA$ )

**G1** Sensible al contexto, puede ser cualquier cosa, sin reglas compresoras y Contexto ( $aS \rightarrow ADC$ )

**G2** De contexto libre, un símbolo a la izquierda pero cualquier cosa a la derecha. También si hay G3LD y G3LI en la misma gramática.

**G3** Gramática regular, son las de la forma  $NT \rightarrow T$  y un tipo de las siguientes reglas:

G3LI  $NT \rightarrow NT T$

G3LD  $NT \rightarrow T NT$