Modelos de computación (2015-2016)
Grado en Ingeniería Informática
Universidad de Granada

Memoria Prácticas

Francisco Javier Navarro Morales

6 de noviembre de 2015

Índice

1. Practica 1: G = (V,T,P,S), donde V = S,A,B, T = a,b, el símbolo de partida es S y las reglas son:

$$\begin{array}{cccc} \text{S} \rightarrow \text{aB}, & \text{S} \rightarrow \text{bA}, & \text{A} \rightarrow \text{a}, & \text{A} \rightarrow \text{aS}, \\ \text{A} \rightarrow \text{bAA}, & \text{B} \rightarrow \text{b}, & \text{B} \rightarrow \text{bS}, & \text{B} \rightarrow \text{aBB} \end{array}$$

Esta gramática genera el lenguaje: $L(G) = \{u | u \, \epsilon \, \{a, b\}^+ \, y \, Na(u) = Nb(u)\}$, es decir, la gramática genera palabras con el mismo número de 'a' que de 'b'.

Podemos extraer las siguientes interpretaciones de las reglas de producción:

- Interpretación de **'A'** → Genera palabras con un Símbolo Terminal 'a' de más.
- Interpretación de **'B'** → Genera palabras con un Símbolo Terminal 'b' de más.
- Interpretación de **'S'** → Genera una cadena con el mismo numero de 'a' que 'b'.

$\overset{w=\lambda v}{\longrightarrow}$

Hay que demostrar dos cosas:

- Todas las palabras generadas por la gramática tienen el mismo número de a que de b.
- Cualquier palabra con el mismo número de a que de b es generada.

Vamos a ir desarrollando cada posibilidad de forma que para cada paso se apliquen todas las reglas de producción posibles, inicialmente tenemos dos posibilidades:

$$^1\text{S} \Rightarrow \text{aB}, \qquad ^2\text{S} \Rightarrow \text{bA}$$

Vamos a iniciar el desarrollo para la primera:

S ⇒ aB ⇒ ab, generamos la palabra **ab**.
S ⇒ aB ⇒ abS ⇒ abaB ⇒ abab, generamos la palabra **abab**.
S ⇒ aB ⇒ aaBB ⇒ aabB ⇒ aabb, generamos la palabra **aabb**.

Continuamos el desarrollo para la segunda posibilidad:

S ⇒ bA ⇒ ba, generamos la palabra **ba**.
S ⇒ bA ⇒ baS ⇒ babA ⇒ baba, generamos la palabra **baba**.
S ⇒ bA ⇒ bbAA ⇒ bbaA ⇒ bbaa, generamos la palabra **bbaa**.

Hemos comprobado que podemos conseguir generar cadenas básicas con $N_a(\text{u}) = N_b(u)$, dónde primero hay símbolos terminales 'a' y luego 'b' ó hay símbolos terminales '$(\text{ab})^+$', y lo mismo cambiando el orden de 'a' y 'b'. Si queremos generar cualquier cadena perteneciente al lenguaje lo podemos conseguir combinando las anteriores palabras generadas. Por ejemplo generemos una palabra usando todas las reglas de producción:

$$S \overset{1}{\Rightarrow} aB \overset{8}{\Rightarrow} aaBB \overset{7}{\Rightarrow} aabSB \overset{1}{\Rightarrow} aabaBB \overset{8}{\Rightarrow} aabaaBBB \overset{7}{\Rightarrow} aabaabSBB \overset{2}{\Rightarrow} aabaabbABB \overset{5}{\Rightarrow} aabaabbbAABB \overset{3}{\Rightarrow} aabaabbbaABB \overset{4}{\Rightarrow} aabaabbbaaSBB \overset{2}{\Rightarrow} aabaabbbaabABB \overset{3}{\Rightarrow} aabaabbbaabaBB \overset{6}{\Rightarrow} aabaabbbaababB \overset{6}{\Rightarrow} aabaabbbaababb$$

Podemos apreciar que encima de cada fecha indicamos el número de la regla utilizada (las identificamos contando de izquierda a derecha y de arriba hacia abajo) en este ejemplo hemos usado las 8 reglas de producción que nos brinda la gramática, comprobando que la palabra que genera **'aabaabbbaababb'** contiene el mismo número de símbolos terminales 'a' que 'b', en concreto N = 7. Con este ejemplo demostramos también que combinando las reglas de producción podemos generar cualquier palabra con la peculiaridad de que el número de 'a' coincide con el de 'b'.

2. Practica 2: Determinar si la gramática G = ({S, A, B}, {a, b, c, d}, P, S) genera un lenguaje de tipo 3 donde P es el conjunto de reglas de producción:

$$S \to AB, \qquad A \to Ab, \qquad A \to a, \qquad B \to cB, \qquad B \to d$$

Gramática libre del contexto, es decir, de tipo 2: Los símbolos terminales dependen entre ellos y alguna regla de producción es del tipo: $A \to u$; $A \to U$.
Esta **gramática genera el lenguaje** $L(G) = \{ab^i c^j d : i, j \ \epsilon \ \mathbb{N}\}$

Primero, vamos a **demostrar que a partir de las reglas de producción de tipo 2 se genera el lenguaje L(G)**.

Comenzamos con el símbolo de partida "S":

$$S \to \underline{A}B^1 \to \underline{A}bB^2 \to \underline{A}bbB^2 \to ab^i\underline{B}^3 \to ab^ic\underline{B}^4 \to ab^icc\underline{B}^4 \to ab^ic^j\underline{B}^4 \to ab^ic^jd^5$$

Notas sup-indices:
1. Como el lenguaje L(G) comienza por un símbolo terminal "a" seguidos por i veces el símbolo terminal "b", aplicamos la regla de producción $A \to Ab$, ya que a partir de la variable "A" podemos generar i veces el símbolo terminal "b" y generar el símbolo terminal "a" de la primera posición cuando se desee.
2. Para generar el símbolo terminal "b" i veces producimos la regla de producción $A \to Ab$ y generamos b^i veces el símbolo terminal "b".
3. Como el lenguaje L(G) comienza por el símbolo terminal "a" y ya disponemos del símbolo terminal b^i, aplicamos la regla de producción $A \to a$ para obtener el primer símbolo terminal del lenguaje L(G).
4. El lenguaje L(G) dispone de j veces el símbolo terminal "c" para ello vamos aplicando

la regla de producción $B \rightarrow cB$ sobre la variable "B" para generar los c^j símbolos terminales.
5. El lenguaje L(G) termina con el símbolo terminal "d", por tanto, aplicamos la regla de producción $B \rightarrow d$ sobre la variable "B" para obtener el símbolo terminal "d".

Como podemos observar con las reglas de producción de tipo 2, el lenguaje L(G) se puede generar y como patrón podemos sacar que para obtener los b^i símbolos terminales vamos aplicando la regla de producción sobre la variable "A" y para obtener los símbolos terminales c^j aplicamos la regla de producción sobre la variable "B".

Segundo, para **generar un lenguaje de tipo 3 (Regular)**: Los símbolos terminales no dependen entre ellos y las reglas de producción tienen que ser del tipo 3, ya que si las reglas de producción son del tipo 3 el lenguaje generado es de tipo 3.

Las reglas de producción de tipo 3 contienen 1 solo símbolo terminal a la izquierda o son del tipo: $A \rightarrow uB$; $A \rightarrow u$; $A \rightarrow B$.

Por tanto, para generar el lenguaje de tipo 3 debemos crear las siguientes reglas de producción de tipo 3:

- $S \rightarrow aB$: Como símbolo de partida comenzamos con la variable "S", por tanto en el primer paso producimos el símbolo terminal "a" primero que se necesita y con la variable "B" vamos obteniendo los b^i símbolos terminales.
- $B \rightarrow bB$: Con la variable "B" obtenemos los b^i símbolos terminales.
- $B \rightarrow C$: Una vez obtenidos los b^i símbolos terminales deseados para obtener los c^j símbolos terminales, producimos un cambio de variable para generar la producción de los símbolos terminales "c".
- $C \rightarrow cC$: Con la variable "C" generamos la producción de los c^j símbolos terminales deseados.
- $C \rightarrow d$: Para acabar con la generación del lenguaje necesitamos generar a partir de la variable "C" un símbolo terminal "d".

Por tanto, con estas reglas de producción de tipo 3 generamos una gramática de tipo 3, tal y como indica el lenguaje L(G).

3. Practica 3: Diseñar un autómata finito determinístico que acepte cadenas que contienen las subcadenas '0110' y '1000'. Las subcadenas pueden aparecer juntas o separadas, también pueden contener más símbolos (0+1) delante o detras de ellas.

Para comenzar esta práctica lo más sencillo es **diseñar el autómata no determinístico** que acepta las cadenas que nos pide el ejercicio. Nos ayudamos de los ejemplos de los autómatas que son capaces de reconocer palaras que contienen las subcadenas '0110' ó '1000, de esta forma conseguimos diseñar el autómata que necesitamos de forma muy fácil, para ello **seguimos los siguientes pasos**:

- Q4 deja de ser un nodo final y P0 deja de ser un nodo inicial.
- Unimos los dos autómatas añadiendo una transición nula entre Q4 y P0.

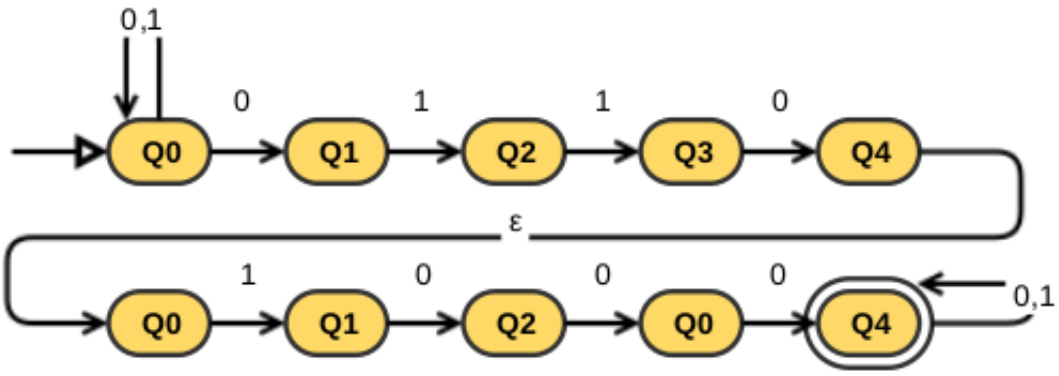

Figura 3.1: NonDeterministic finite automaton(NFA)

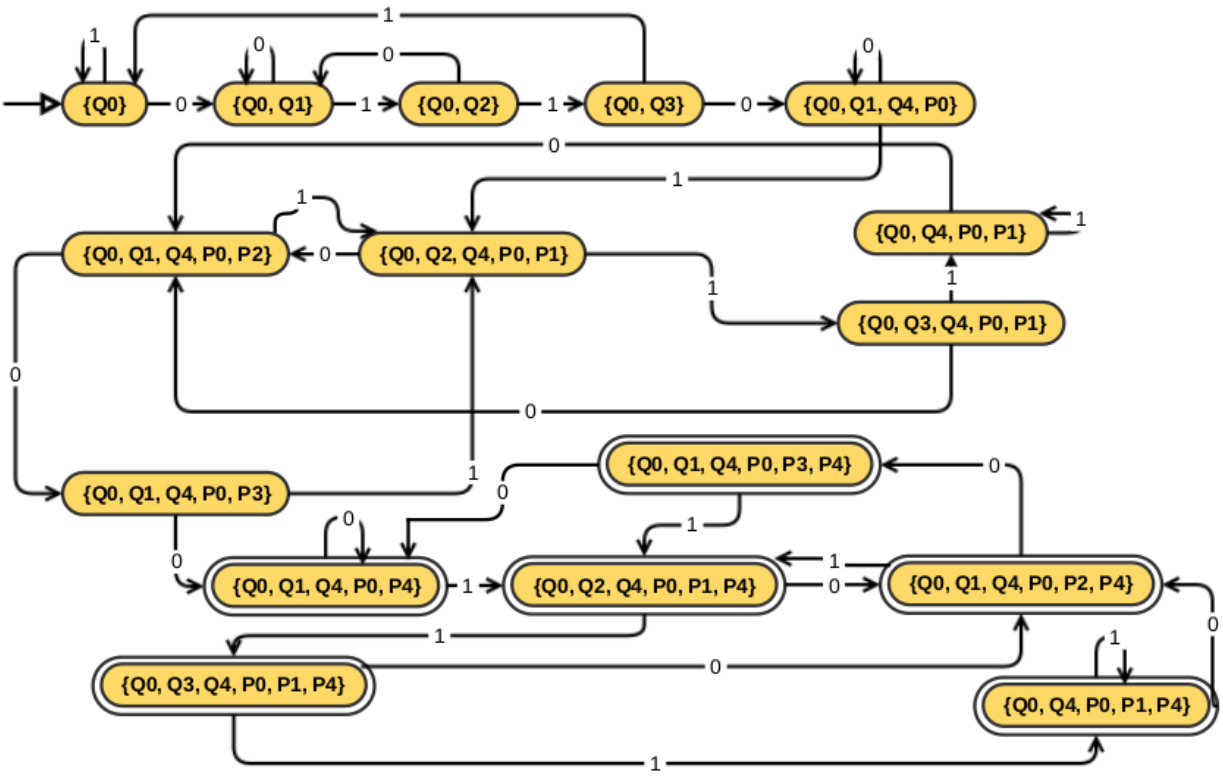

Figura 3.2: Deterministic finite automaton(DFA)