Modelos de Computación

David Gil Bautista Grupo C2 - José A. García

Índice

Práctica 1 - Gramática

Determinar si una gramática G con su conjunto de producciones genera un lenguaje de tipo regular.

Para G = ({S,A,B}, {a,b,c,d}, P, S) donde P es el conjunto de reglas de producción con las siguientes reglas:

S → AB
B → cB
A → Ab
B → d
A → a

Comprobar si genera un lenguaje de tipo 3.

Solución

Para comenzar deberemos obtener el lenguaje que genera dicha gramática y para ello sustituiremos la producción inicial con las reglas que tiene la gramática.

A S → AB le podemos aplicar las producciones 5 y 3 para obtener lo siguiente:

S → AB → aB
S → AB → AbB

Tras conseguir esto podríamos seguir aplicando producciones para desarrollar los lenguajes anteriores.

Con el primer lenguaje:

S → AB → aB; aB → acB
aB → ad

Con el segundo:

S → AB → AbB; AbB → AbcB
AbB → Abd
AbB → abB
AbB → AbB

La variable A siembre será sustituida por la constante ‘**a**’ o por ‘**Ab**’, por lo que cada vez que aparezca una A solo se sustituirá por una ‘**a**’ u otra A acompañada de una constante ‘**b**’. Lo que significa que tendremos una sola ‘**a**’ acompañada de una o más ‘**b**’. ’

Comprobamos ahora lo obtenido al aplicar las reglas de producción de B:

Con el primer caso:
S → AB → aB; aB → acB → accB
aB → acB → acd
Con el segundo caso:

S → AB → AbB; AbB → AbcB → AbccB
AbB → AbcB → Abcd
AbB → AbbB → AbbcB
AbB → AbbB → Abbd

Para el caso de la B se da el mismo caso, pues podrán aparecer varias ‘**c**’ y al final una única ‘**d**

En conclusión podemos ver que se obtiene un lenguaje del tipo ***ab...bc...cd***, es decir:

L = { ab^nc^md donde ∃ n,m ∈ ℕ tal que n >= 0, m >= 0 }

Ahora deberemos generar una gramática de tipo 3 que pueda generar este lenguaje.
.
Comenzaremos con un una variable inicial que produzca la primera constante y otra variable.

S → aS1

Una vez tengamos la primera constante se deben generar varias ‘***b***’.

S1 → bS1

Por último añadiremos las ‘**c**’ y la última ‘**d**’.

S1 → S2
S2 → cS2
S2 → d

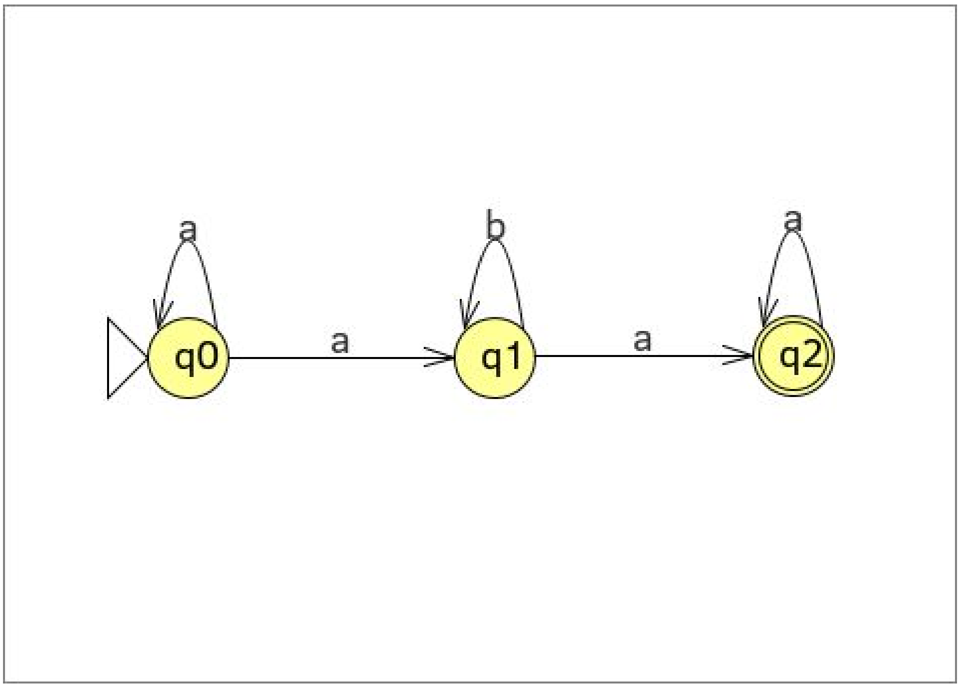
Por lo que nos queda una gramática de tipo 3 con las siguientes reglas de producción.

S → aS1 S1 → bS1 S2 → cS2
S1 → S2 S2 → d

Práctica 2 - JFLAP

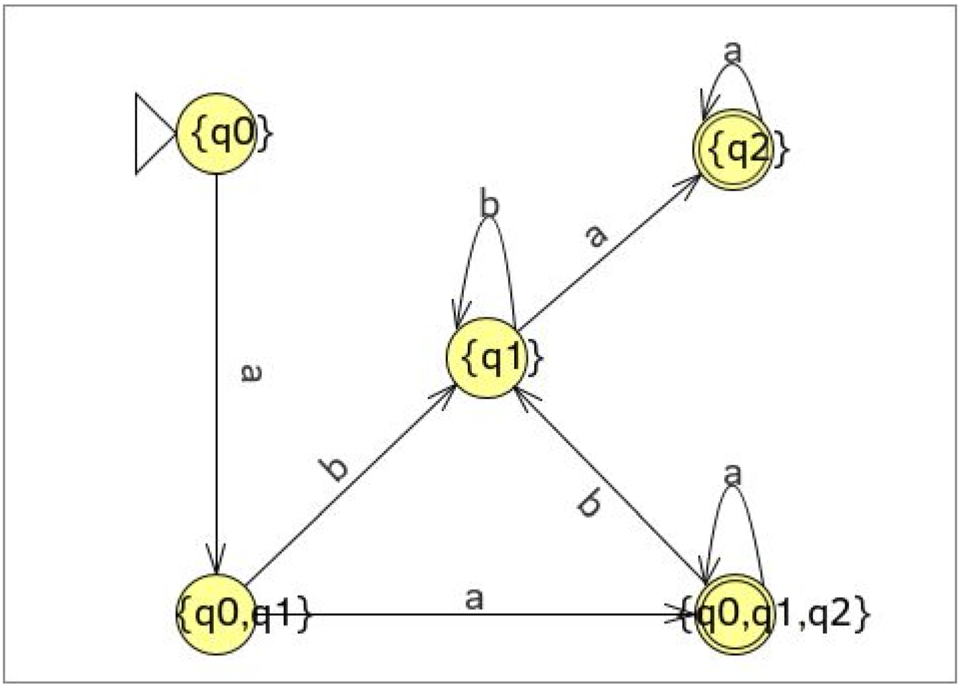
Implemente con JFLAP un autómata determinista aportándole a la herramienta un autómata no determinista y uno de transiciones nulas.

Autómata no gordito No Determinista (**AFND**):



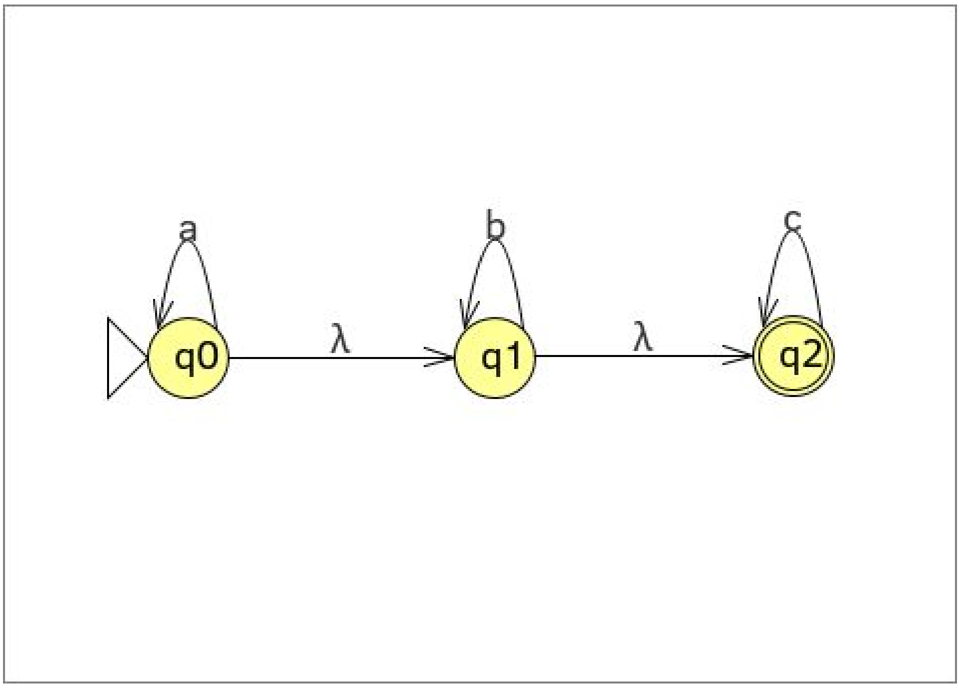
1. Autómata Finito No Determinista

A partir del **AFND** de la imagen 1 obtenemos el siguiente **AFD** usando JFLAP:



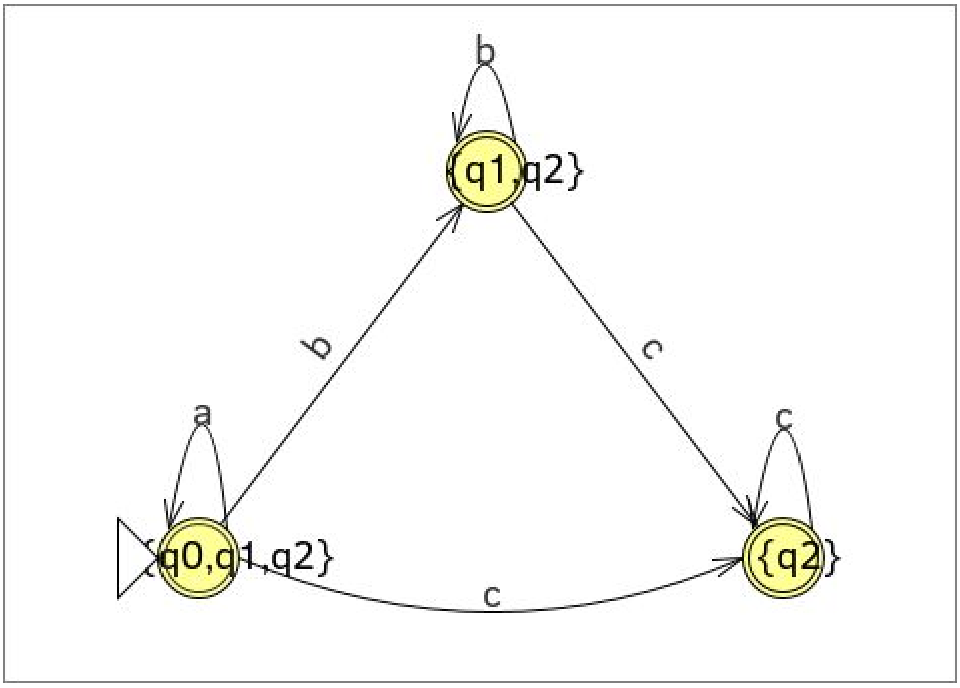
2. Autómata Finito Determinista

Para la conversión del autómata finito con transiciones nulas he utilizado el siguiente:



3. Autómata Finito con transiciones nulas

Y a partir de este autómata, tras usar JFLAP, obtenemos el siguiente:



4. Autómata Finito Determinista

Como podemos comprobar al comienzo, ambos autómatas son prácticamente iguales diferenciados por las transiciones. Al convertirlos a **AFD** obtenemos dos resultados completamente distintos donde el Autómata No Gordito con Transiciones Nulas tiene dos estados menos.

Práctica 3 - LEX

Usando LEX crear un programa que lea un archivo de texto y produzca una salida.
He sido incapaz de compilar lex en mi sistema operativo y debido a la poca documentación hallada he desistido tristemente.

Práctica 4 - Autómata con pila

Obtener la gramática libre de contexto de un autómata con pila vacía.

Partiremos del autómata con pila vacía que aparece en el último ejemplo de las diapositivas del tema 5.

M = ({q0,q1}, {0,1}, {X,Z}, δ, q0, Z0, 0),

donde

δ(q0,0,Z0) = {(q0,XZ0)}
δ(q0,0,X) = {(q0,XX)}
δ(q0,1,X) = {(q1,ε)}

δ(q1,1,X) = {(q1,ε)}
δ(q1,ε,X) = {(q1,ε)}
δ(q1,ε,Z0) = {(q1,ε)}

S → [q0, Z0, q0], S → [q0, Z0, q1]

[q1,Z0,q1] → ε

Una vez tenemos las reglas de producción deberemos ver como desarrollarlas para comprobar la interacción que tienen con la pila.

- δ(q0,0,Z0) = {(q0,XZ0)}
    - [q0,Z0,q0] → 0[q0,X,q0] [q0,Z0,q0]
    - [q0,Z0,q1] → 0[q0,X,q0] [q0,Z0,q1]
    - [q0,Z0,q0] → 0[q0,X,q1] [q1,Z0,q0]
    - [q0,Z0,q1] → 0[q0,X,q1] [q1,Z0,q1]
    -

- δ(q1,1,X) = {(q1,ε)}
    - [q1,X,q1] → 1

- δ(q0,0,X) = {(q0,XX)}
    - [q0,X,q0] → 0[q0,X,q0] [q0,X,q0]
    - [q0,X,q1] → 0[q0,X,q0] [q0,X,q1]
    - [q0,X,q0] → 0[q0,X,q1] [q1,X,q0]
    - [q0,X,q1] → 0[q0,X,q1] [q1,X,q1]

- δ(q1,ε,X) = {(q1,ε)}
    - [q1,X,q1] → ε

- δ(q0,1,X) = {(q1,ε)}
    - [q0,X,q1] → 1

- δ(q1,ε,Z0) = {(q1,ε)}
    - [q1,Z0,q1] → ε

Hemos obtenido las siguientes producciones

- S → [q0, Z0, q0],   S → [q0, Z0, q1]
    - [q0,Z0,q0] → 0[q0,X,q0] [q0,Z0,q0]
    - [q0,Z0,q1] → 0[q0,X,q0] [q0,Z0,q1]
    - [q0,Z0,q0] → 0[q0,X,q1] [q1,Z0,q0]
    - [q0,Z0,q1] → 0[q0,X,q1] [q1,Z0,q1]
    - [q1,X,q1] → 1
    - [q0,X,q0] → 0[q0,X,q0] [q0,X,q0]
    - [q0,X,q1] → 0[q0,X,q0] [q0,X,q1]
    - [q0,X,q0] → 0[q0,X,q1] [q1,X,q0]
    - [q0,X,q1] → 0[q0,X,q1] [q1,X,q1]
    - [q1,X,q1] → ε
    - [q0,X,q1] → 1
    - [q1,Z0,q1] → ε

De las cuales simplificando obtenemos lo siguiente:

- S → [q0, Z0, q1]
    - [q0,Z0,q1] → 0[q0,X,q1] [q1,Z0,q1]
    - [q1,X,q1] → 1
    - [q0,X,q1] → 0[q0,X,q1] [q1,X,q1]
    - [q1,X,q1] → ε
    - [q0,X,q1] → 1
    - [q1,Z0,q1] → ε

Y dichas producciones generan el siguiente lenguaje, cuya gramática es libre de contexto.

L={0^i 1^j : i >= j >= 0}