Autómatas I.

Parcial III. Tarea I.

Aceptadores de Pila (NPDA).

Juan Francisco Gallo Ramírez

***5to** Semestre UAA*

I.C.I.

10

Ejercicio. *(Sección 7.1)*

10. ¿Qué *lenguaje* es aceptado por el *PDA*

$$\boldsymbol{M} = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{a, b\}, \{0, 1, a, z\}, \delta, z, q_0, \{q_5\}),$$

Con

$$\begin{aligned}
\boldsymbol{\delta}(q_0, b, z) &= \{(q_1, 1z)\},\\
\boldsymbol{\delta}(q_1, b, 1) &= \{(q_1, 11)\},\\
\boldsymbol{\delta}(q_2, a, 1) &= \{(q_3, \lambda)\},\\
\boldsymbol{\delta}(q_3, a, 1) &= \{(q_4, \lambda)\},\\
\boldsymbol{\delta}(q_4, a, z) &= \{(q_4, z), (q_5, z)\}\,?
\end{aligned}$$

10. What language is accepted by the pda

$$M = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{a, b\}, \{0, 1, a, z\}, \delta, z, q_0, \{q_5\}),$$

with

$$\begin{aligned}
\delta(q_0, b, z) &= \{(q_1, 1z)\},\\
\delta(q_1, b, 1) &= \{(q_1, 11)\},\\
\delta(q_2, a, 1) &= \{(q_3, \lambda)\},\\
\delta(q_3, a, 1) &= \{(q_4, \lambda)\},\\
\delta(q_4, a, z) &= \{(q_4, z), (q_5, z)\}?
\end{aligned}$$

Solución

Sabemos que el autómata anterior, y gracias a la definición misma de un autómata no determinista de pila (*NPDA*), tiene únicamente como estado final q_5, es decir, $\boldsymbol{F} = \{q_5\}$. Sin embargo, analizando las funciones de transición $\boldsymbol{\delta}$, nos podemos dar cuenta que hay un punto en el que el autómata no puede hacer transiciones hacia los estados que pueden llevarnos al estado final q_5, concretamente en:

$$\boldsymbol{\delta}(q_1, b, 1) = \{(q_1, 11)\}$$

Entonces, y debido a que el estado inicial es q_0, el autómata sólo puede ciclarse en los estados q_0 y q_1 o bien, directamente rechazar la cadena a falta de una transición definida.

Por lo tanto, se podría decir que el autómata no acepta ninguna cadena y, en consecuencia, el lenguaje queda vacío.

- **Lenguaje aceptado:**

$$\mathrm{L} = \emptyset$$

11

Ejercicio. *(Sección 7.1)*

11. ¿Qué *lenguaje* es aceptado por el *NPDA*

$$\boldsymbol{M} = (\{q_0, q_1, q_2\}, \{a, b\}, \{a, b, z\}, \delta, z, q_0, \{q_2\}),$$

Con

$$\begin{aligned}\boldsymbol{\delta}(q_0, a, z) &= \{(q_1, a), (q_2, \lambda)\},\\ \boldsymbol{\delta}(q_1, b, a) &= \{(q_1, b)\},\\ \boldsymbol{\delta}(q_1, b, b) &= \{(q_1, b)\},\\ \boldsymbol{\delta}(q_1, a, b) &= \{(q_2, \lambda)\}\ ?\end{aligned}$$

11. What language is accepted by the npda $M = (\{q_0, q_1, q_2\}, \{a, b\}, \{a, b, z\}, \delta, q_0, z, \{q_2\})$ with transitions

$$\begin{aligned}\delta(q_0, a, z) &= \{(q_1, a), (q_2, \lambda)\},\\ \delta(q_1, b, a) &= \{(q_1, b)\},\\ \delta(q_1, b, b) &= \{(q_1, b)\},\\ \delta(q_1, a, b) &= \{(q_2, \lambda)\}?\end{aligned}$$

Solución

Para encontrar el lenguaje aceptado por el *NPDA* se va a analizar todas las funciones de transición $\boldsymbol{\delta}$, con la finalidad de encontrar una estructura de las cadenas aceptadas.

Tenemos primeramente:

$$\boldsymbol{\delta}(q_0, a, z) = \{(q_1, a), (q_2, \lambda)\}$$

Esta función $\boldsymbol{\delta}$ nos indica que al iniciar en q_0 solo se podrá continuar si se lee una letra a, y ya que no hay más transiciones definidas para este estado entonces se puede confirmar que la cadena tiene obligatoriamente que empezar con una letra a. Además observamos que la transición nos puede llevar a q_2 que es estado final, por lo que se acepta la cadena $\boldsymbol{w} = a$.

Después tenemos:

$$\boldsymbol{\delta}(q_1, b, a) = \{(q_1, b)\}$$
$$\boldsymbol{\delta}(q_1, b, b) = \{(q_1, b)\}$$
$$\boldsymbol{\delta}(q_1, a, b) = \{(q_2, \lambda)\}$$

Se llega q_1 después de haber leído una letra a como se mencionó antes y solo se puede ir al estado final q_2 si en la pila se tiene una letra b y se lee a posterior una letra a, pero para tener una letra b en la pila, se tuvo que haber leído antes una b, sin importar que haya en la pila. Entonces esto nos siguiere que tiene que haberse leído forzosamente al menos una letra b, y posterior a eso, una letra a.

Juntando todos los requerimientos para la cadena, es sencillo llegar a una estructura que sigue la forma:

$$\boldsymbol{ab^n a}, n > 0$$

Recordando que también acepta:

$$\boldsymbol{a}$$

- **Lenguaje aceptado:**

$$\mathrm{L} = \{a\} \cup \{ab^n a : n > 0\}$$

12

Ejercicio. *(Sección 7.1)*

12. ¿Qué *lenguaje* es aceptado por el *NPDA* en el **Ejemplo 7.4** si usamos $\boldsymbol{F} = \{q_0, q_f\}$?

12. What language is accepted by the npda in Example 7.4 if we use $F = \{q_0, q_f\}$?

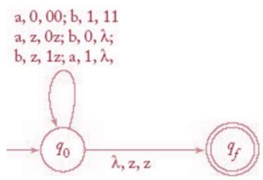
Example 7.4

Construct an npda for the language

$$L = \{w \in \{a, b\}^* : n_a(w) = n_b(w)\}$$

As in Example 7.2, the solution to this problem involves counting the number of a's and b's, which is easily done with a stack. Here we need not even worry about the order of the a's and b's. We can insert a counter symbol, say 0, into the stack whenever an a is read, then pop one counter symbol from the stack when a b is found. The only difficulty with this is that if there is a prefix of w with more b's than a's, we will not find a 0 to use. But this is easy to fix; we can use a negative counter symbol, say 1, for counting the b's that are to be matched against a's later. The complete solution is given in the transition graph in Figure 7.3.

Figure 7.3



Solución

Sabemos que el *NPDA* anterior acepta todas las cadenas de letras a y b donde se tenga la misma cantidad de unas que de otras, y un factor clave para entender el autómata es que no importa el orden de aparición de los caracteres, por lo que realmente cualquier cadena de letras a y b puede ser candidata a ser aceptada.

Entendiendo un poco lo anterior y observando el autómata en sí, nos daremos cuenta que el estado q_0 es un estado de prueba a la espera de que se encuentren las mismas cantidades de letras de cualquier cadena de letras a y b, por lo tanto si aplicamos que $\boldsymbol{F} = \{q_0, q_f\}$ nos resultará que todas las cadenas del universo serán aceptadas por el autómata.

- **Lenguaje generado:**

$$\mathrm{L} = \boldsymbol{\Sigma}^* = \{a, b\}^*$$

4 Ejercicio. *(Sección 7.1)*

4. Construye los *NPDA's* que acepten los siguientes *lenguajes* en $\boldsymbol{\Sigma} = \{a, b, c\}$.

(a) $L = \{a^n b^{2n} : n \geq 0\}$.

(b) $L = \{\boldsymbol{wcw}^R : \boldsymbol{w} \epsilon \{a, b\}^*\}$.

(c) $L = \{a^n b^m c^{n+m} : n \geq 0, m \geq 0\}$.

4. Construct npda's that accept the following languages on $\Sigma = \{a, b, c\}$.

(a) $L = \{a^n b^{2n} : n \geq 0\}$.

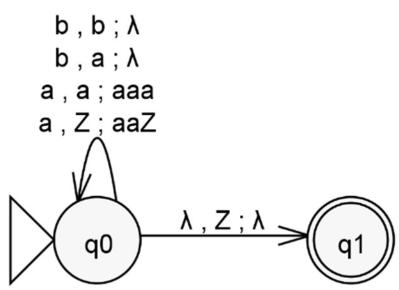
(b) $L = \{wcw^R : w \in \{a, b\}^*\}$.

(c) $L = \{a^n b^m c^{n+m} : n \geq 0, m \geq 0\}$.

Solución

Para dar solución a los inicios se procederá al uso de **JFlap** como herramienta para realizar los grafos de transición de los respectivos autómatas que aceptan los lenguajes anteriores y posteriormente hacer la respectiva definición de los *NPDA*.

(a) $L = \{a^n b^{2n} : n \geq 0\}$.



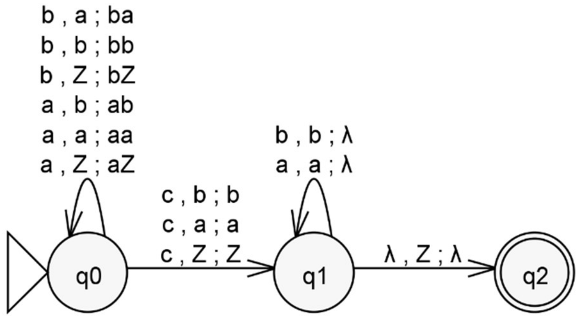
$$\boldsymbol{M} = (\{q_0, q_1\}, \{a, b\}, \{a, Z\}, \delta, Z, q_0, \{q_1\}),$$

Con transiciones:

$$\boldsymbol{\delta}(q_0, a, Z) = \{(q_0, aaZ)\},$$
$$\boldsymbol{\delta}(q_0, a, a) = \{(q_0, aaa)\},$$
$$\boldsymbol{\delta}(q_0, b, a) = \{(q_0, \lambda)\},$$
$$\boldsymbol{\delta}(q_0, b, a) = \{(q_0, \lambda)\},$$
$$\boldsymbol{\delta}(q_0, \lambda, Z) = \{(q_1, \lambda)\}$$

Tenemos que en el estado q_0 se van leyendo primeramente letras a y agregando el doble de estas a la pila, con la finalidad de que para llegar al estado final q_1 con la pila en el carácter Z primero se hayan tenido que quitar todas esas letras a cuando se leyera alguna letra b, lo que en esencia toma el doble de letras b que de a para ello.

(b) $L = \{\boldsymbol{w}c\boldsymbol{w}^R : \boldsymbol{w} \epsilon \{a, b\}^*\}$.



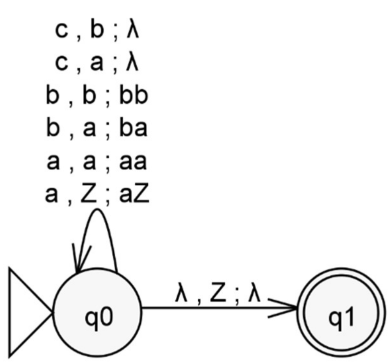
$$\boldsymbol{M} = (\{q_0, q_1, q_2\}, \{a, b, c\}, \{a, b, Z\}, \delta, Z, q_0, \{q_2\}),$$

Con transiciones:

$$\boldsymbol{\delta}(q_0, a, Z) = \{(q_0, aZ)\},$$
$$\boldsymbol{\delta}(q_0, a, a) = \{(q_0, aa)\},$$
$$\boldsymbol{\delta}(q_0, a, b) = \{(q_0, ab)\},$$
$$\boldsymbol{\delta}(q_0, b, Z) = \{(q_0, bZ)\},$$
$$\boldsymbol{\delta}(q_0, b, b) = \{(q_0, bb)\},$$
$$\boldsymbol{\delta}(q_0, b, a) = \{(q_0, ba)\},$$
$$\boldsymbol{\delta}(q_0, c, Z) = \{(q_1, Z)\},$$
$$\boldsymbol{\delta}(q_0, c, a) = \{(q_1, a)\},$$
$$\boldsymbol{\delta}(q_0, c, b) = \{(q_1, b)\},$$
$$\boldsymbol{\delta}(q_1, a, a) = \{(q_1, a)\},$$
$$\boldsymbol{\delta}(q_1, b, b) = \{(q_1, b)\},$$
$$\boldsymbol{\delta}(q_1, \lambda, Z) = \{(q_1, \lambda)\}$$

Tenemos que en el estado q_0 se van leyendo y agregando a la pila cualquier combinación de letras a y b, para que una vez presentada la letra c, se proceda a cambiar al estado q_1 el cual se encarga de validar que la cadena sea el reverso de la primera cadena ingresada antes de la letra c quitando caracteres hasta encontrarse con Z para finalmente hacer la transición al estado final q_2.

(c) $L = \{a^n b^m c^{n+m} : n \geq 0, m \geq 0\}$.



$$\boldsymbol{M} = (\{q_0, q_1\}, \{a, b, c\}, \{a, b, Z\}, \delta, Z, q_0, \{q_1\}),$$

Con transiciones:

$$\boldsymbol{\delta}(q_0, a, Z) = \{(q_0, aZ)\},$$
$$\boldsymbol{\delta}(q_0, a, a) = \{(q_0, aa)\},$$
$$\boldsymbol{\delta}(q_0, b, a) = \{(q_0, ba)\},$$
$$\boldsymbol{\delta}(q_0, b, b) = \{(q_0, bb)\},$$
$$\boldsymbol{\delta}(q_1, c, a) = \{(q_1, \lambda)\},$$
$$\boldsymbol{\delta}(q_1, c, b) = \{(q_1, \lambda)\},$$
$$\boldsymbol{\delta}(q_1, \lambda, Z) = \{(q_1, \lambda)\}$$

Tenemos que en el estado q_0 se van leyendo y agregando a la pila las letras a y posteriormente las letras b, esto con la finalidad de que al ingresarse letras c, sin importar si hay letras a o b en la pila, las vaya sacando de la misma hasta toparse con Z si es que la cantidad de letras c fue igual a la suma de las cantidades de letras a o b, en cuyo caso se haría la transición al estado final q_2.

1 Ejercicio. *(Sección 7.3)*

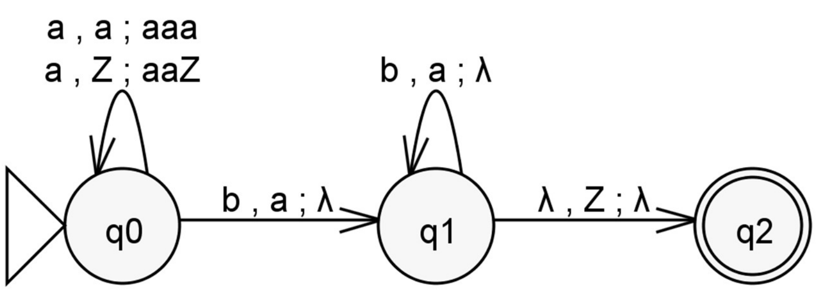
1. Muestre que $L = \{a^n b^{2n} : n \geq 0\}$ es un *lenguaje* *determinista libre de contexto*.

1. Show that $L = \{a^n b^{2n} : n \geq 0\}$ is a deterministic context-free language.

Solución

Sabemos que un *lenguaje* L es *determinista libre de contexto* si y solo si existe un *DPDA* que logre generar al *lenguaje*. Entonces basta con encontrar a un autómata que cumpla con las restricciones de un *DPDA* que logre generar dicho *lenguaje*.

- **DPDA (grafo de transiciones):**



Se ha podido encontrar un *DPDA* que genera al lenguaje anterior, por lo tanto el lenguaje es un lenguaje determinista libre de contexto.

L es **determinista libre de contexto**.

2 Ejercicio. *(Sección 7.3)*

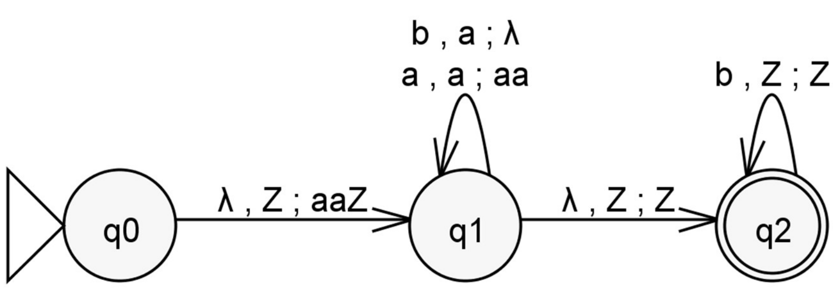
2. Muestre que $L = \{a^n b^m : m \geq n + 2\}$ es un *lenguaje* *determinista libre de contexto*.

2. Show that $L = \{a^n b^m : m \geq n + 2\}$ is deterministic.

Solución

Sabemos que un *lenguaje* L es *determinista libre de contexto* si y solo si existe un *DPDA* que logre generar al *lenguaje*. Entonces basta con encontrar a un autómata que cumpla con las restricciones de un *DPDA* que logre generar dicho *lenguaje*.

- **DPDA (grafo de transiciones):**



Se ha podido encontrar un *DPDA* que genera al lenguaje anterior, por lo tanto el lenguaje es un lenguaje determinista libre de contexto.

L es **determinista libre de contexto**.

3 **Ejercicio.** *(Sección 7.3)*

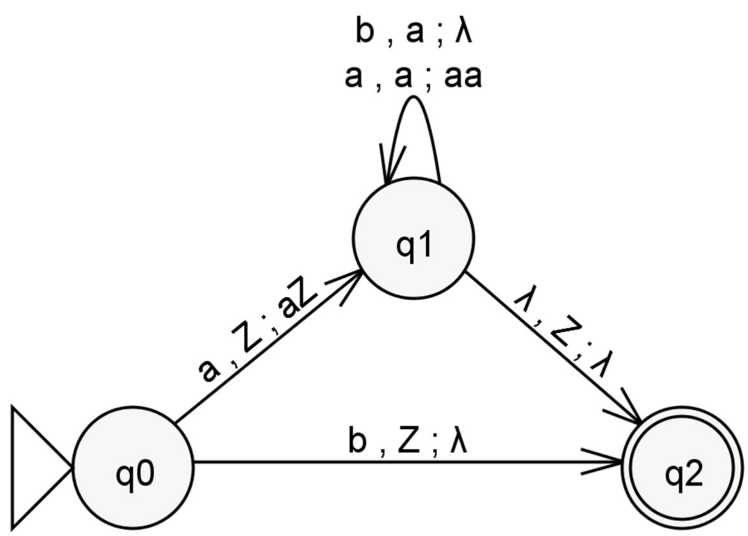
2. ¿Es el *lenguaje* $L = \{a^n b^n : n \geq 0\} \cup \{b\}$ *determinista*?

> 3. Is the language $L = \{a^n b^n : n \geq 1\} \cup \{b\}$ deterministic?

Solución

Sabemos que un *lenguaje* L es *determinista libre de contexto* si y solo si existe un *DPDA* que logre generar al *lenguaje*. Entonces basta con encontrar a un autómata que cumpla con las restricciones de un *DPDA* que logre generar dicho *lenguaje*.

- **DPDA (grafo de transiciones):**



Se ha podido encontrar un *DPDA* que genera al lenguaje anterior, por lo tanto el lenguaje es un lenguaje determinista libre de contexto.

L es **determinista libre de contexto**.

5 Ejercicio. *(Sección 7.3)*

5. Demuestra que el *autómata de pila* del **Ejemplo 7.4** no es determinista, pero que el *lenguaje* en el ejemplo es, sin embargo, *determinista*.

5. Show that the pushdown automaton in Example 7.4 is not deterministic, but that the language in the example is nevertheless deterministic.

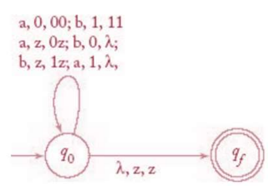
Example 7.4

Construct an npda for the language

$$L = \{w \in \{a,b\}^* : n_a(w) = n_b(w)\}$$

As in Example 7.2, the solution to this problem involves counting the number of a's and b's, which is easily done with a stack. Here we need not even worry about the order of the a's and b's. We can insert a counter symbol, say 0, into the stack whenever an a is read, then pop one counter symbol from the stack when a b is found. The only difficulty with this is that if there is a prefix of w with more b's than a's, we will not find a 0 to use. But this is easy to fix; we can use a negative counter symbol, say 1, for counting the b's that are to be matched against a's later. The complete solution is given in the transition graph in Figure 7.3.

Figure 7.3



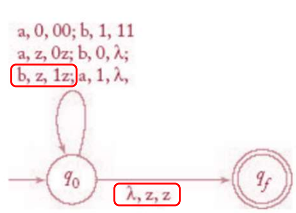
Solución

Recordando que para que un autómata de pila sea determinista debe cumplir que para todo $q \in Q, a \in (\Sigma \cup \{\lambda\}), b \in \Gamma$:

- $\boldsymbol{\delta}(q, a, b)$ contiene máximo un elemento.
- Si $\boldsymbol{\delta}(q, \lambda, b)$ no está vacía, entonces $\boldsymbol{\delta}(q, c, b)$ debe estar vacía para todo $c \in \boldsymbol{\Sigma}$.

Entonces, el autómata del **Ejemplo 7.4** rompe con la segunda condición en las siguientes transiciones:

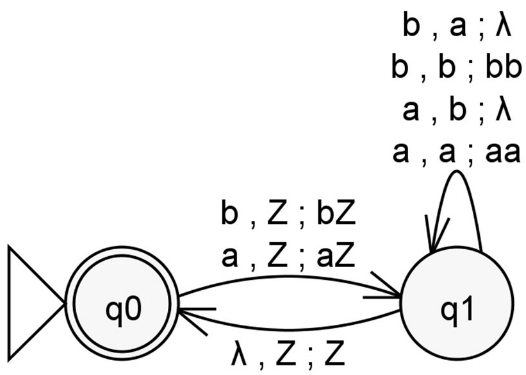
Figure 7.3



Por lo tanto:

El autómata no es **determinista**.

Ahora bien, en cuanto al lenguaje, sabemos que un *lenguaje* L es *determinista libre de contexto* si y solo si existe un *DPDA* que logre generar al *lenguaje*. Entonces basta con encontrar a un autómata que cumpla con las restricciones de un *DPDA* que logre generar dicho *lenguaje*.



Se ha podido encontrar un *DPDA* que genera al lenguaje anterior, por lo tanto el lenguaje es un lenguaje determinista libre de contexto.

L es **determinista libre de contexto**.

15

Ejercicio. *(Sección 7.3)*

15. Demuestre que todo *lenguaje* *regular* es un *lenguaje* *determinista libre de contexto*.

> 15. Show that every regular language is a deterministic context-free language.

Solución

Si recordamos que para que un *lenguaje sea regular* tiene que existir un *DFA* que lo pueda representar, y que un *DFA* puede ser adaptado de tal forma que la pila no tenga utilidad (podemos siempre leer un mismo carácter arbitrario de la pila y volverlo a colocar en todos los estados), entonces estaríamos obteniendo un *DPDA*, ya que el no determinismo no existe en los *DFA* y de esta forma estaría cumpliendo con todas las restricciones necesarias de un *DPDA*, en consecuencia, como un *lenguaje determinista libre de contexto* tiene un *DPDA* que lo represente, se podría decir que **todo lenguaje regular es a su vez, un lenguaje determinista libre de contexto**.