Sintaxis y Semántica del Lenguaje

UTN – FRLP

Clase 8 - Gramática Libre de Contexto y Autómatas de Pila

Diseñar Gramáticas Libres de Contexto

Diseñar Autómatas Finitos de Pila

Resolución de ejercicios TP 6

Gramáticas Libres de Contexto

Diseñar Gramáticas Libres de Contexto

Las gramáticas libres de contexto se caracterizan por presentar un solo símbolo no terminal del lado izquierdo de una regla de producción y del lado derecho una combinación de dos o más símbolos terminales y/o no terminales.

$A \rightarrow ab$	<A>::=a<B>
$A \rightarrow CbD$	<A>::=<C>b<D>

<dig> ::= 0|1|2….|9
<nro>::= <dig>| <nro><dig>

Diseñar Gramáticas Libres de Contexto

Definiciones

::= → se lee ‘se define como’

| → se lee ‘or’

< > → representa un símbolo no terminal

<A> <B> → se lee ‘A seguido de B’

Diseñar Gramáticas Libres de Contexto

Pensemos en el formato de el nombre de una variable:

x, suma, valor1, porc2A

Puede tener una sola letra, varias letras seguidas, o letras con números; pero no puede comenzar con un número:

<id> ::= <letra>
<id> ::= <id><letra> /*regla recursiva para repetir letras
<id> ::= <id><dig> /*regla recursiva a izq, para evitar que comience con un número
<dig> ::= 0|1|2….|9
<letra> ::= a|b|….z|A|..|Z

Diseñar Gramáticas Libres de Contexto

Ahora, si nos fijamos podríamos reemplazar las primeras 3 líneas:

<id> ::= <letra>
<id> ::= <id><letra>
<id> ::= <id><dig>

Por una sola regla utilizando el or “|”:

<id> ::= <letra> | <id><letra> | <id><dig>

Y nos quedaría algo así:

<id> ::= <letra> | <id><letra> | <id><dig>
<dig> ::= 0|1|2….|9
<letra> ::= a|b|….z|A|..|Z

Diseñar Gramáticas Libres de Contexto

Ejemplo 1: Import en Python

<import>::= import <listamodulos>

<listamodulos>::=<unmodulo> | <unmodulo>,<listamodulos>

<unmodulo>::= os|sys|math| <iden>| ……

<iden>::= <letra> | <iden>_<letra>| <iden>_<dig>|<iden><letra>| <iden><dig>

<letra>::=a|b|c|d…….|z

<dig>::=0|1|2 …….|9

Diseñar Gramáticas Libres de Contexto

Ejemplo 2: Asignación en Python

<asignacion>::= <iden>= <expresión>
<expresión>::=<expresión>+<término>| <expresión> -
<término>|<término>
<termino>::= <termino> * <factor> | <termino> / <factor> | <factor>
<factor>::=<iden> |<num>
<iden>::= <letra> | <iden>_<letra>| <iden>_<dig>|<iden><letra>|
<iden><dig>
<letra>::=a|b|c|d……..……|z
<num>::=<num><dig>|<dig>
<dig>::=0|1|2 ……..……|9

Diseñar Gramáticas Libres de Contexto

Ejemplo 3: Función en Python

<defineFun>::= def <iden>: <cuerpo>| def <iden>(<param>): <cuerpo>
<cuerpo>::= <listasentencias> return <expresión>
<param>::= <param>,<unpar>| <unpar>
<unpar>::= <iden> | <iden>=<num>
<lista sentencias>::= <listasentencias><unasent>|<unasent>
<unasent>::= <while> | <if> | <if-elif>| …..
<expresión>::= <expresión>+<término>| <expresión> - <término>|<término>
<termino>::= <termino> * <factor> | <termino> / <factor> | <factor>
<factor>::= <iden> |<num>
<iden>::= <letra> | <iden>_<letra>| <iden>_<dig>|<iden><letra>| <iden><dig>
<letra>::= a|b|c|d……|z
<dig>::= 0|1|2 ……|9
<num>::= <num><dig>|<dig>

Diseñar Gramáticas Libres de Contexto

Ejemplo 4: For en Python

<for>::= for <iden> in range (<rango_valores>):<lista_sentencias>
<lista_sentencias>::= <sentencia><lista_sentencias> | <sentencia>
<sentencia>::= <if>|<if-elif>|<for>|<while>|....
<rango_valores>::= <expresion>|<expresion>, <expresion>|<expresion>, <expresion>, <expresion>
<expresión>::= <expresión>+<término>| <expresión> - <término>|<término>
<termino>::= <termino> * <factor> | <termino> / <factor> | <factor>
<factor>::= <iden> |<num>
<num>::=<num><dig>|<dig>
<iden>::= <letra> | <iden>_<letra>| <iden>_<dig>|<iden><letra>|
<iden><dig>
<letra>::= a|b|c|d……|z
<dig>::= 0|1|2 ……|9

Autómatas Finitos de Pila

Diseñar Autómatas Finitos de Pila

El AFP es un autómata finito que cuenta con una pila como estructura de almacenamiento auxiliar. El primer elemento de la pila siempre es el caracter ‘z’ con el cual podemos indicar si la palabra es válida o no.

- Si llegamos al final de la palabra y solo queda la z por desapilar la palabra es válida.
- Si al finalizar el análisis de la palabra aún quedan más caracteres por desapilar la palabra no es válida.

Vamos a verlo en un ejemplo.

Diseñar Autómatas Finitos de Pila

Diseñar el AFP sobre el alfabeto $\Sigma=\{a,b\}$ para el lenguaje $L=\{w/w\ a^n b^n, n>=1\}$

Primero defino las primeras 3 palabras posible

w_1 (para n=1) = ab

w_2 (para n=2) = aabb

w_3 (para n=3) = aaabbb

Diseñar Autómatas Finitos de Pila

Diseñar el AFP sobre el alfabeto Σ={a,b} para el lenguaje L={w/w $a^n b^n$, n>=1}

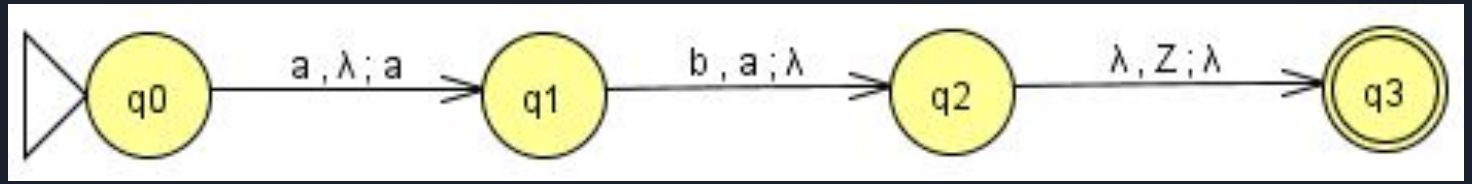
Ahora armo el autómata que cumpla con

la primera palabra.

w_1 (para n=1) = ab

w_2 (para n=2) = aabb

w_3 (para n=3) = aaabbb

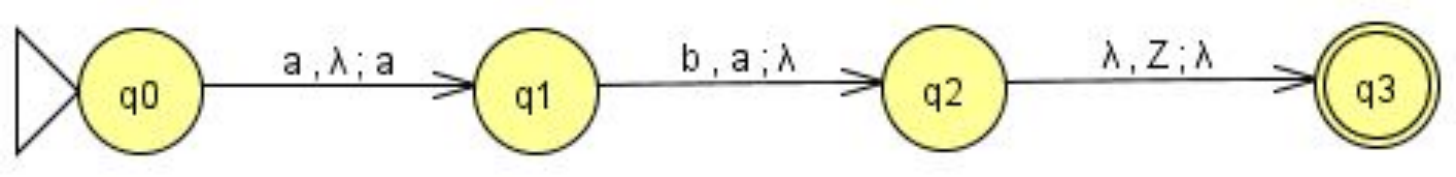


Este autómata valida la palabra ‘ab’.

Diseñar Autómatas Finitos de Pila

A tener en cuenta: Si notan el formato de este autómata, las transiciones tienen la forma $\omega/\alpha/\beta$.

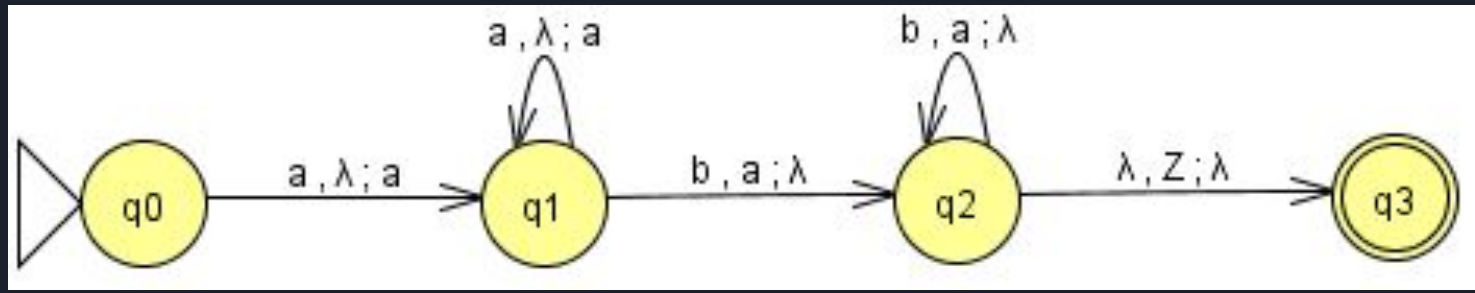
- ω representa el caracter que se lee.
- α representa el caracter que se desapila.
- β representa el caracter que apilo.



Podemos notar que al final se desapila la Z (*Nota: JFLAP no toma este último caracter si no se lo coloca en mayúscula*)

Diseñar Autómatas Finitos de Pila

Ahora acomodamos el autómata para que valide las palabras w_2 y w_3.



Analicemos qué pasa con la pila para las palabras ‘aaabbb’ y ‘aabbb’ (*para hacer en clase*)

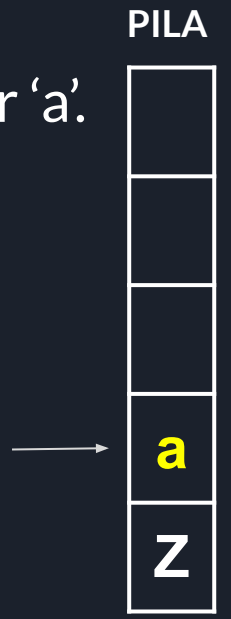
Diseñar Autómatas Finitos de Pila

Caso 1: ‘aaabbb’

aaabbb

Leo a, el autómata me indica que apile el caracter ‘a’.

Me queda por leer ‘aabbb’



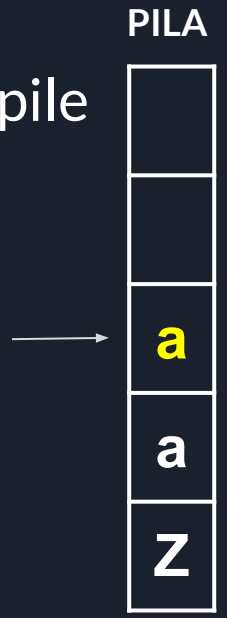
Diseñar Autómatas Finitos de Pila

Caso 1: ‘aaabbb’

aabbb

Leo nuevamente a, el autómata me indica que apile el caracter ‘a’.

Me queda por leer ‘abbb’.



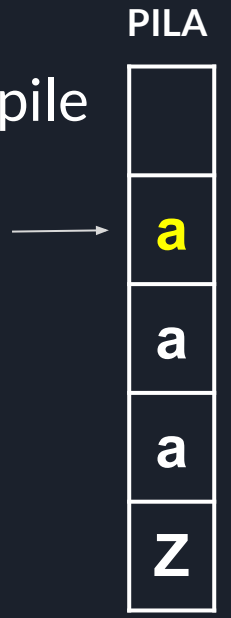
Diseñar Autómatas Finitos de Pila

Caso 1: ‘aaabbb’

abbb

Leo nuevamente a, el autómata me indica que apile el caracter ‘a’.

Me queda por leer ‘bbb’.



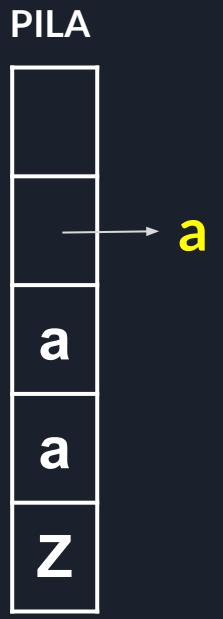
Diseñar Autómatas Finitos de Pila

Caso 1: ‘aaabbb’

bbb

Ahora llega una ‘b’, el autómata me indica que desapile el caracter ‘a’.

Me queda por leer ‘bb’.



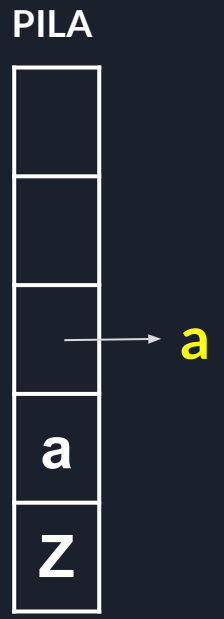
Diseñar Autómatas Finitos de Pila

Caso 1: ‘aaabbb’

bb

Llega otra ‘b’, el autómata me indica que desapile el caracter ‘a’.

Me queda por leer ‘b’.



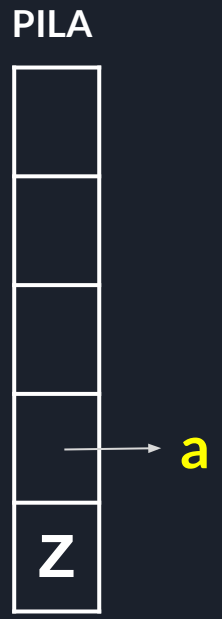
Diseñar Autómatas Finitos de Pila

Caso 1: ‘aaabbb’

b

Llega otra ‘b’ que es el último caracter de la palabra, el autómata me indica que desapile el caracter ‘a’.

Llegué al final de la palabra.



Diseñar Autómatas Finitos de Pila

Caso 1: ‘aaabbb’

Como no llega ningún caracter más y estoy en el nodo q_2, el autómata me indica que desapile ‘Z’.

Como no tengo nada que tape la Z, desapilo y mi pila queda vacía, la palabra es válida.

Diseñar Autómatas Finitos de Pila

Caso 2: ‘aabbb’

aabbb

Leo a, el autómata me indica que apile el caracter ‘a’.

Me queda por leer ‘abbb’

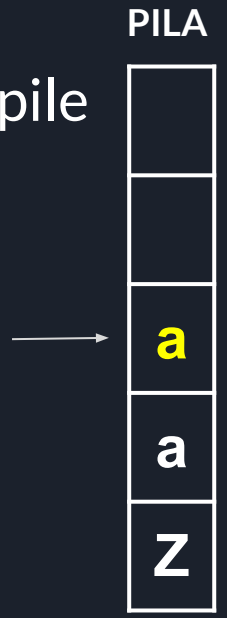
Diseñar Autómatas Finitos de Pila

Caso 2: ‘aabbb’

abbb

Leo nuevamente a, el autómata me indica que apile el caracter ‘a’.

Me queda por leer ‘bbb’.



Diseñar Autómatas Finitos de Pila

Caso 2: ‘aabbb’

bbb

Ahora llega una ‘b’, el autómata me indica que desapile el caracter ‘a’.

Me queda por leer ‘bb’.

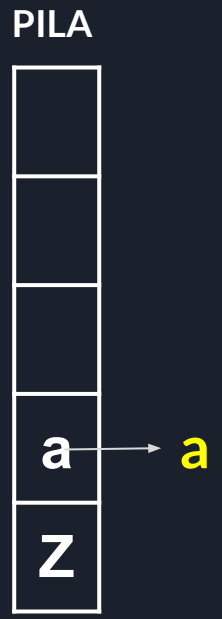
Diseñar Autómatas Finitos de Pila

Caso 2: ‘aabbb’

bb

Llega otra ‘b’, el autómata me indica que desapile el caracter ‘a’.

Me queda por leer ‘b’.



Diseñar Autómatas Finitos de Pila

Caso 2: ‘aabbb’

b

Llega otra ‘b’ que es el último caracter de la palabra, el autómata me indica que desapile el caracter ‘a’.

Al intentar desapilar, puedo ver que el único caracter restante es ‘Z’, por lo tanto no puedo desapilar ‘a’. La palabra no es válida

AFP: Ejercicio para resolver

Definir gráficamente el AFP sobre el alfabeto $\Sigma=\{a,b\}$ que acepte el lenguaje $L=\{w/w\ a^n b^{n-1}, n>=1\}$

Ejercicios para resolver

Resolver los ejercicios del TP 6