|  |  |  |  |
| --- | --- | --- | --- |
| **CARRERA** | **PLAN DE ESTUDIO** | **CLAVE ASIGNATURA** | **NOMBRE DE LA ASIGNATURA** |
| Ingeniería en Computación | 2003-1 | 5044 | Teoría de la Computación |

|  |  |  |  |
| --- | --- | --- | --- |
| **Práctica No.** | **LABORATORIO DE** | Teoría de la Computación | **DURACIÓN**  **(HORAS)** |
| 5 | **Nombre de la Práctica** | Lenguajes libres del contexto | 2 |

|  |  |
| --- | --- |
| Elaboró:  Christian Navarro Cota | Revisó:  Odin Isaac Meling López |

**1.- INTRODUCCIÓN:**

Toda gramática en bruto ha de limpiarse con el objetivo de eliminar todos los símbolos extraños.

El método de limpiar las gramáticas sucias consiste

* En detectar en primer lugar todos los símbolos muertos, y a continuación
* Se detectan todos los símbolos inaccesibles. Es importante seguir este orden, puesto que la eliminación de símbolos muertos puede generar nuevos símbolos inaccesibles.

**2.- OBJETIVO (COMPETENCIA):**

El alumno conocerá las propiedades de clausura de los lenguajes independientes del contexto, utilizando las diversas transformaciones elementales.

**3.- MARCO TEÓRICO:**

**Transformaciones elementales de las gramáticas independientes al contexto**

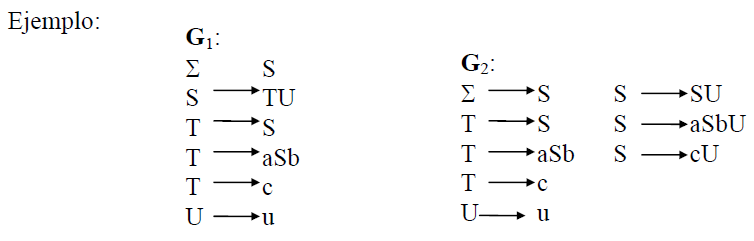
**• Sustitución:**

Una gramática se transforma por sustitución cuando para algún no terminal B, las partes derechas de la regla B sustituyen todas las instancias de B en la parte derecha de una producción.

La transformación de **G** en **G**’ mediante la sustitución del no terminal B que aparece en la parte derecha de la regla: A🡪 ϕBψ y se obtiene aplicando los siguientes pasos:

1.- Todas las producciones de **G** son producciones de **G’** excepto la producción A 🡪 ϕBψ

2.- Cada producción A 🡪 ϕβiψ , i = 1,2, ..., n están en G’



**G**2 es el resultado de la sustitución por las instancias de T en la producción S TU de

**G**1. La producción S TU es omitido en **G**2

**• Expansión:**

Es la operación inversa de la sustitución, en lugar de sustituir un no terminal por la parte derecha de sus reglas, ahora se introduce un nuevo no terminal.

Sea G:

Σ 🡪A

A 🡪ϕψ

La producción A🡪 ϕψ , ϕ, ψ ∈(**N** ∪ **T**)\* - {λ}

De G se transforma en G’ mediante la expansión de la siguiente manera:

Σ 🡪A

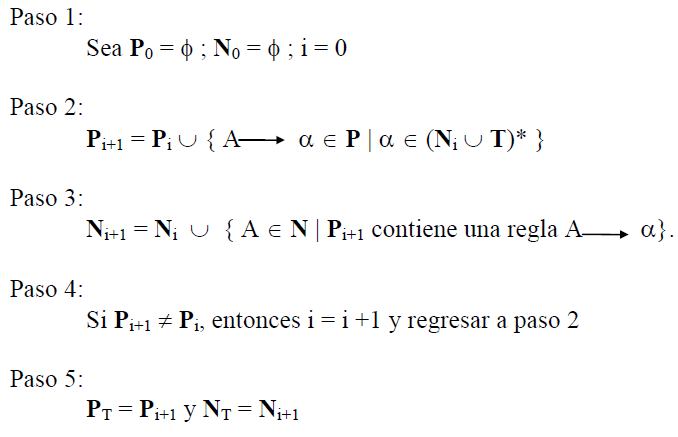
A🡪Xψ

X 🡪ϕ

• Eliminación de producciones inútiles:

Sea **G** = (**N**, **T**, **P**, Σ) una gramática independiente al contexto. Una producción de la forma A α de **G** es útil si **G** permite una secuencia de derivación Σ ===> ϕAψ===>. ϕαψ===> ω¸ ω∈ T\*

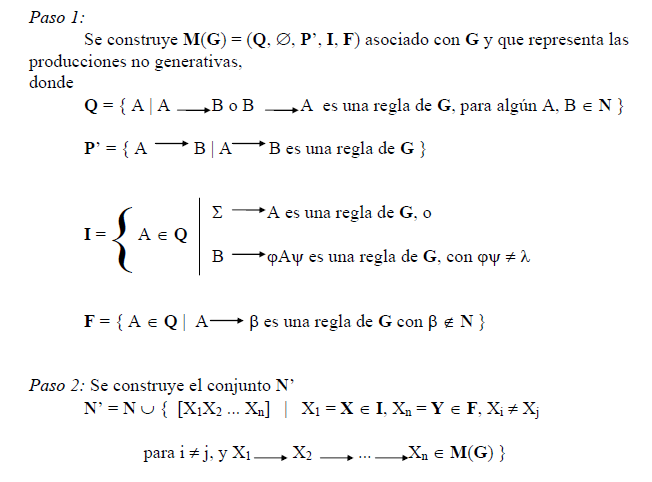
Sea G una gramática independiente al contexto con producciones P. Se crean una secuencia P0, P1, ... subconjuntos de P y N0, N1, ... de subconjuntos de N de acuerdo al siguiente algoritmo:

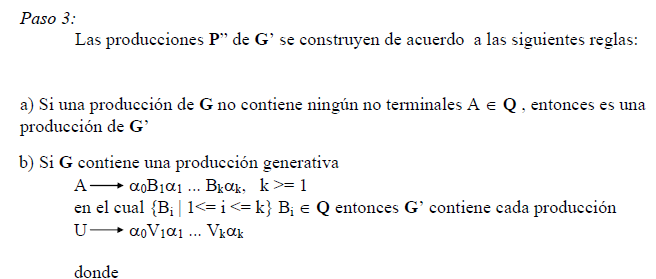


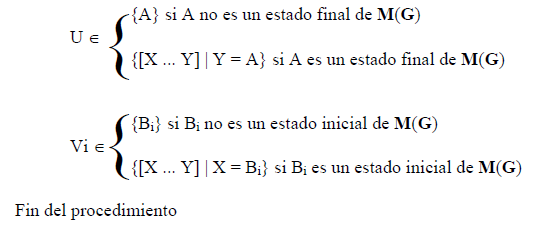
**• Eliminación de producciones no generativas:**

Al eliminar las producciones no generativas de una gramática independiente al contexto también se eliminan los no terminales cíclicos, este es un paso importante para convertir gramáticas en su forma canónica.

Dado una gramática independiente al contexto G = ( N, T, P, Σ ) construir otra G’ = ( N’, T, P”, Σ ) que no contenga producciones no generativas. Se sigue el siguiente algoritmo:





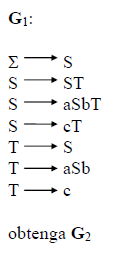


**4.- DESCRIPCIÓN**

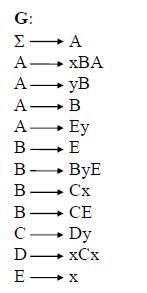
**A) PROCEDIMIENTO Y DURACIÓN DE LA PRÁCTICA:**

**Lea y comprenda la información anterior y con base en ella resuelva los siguientes problemas.**

* Para la siguiente gramática **G**1 aplique la operación de expansión de tal manera que solo aparezca a lo mucho dos símbolos en la parte derecha.

****

* Correr el algoritmo de producciones inútiles con la siguiente gramática:

****

**B) CÁLCULOS Y REPORTE:**

Entregar al maestro los resultados obtenidos de sus operaciones.

**C) RESULTADOS Y CONCLUSIONES:**

Al finalizar la practica, comparara sus resultados con los de sus compañeros y discutiremos en clase el porqué de ellos y dudas que surgieron a lo largo de la practica.

**5.- ANEXOS:**

* Introducción a la teoría de autómatas, lenguajes y computación.

Hopcroft, J. E.; Motwani, R.; Ullman, J. D.

* Introduction to the theory of computation, Michael Sipser, PWS Publishing Company, 1997