

**Instituto Politecnico Nacional**

**ESCOM “ESCUELA SUPERIOR DE CÓMPUTO”**

*TEORÍA COMPUTACIONAL*

*PRÁCTICA 6: AMBIGÜEDAD EN GLC*

PROFA: Luz María Sánchez García

ALUMMNO: Rojas Alvarado Luis Enrique

GRUPO: 2CM11

INTRODUCCIÓN

El propósito de la práctica consistió en que en base a un lenguaje generado a partir de las reglas dadas por la práctica se requiere realizar el árbol de derivación para generar distintas cadenas que posteriormente serán aceptadas o no aceptadas en la gramática y verificar tanto el tipo de gramática cómo si es una gramática ambigua.

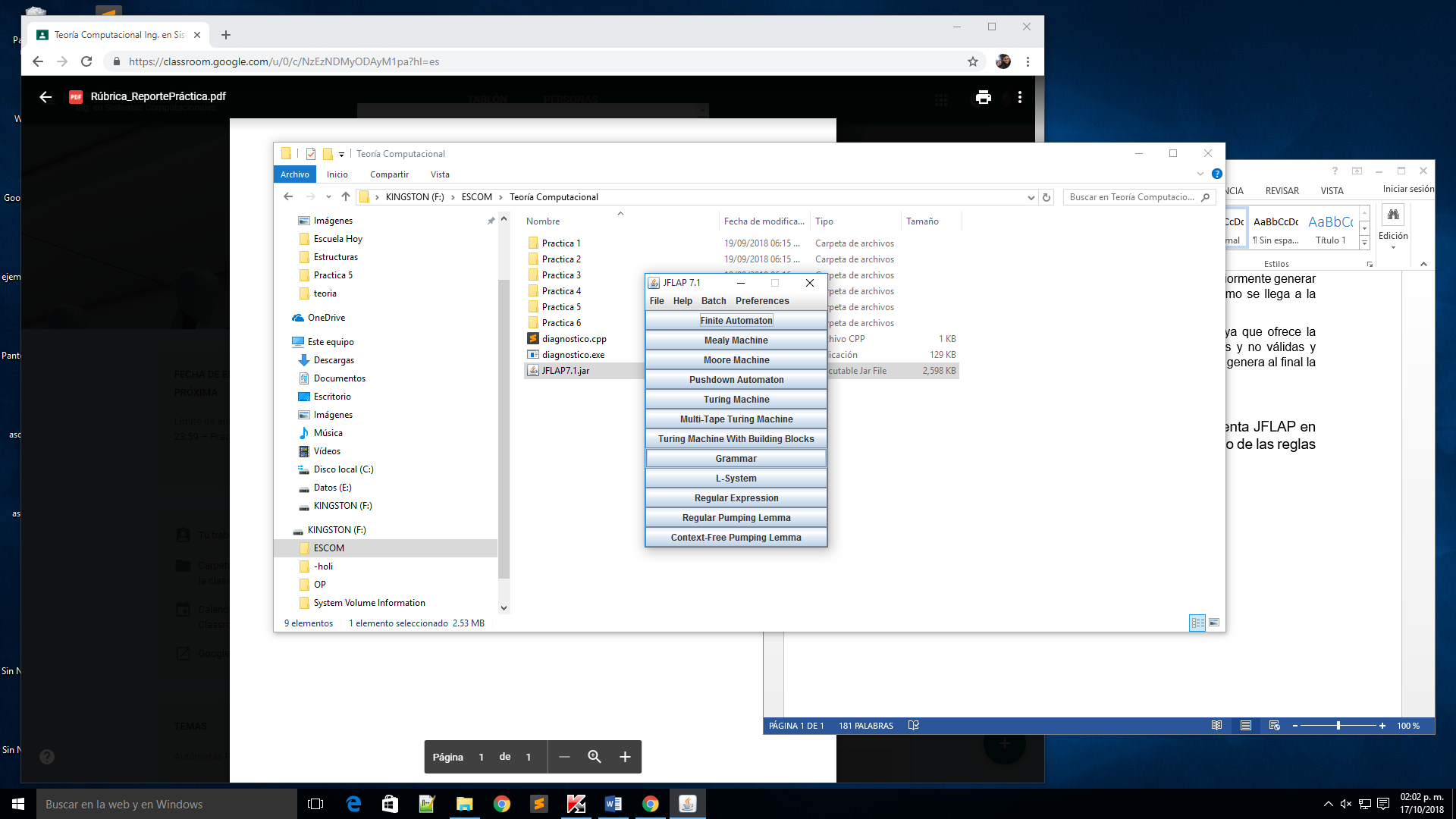
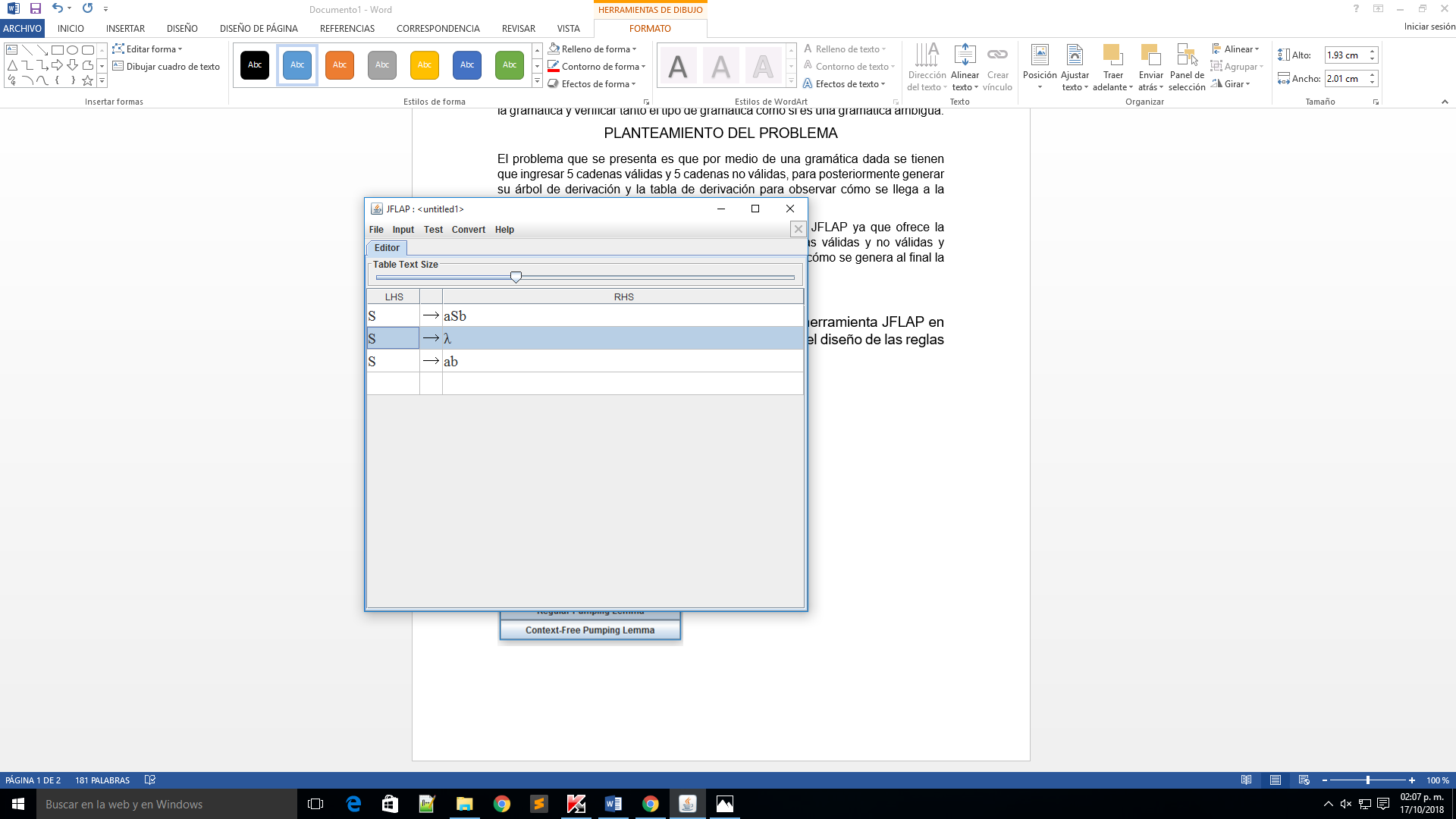
PLANTEAMIENTO DEL PROBLEMA

El problema que se presenta es que por medio de una gramática dada se tienen que ingresar 5 cadenas válidas y 5 cadenas no válidas, para posteriormente generar su árbol de derivación y la tabla de derivación para observar cómo se llega a la cadena ingresada.

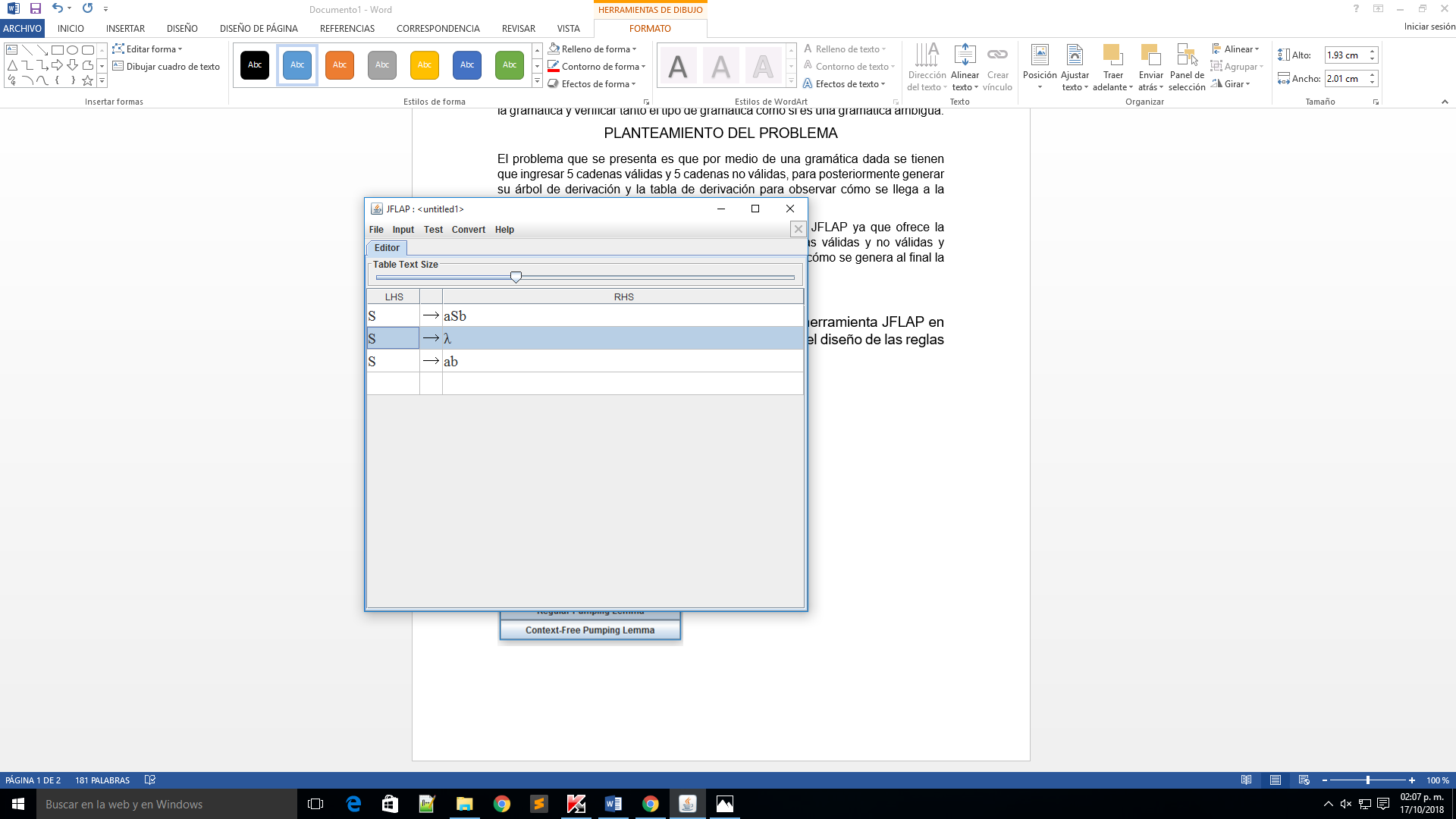
Para darle solución se optó por usar la herramienta de JFLAP ya que ofrece la opción para generar la gramática e ingresar las cadenas válidas y no válidas y posteriormente generar el árbol de derivación mostrando cómo se genera al final la cadena.

DISEÑO DE LA SOLUCIÓN

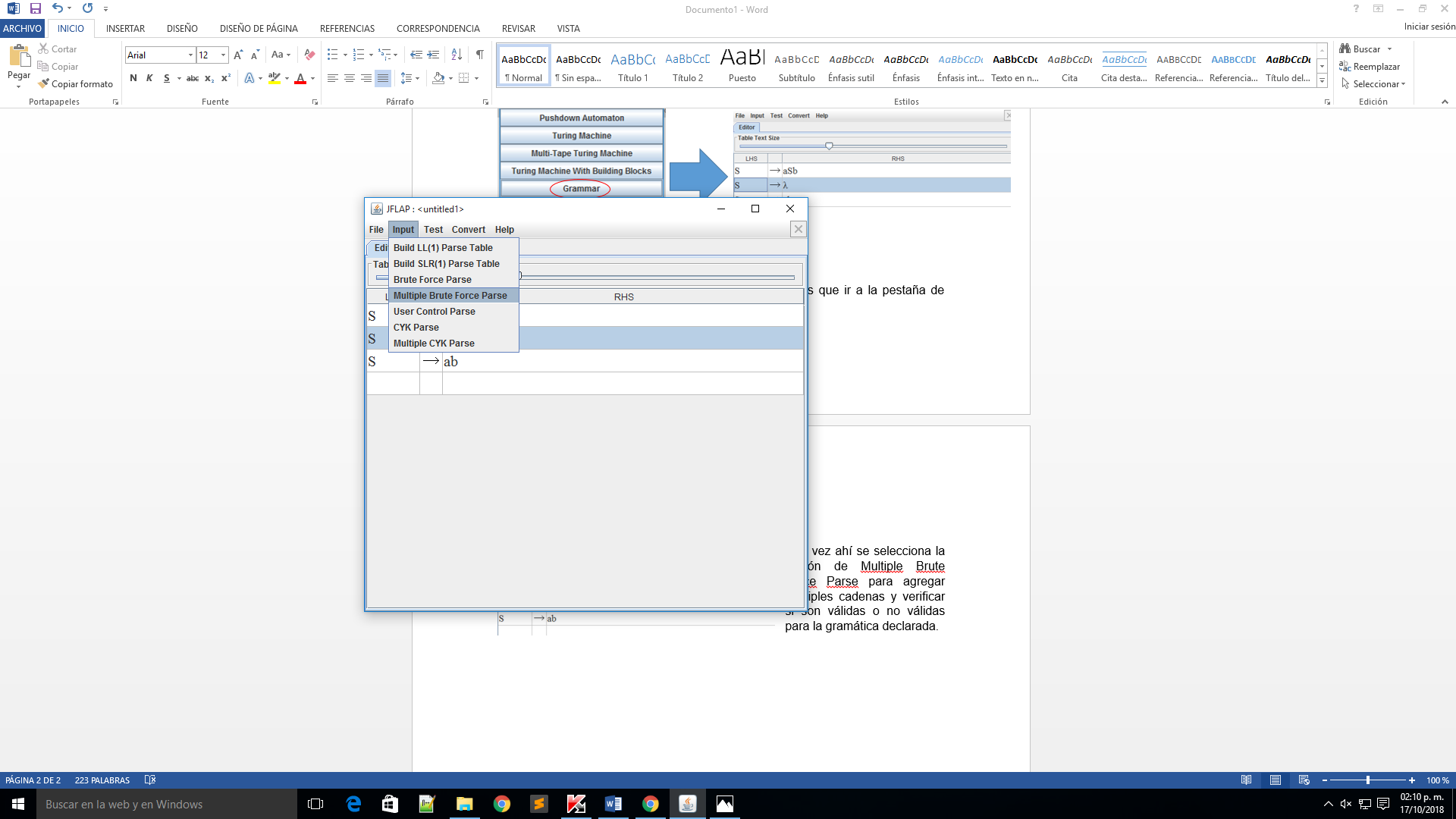
Como ya se comentó anteriormente se utilizó la herramienta JFLAP en la interfaz de gramática y posteriormente se hace el diseño de las reglas de la gramática.



Para ingresar las expresiones simplemente nos tenemos que ir a la pestaña de input.

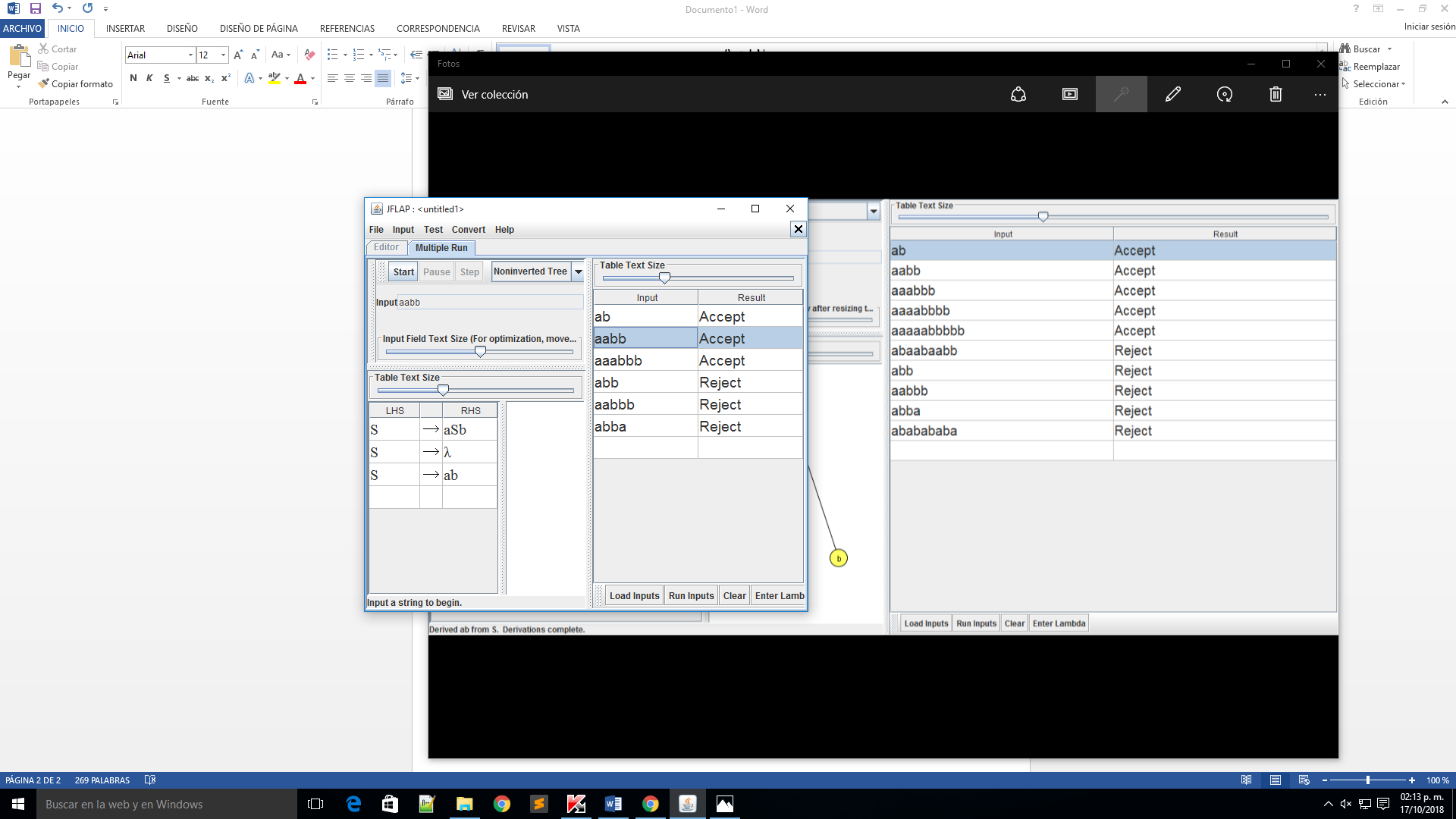
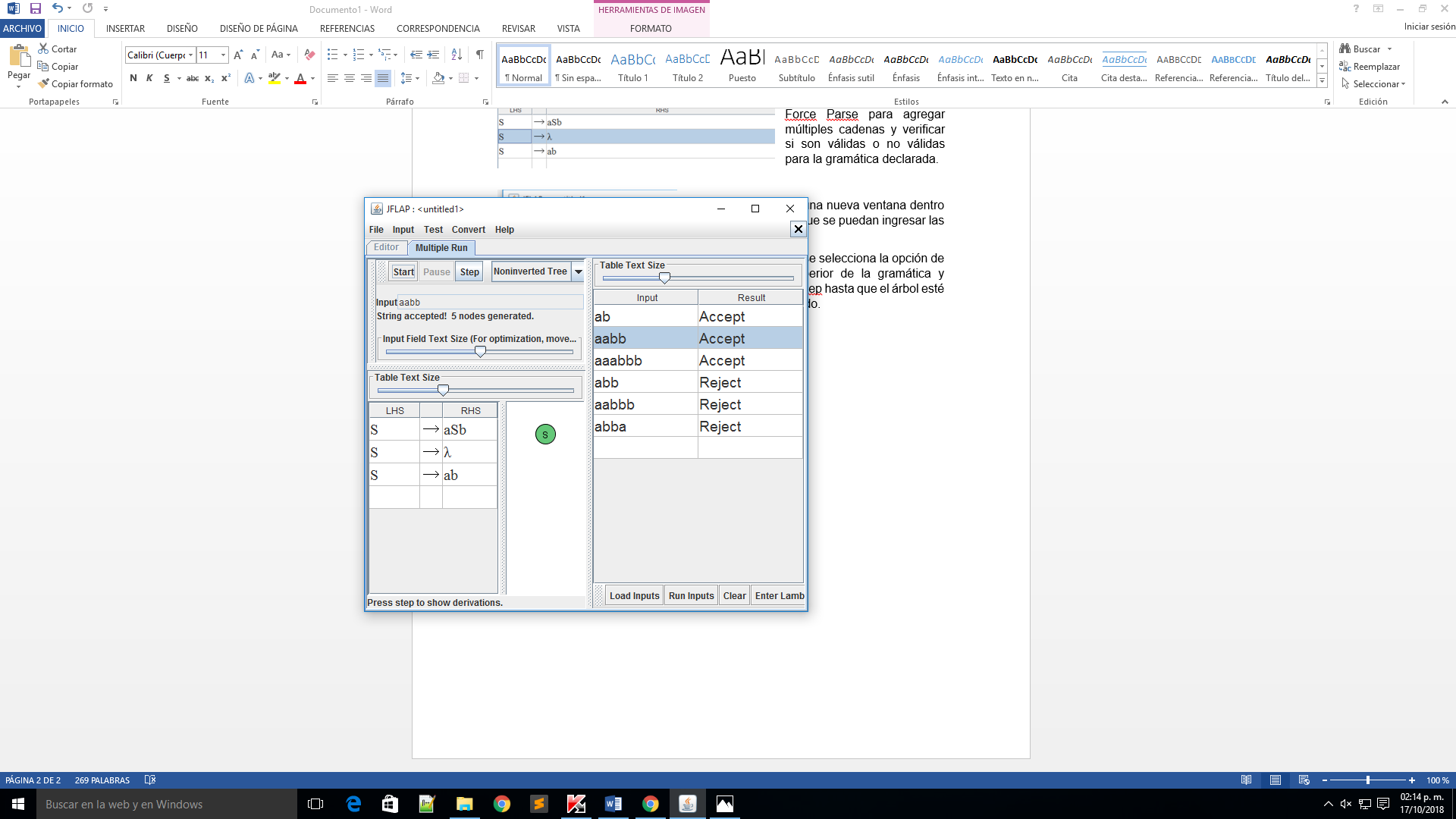


Una vez ahí se selecciona la opción de Multiple Brute Force Parse para agregar múltiples cadenas y verificar si son válidas o no válidas para la gramática declarada.

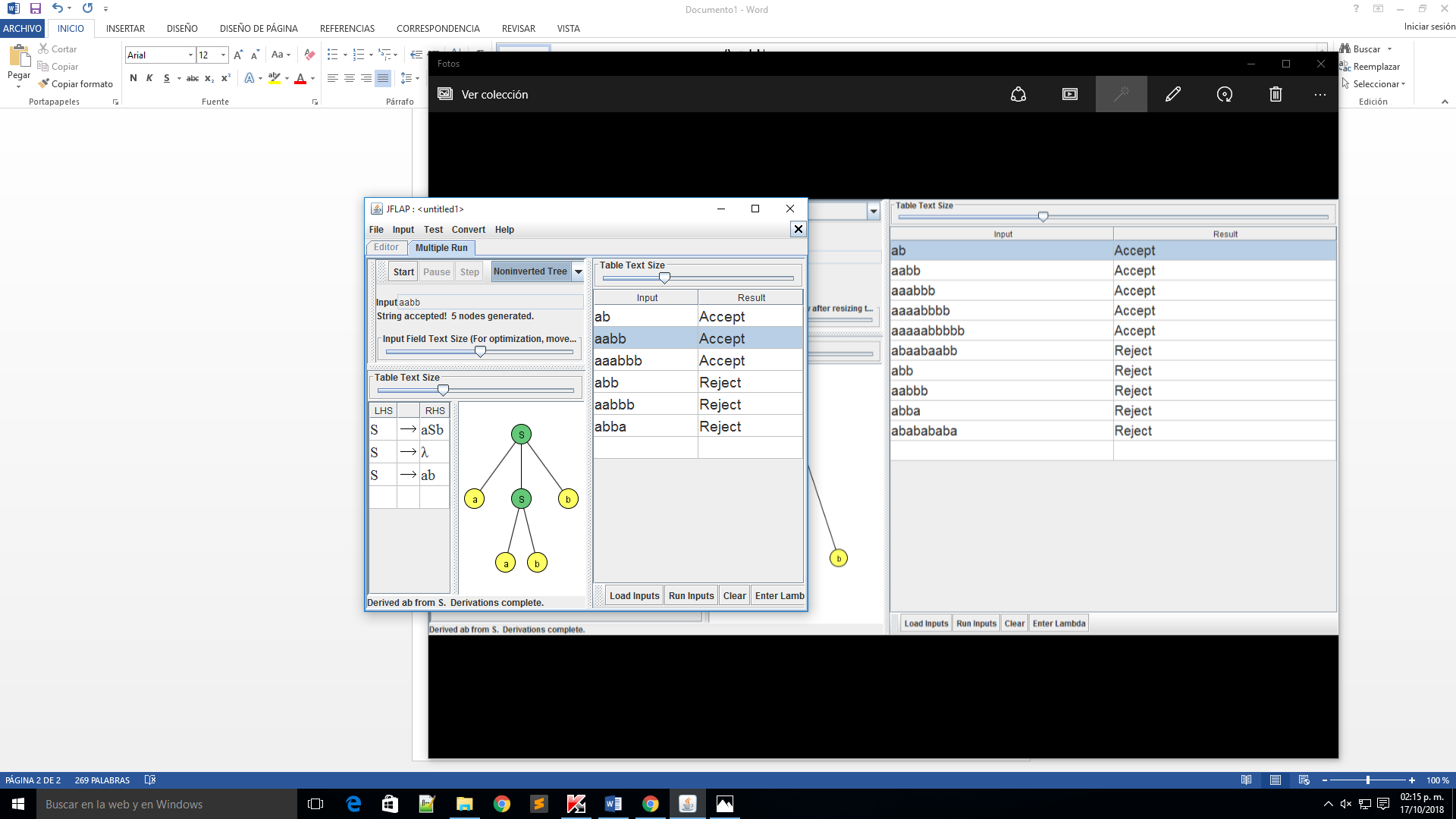
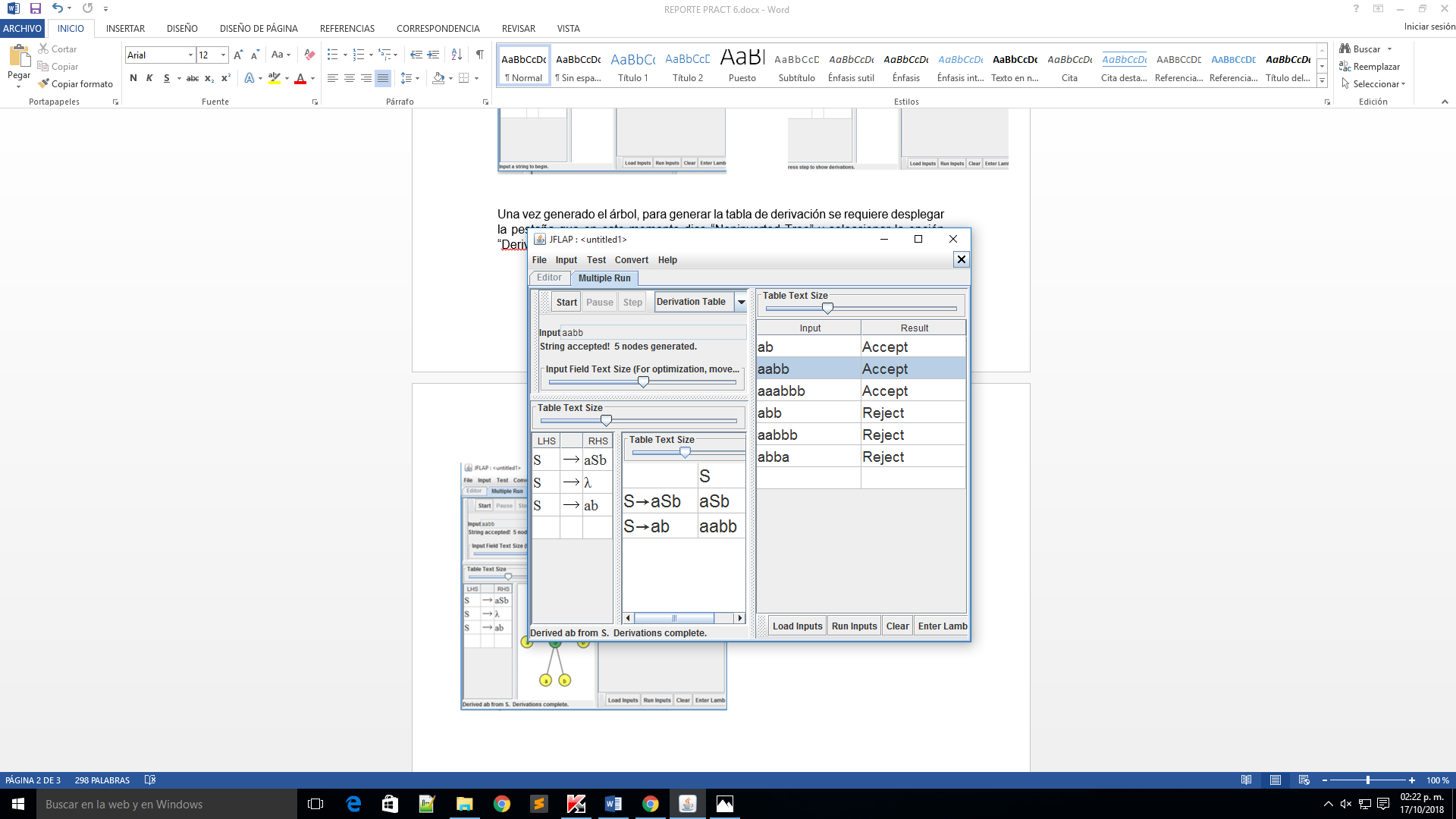


La pestaña generará una nueva ventana dentro de la aplicación para que se puedan ingresar las cadenas deseadas.

Para generar el árbol se selecciona la opción de start en la parte superior de la gramática y después seleccionar step hasta que el árbol esté completamente formado.



Una vez generado el árbol, para generar la tabla de derivación se requiere desplegar la pestaña que en este momento dice “Noninverted Tree” y seleccionar la opción “Derivation Table”



De ésta manera obtendremos tanto las cadenas válidas y no válidas así como el árbol de derivación y la tabla de derivación. Para así poder sacar una conclusión sobre el lenguaje que maneja la gramática, que tipo de expresión es y si es ambigua o no.

IMPLEMENTACIÓN DE LA SOLUCIÓN

1. S->aSb | ab | e
2. S->SS+ | SS\* |A

A->0 | 1

1. S->(S)S | e
2. S->Aa | b

A->Ac | Sd | e

1. S->aSbS | bSaS | e
2. S->0 | 1 |S+S | S\* |SS | (S)
3. S->a | (L)

L->SL | e

1. S->aSbb | abb | e
2. S->AB

A->aAb | e

B->cBd | e

1. A->a

E->b

A->azb

A->aX

E->E

G->g

X->XE

D->eI

X->z

Y->b

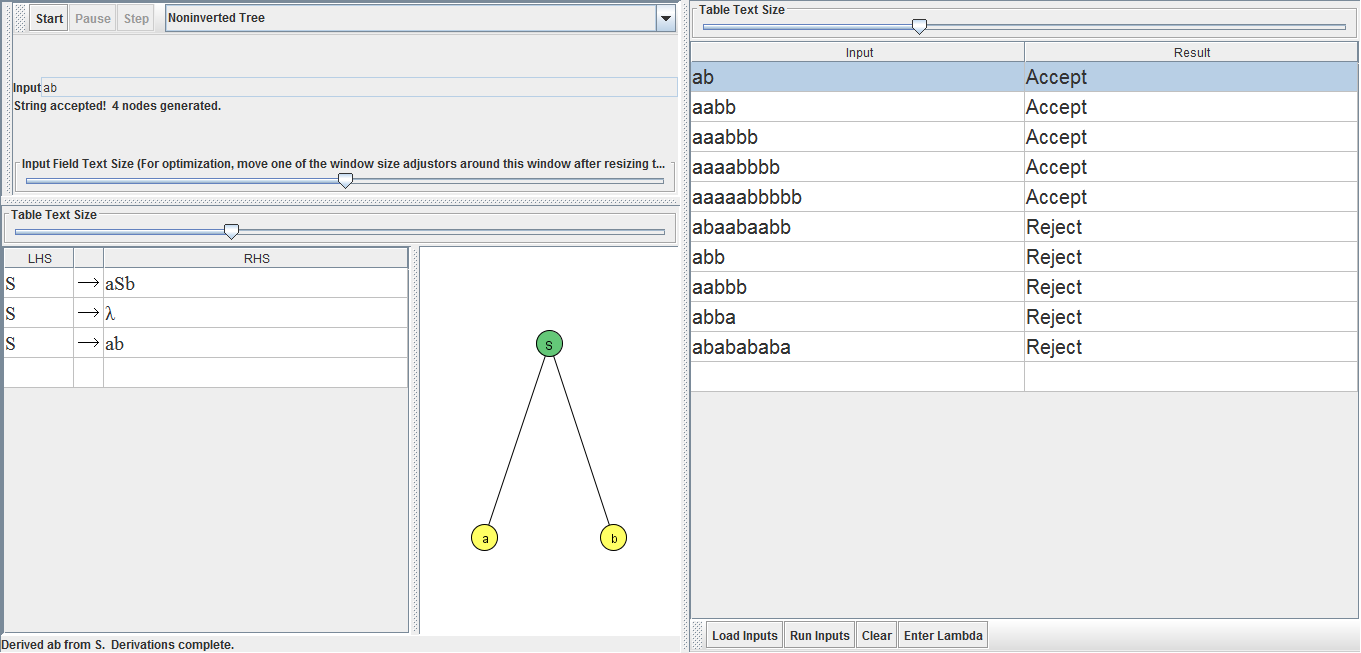
I->fG

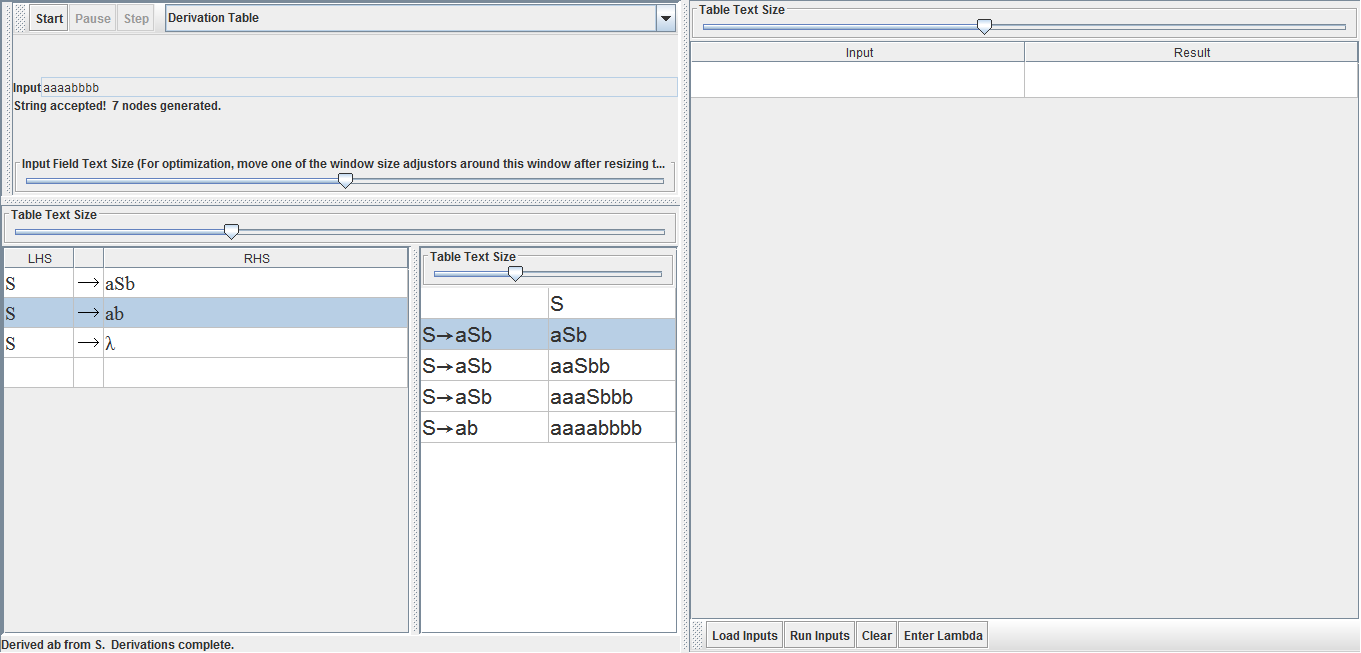
x->Xb

E->d

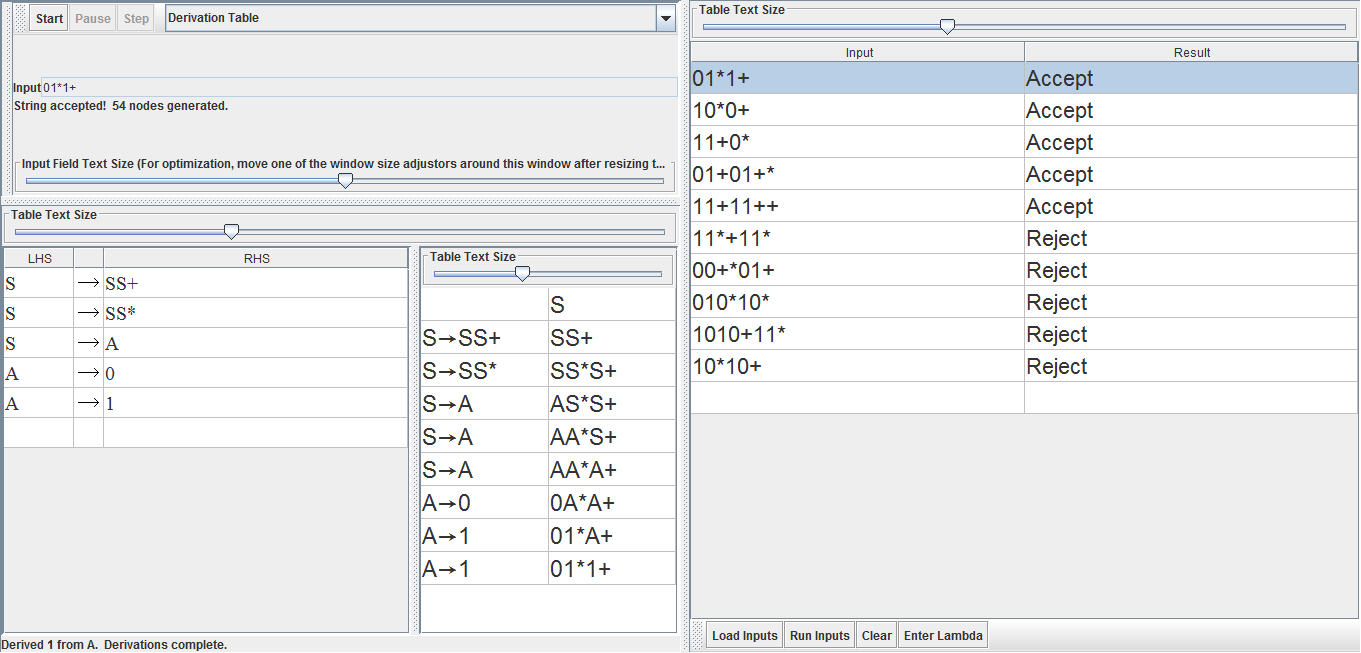
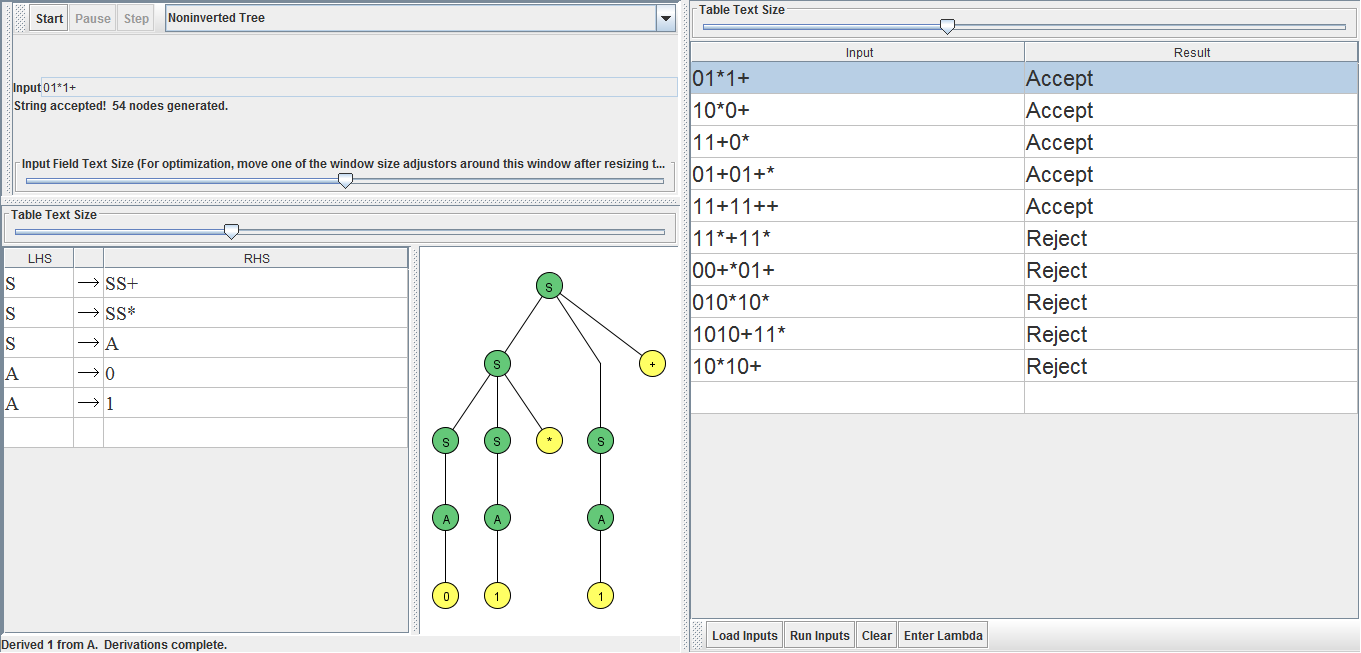
1. S->abS | Sab | aSb | SS | e
2. S->aSbS | bSaS | e

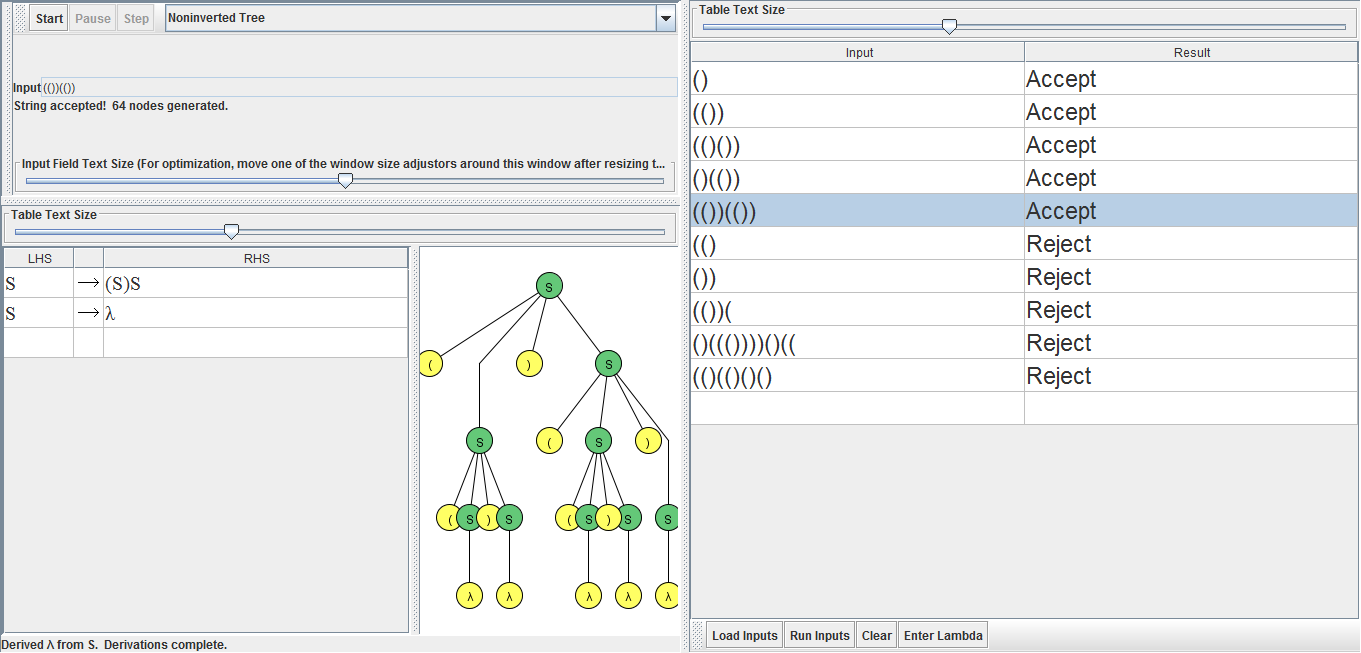
FUNCIONAMIENTO

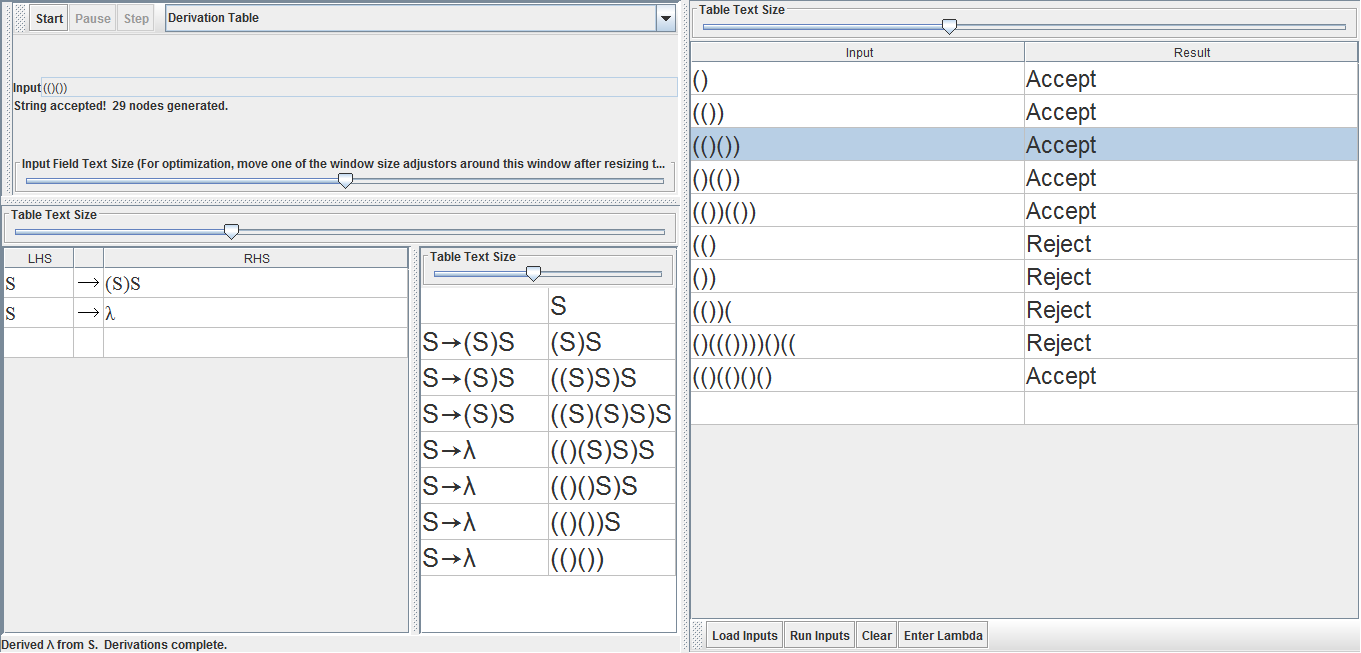
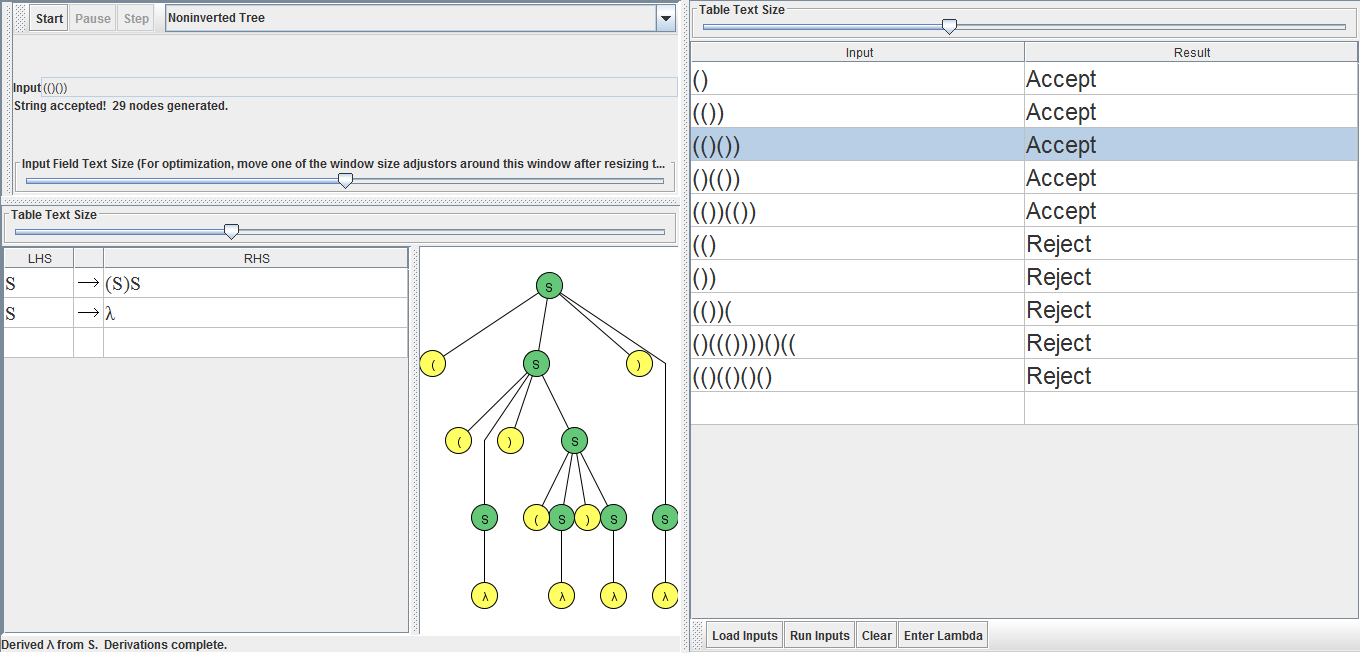
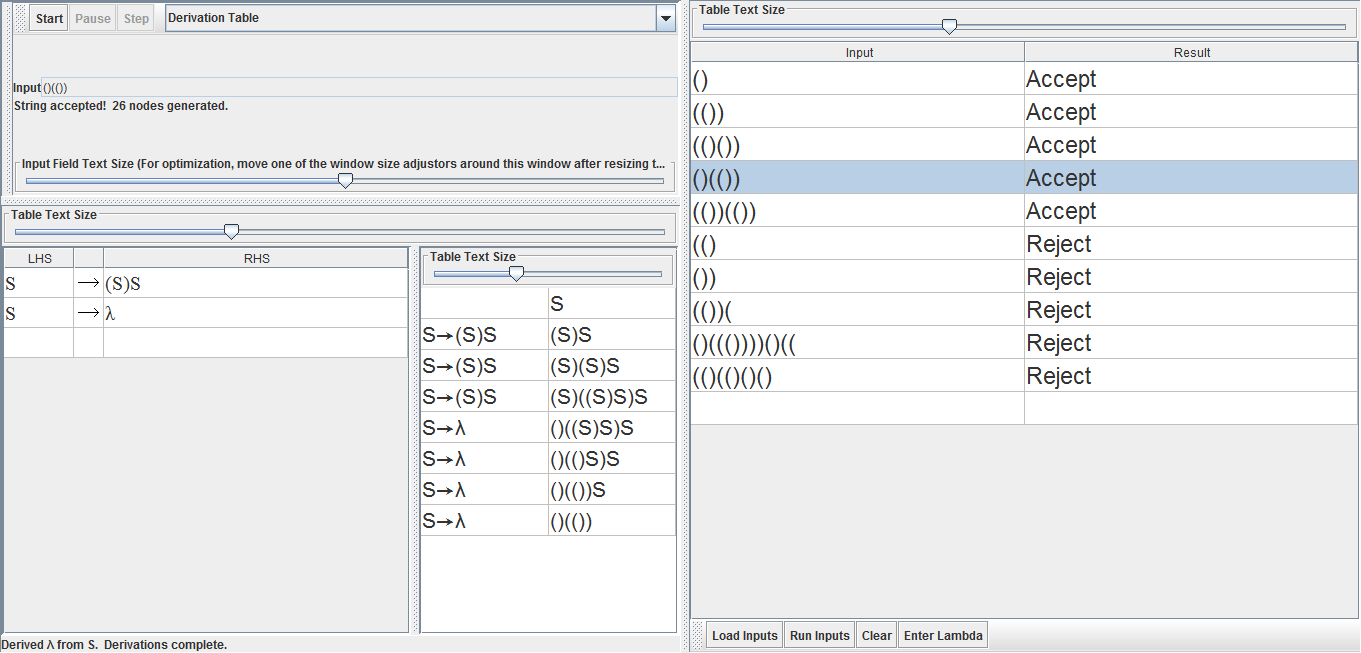
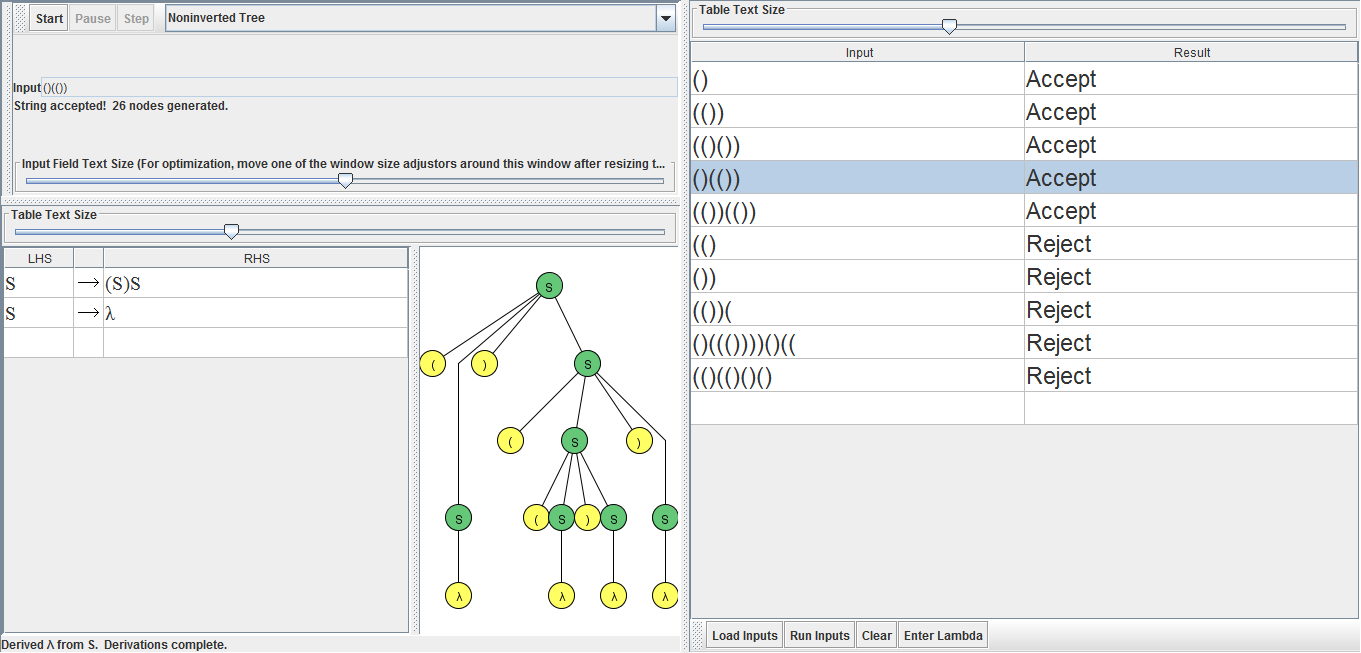
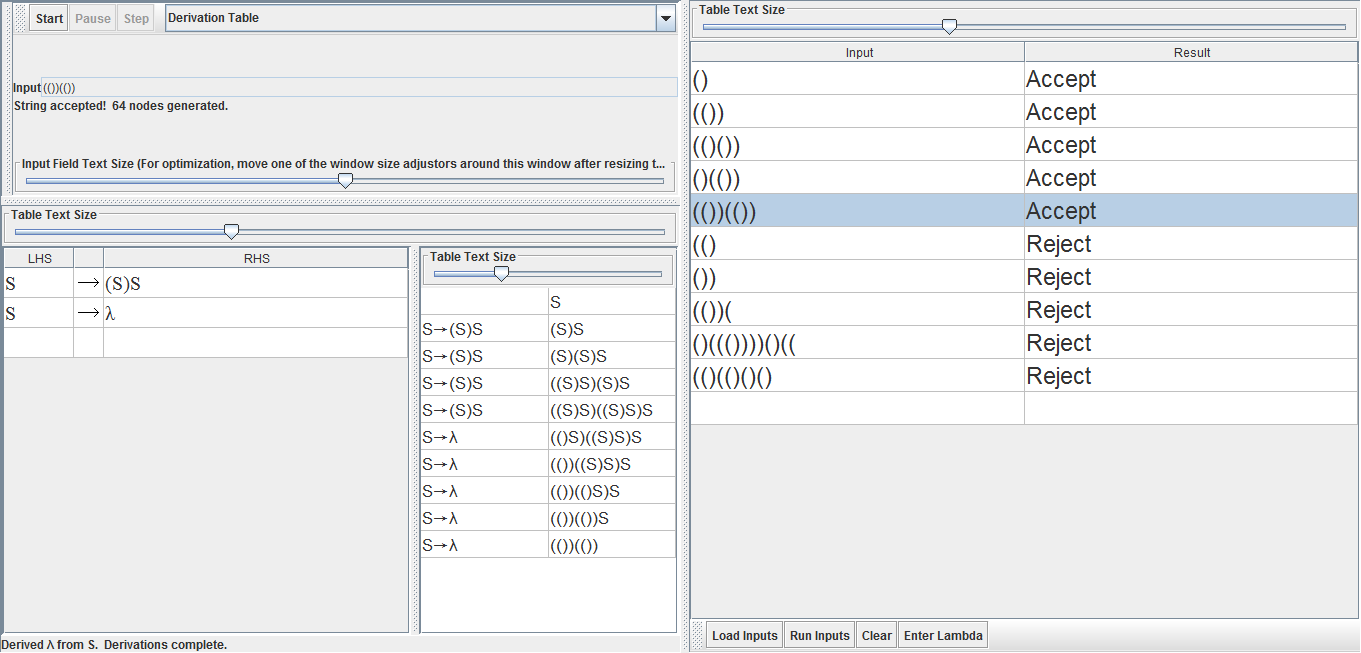
1. L(G1)={an bn | n=m=0,1,2,…} Tipo 2, No ambigua

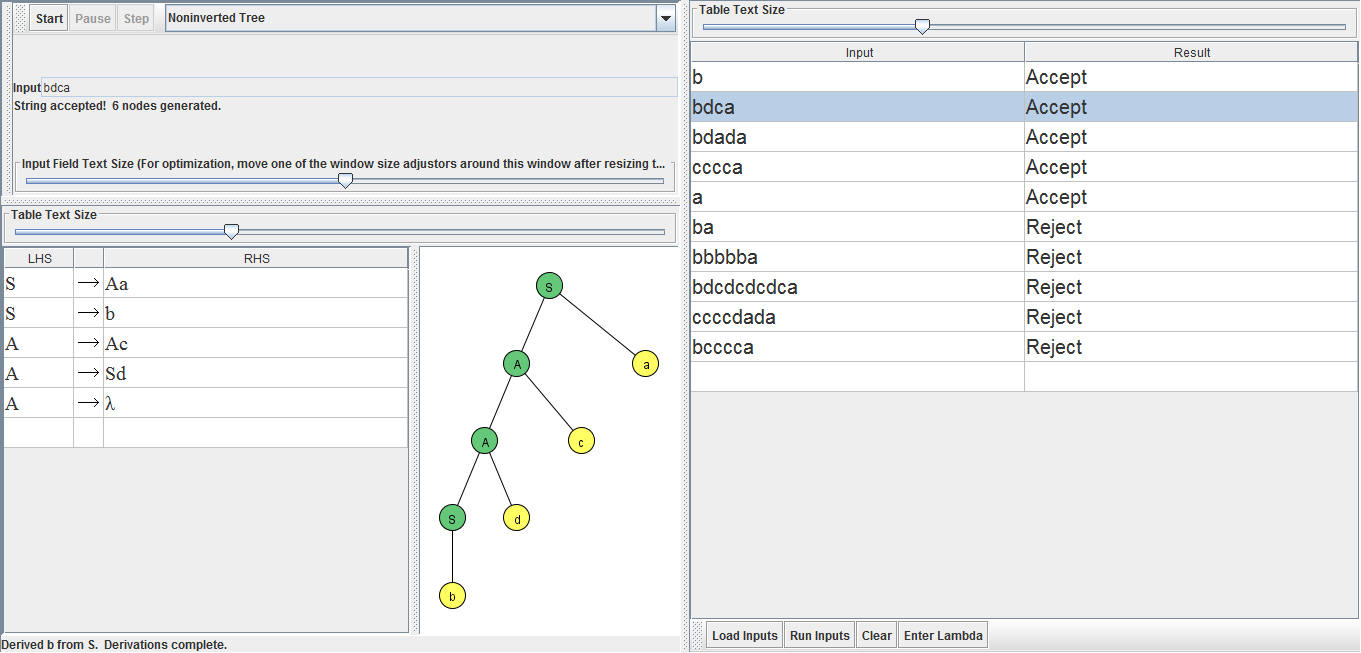


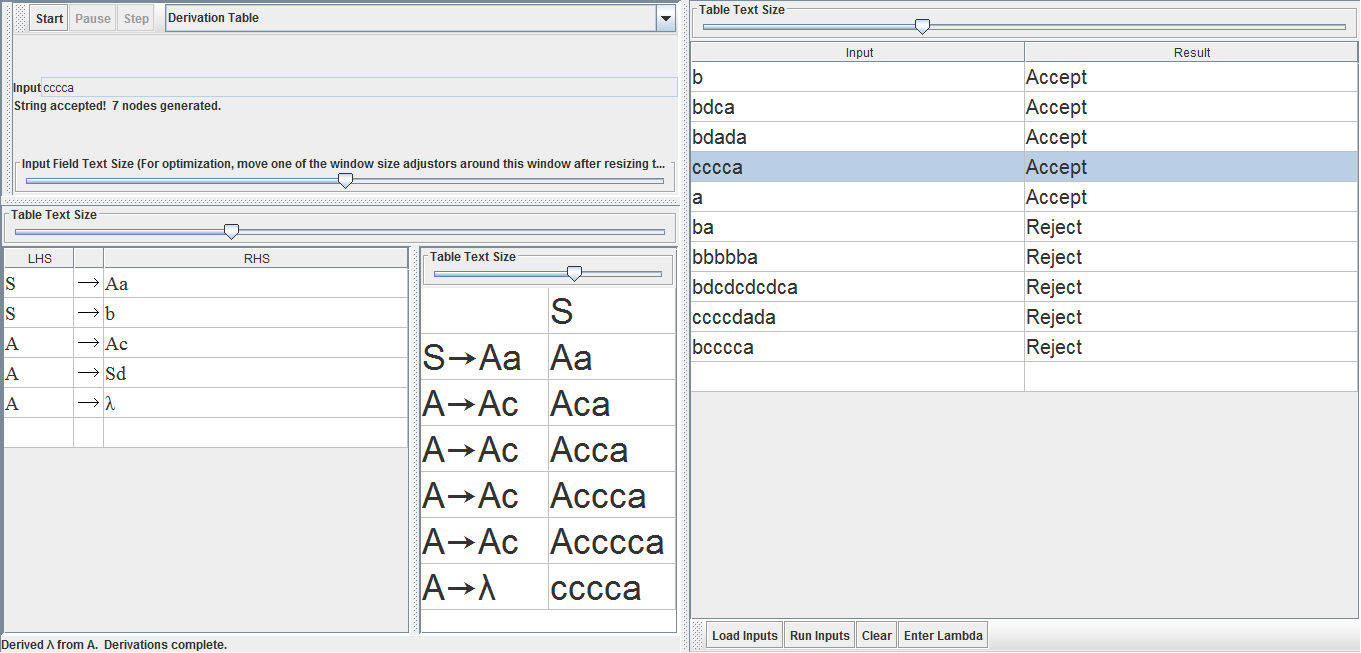
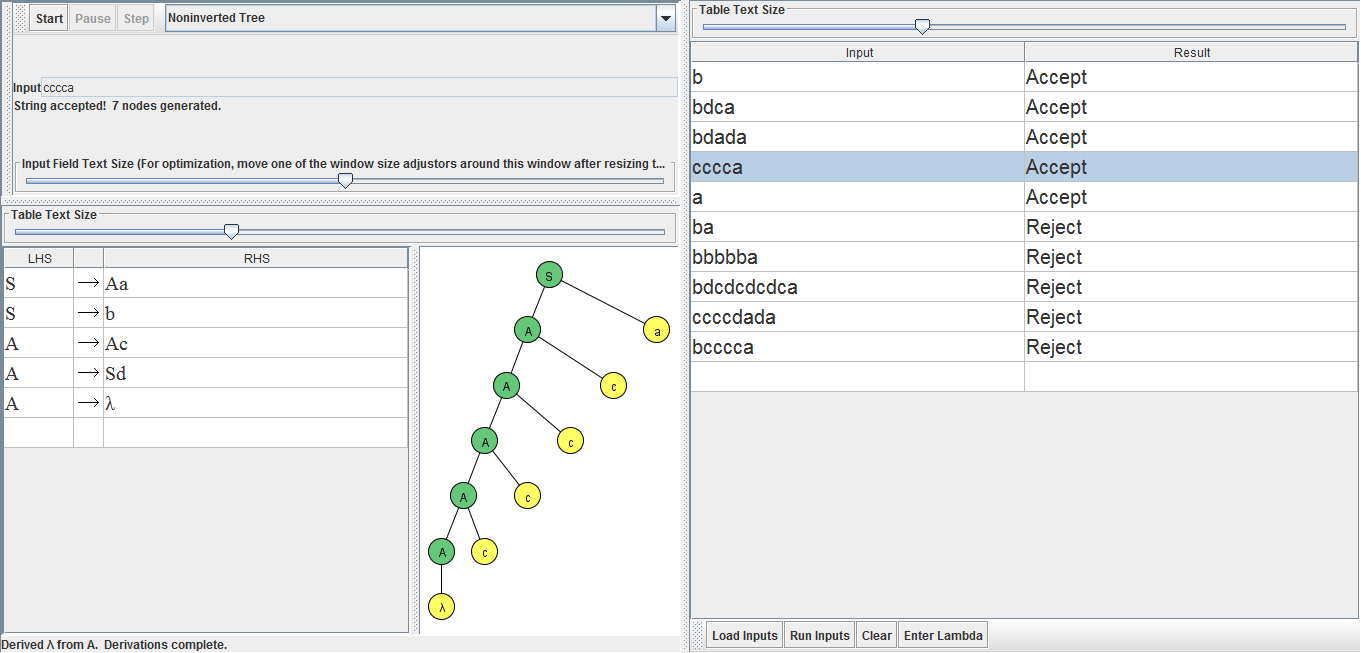
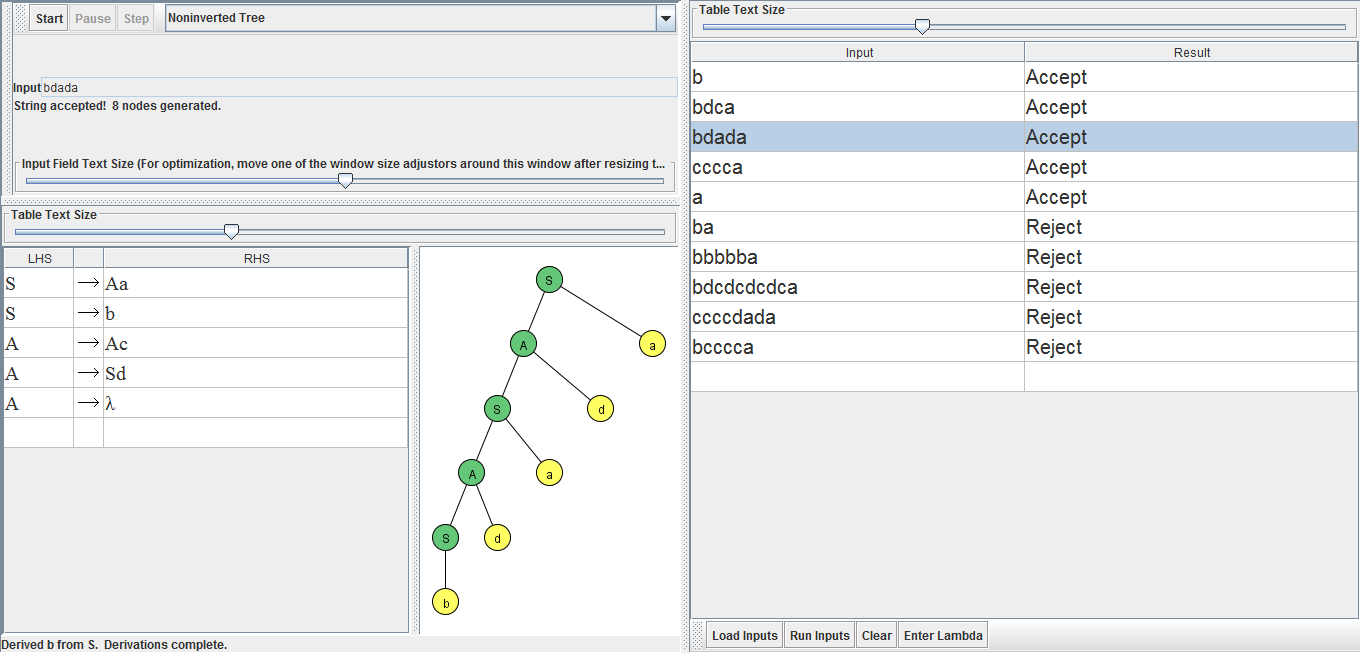
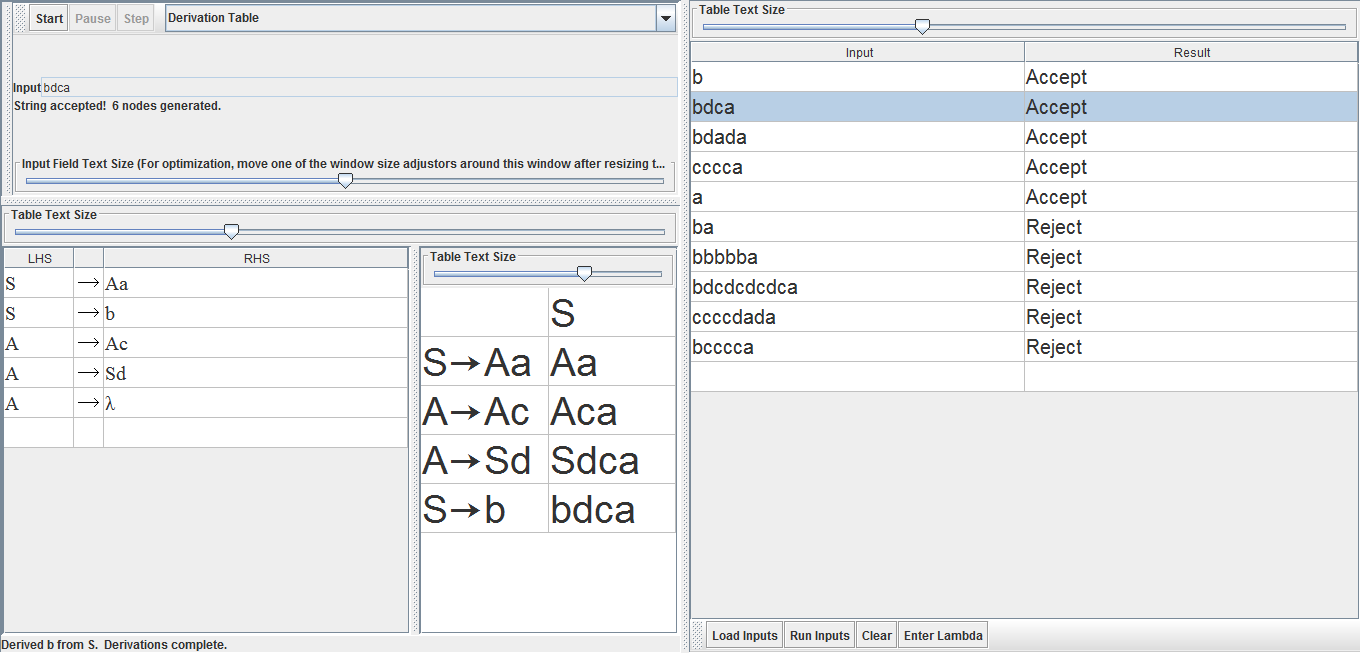
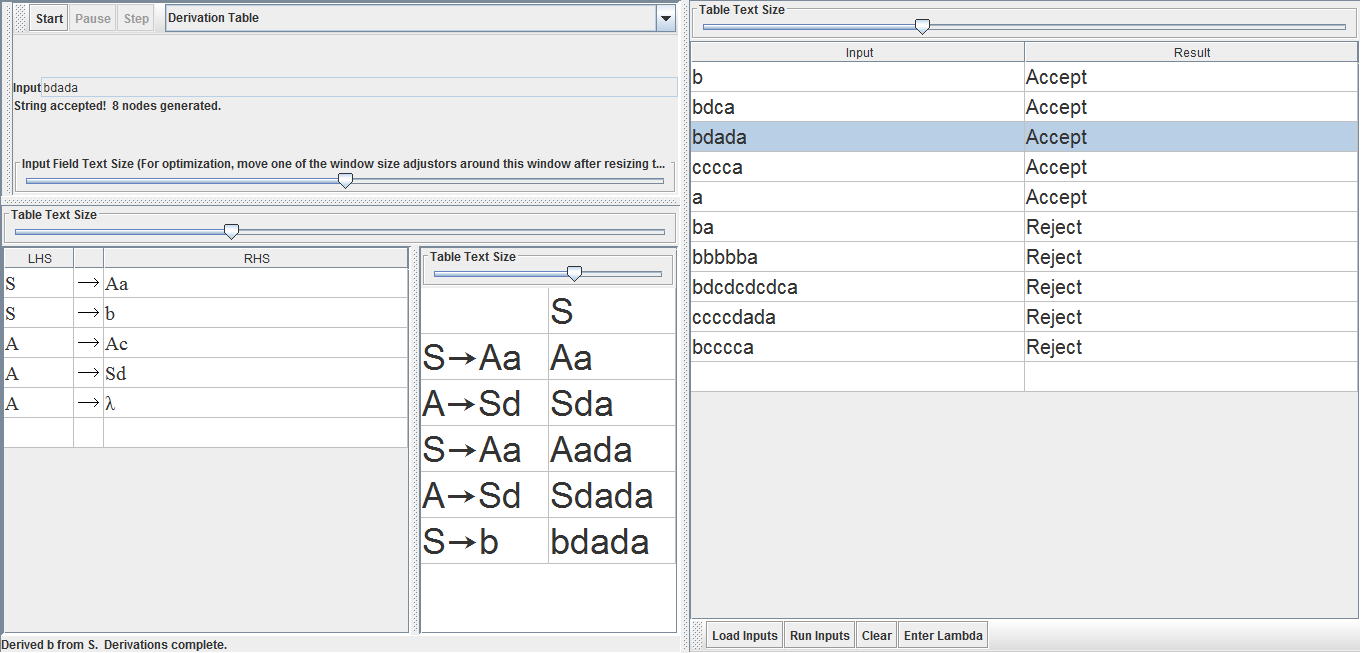
1. L(G2)={01\*,10+,01\*+,10+\*,01+01\*} Tipo 2, No ambigua

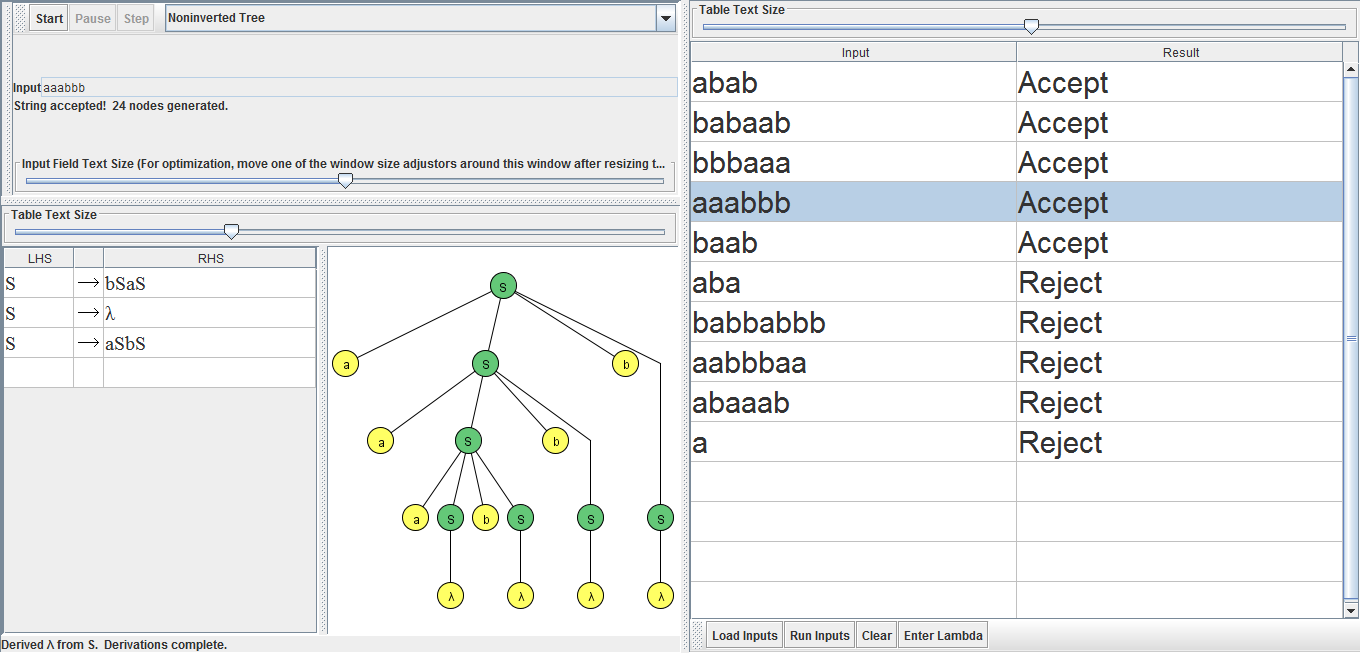


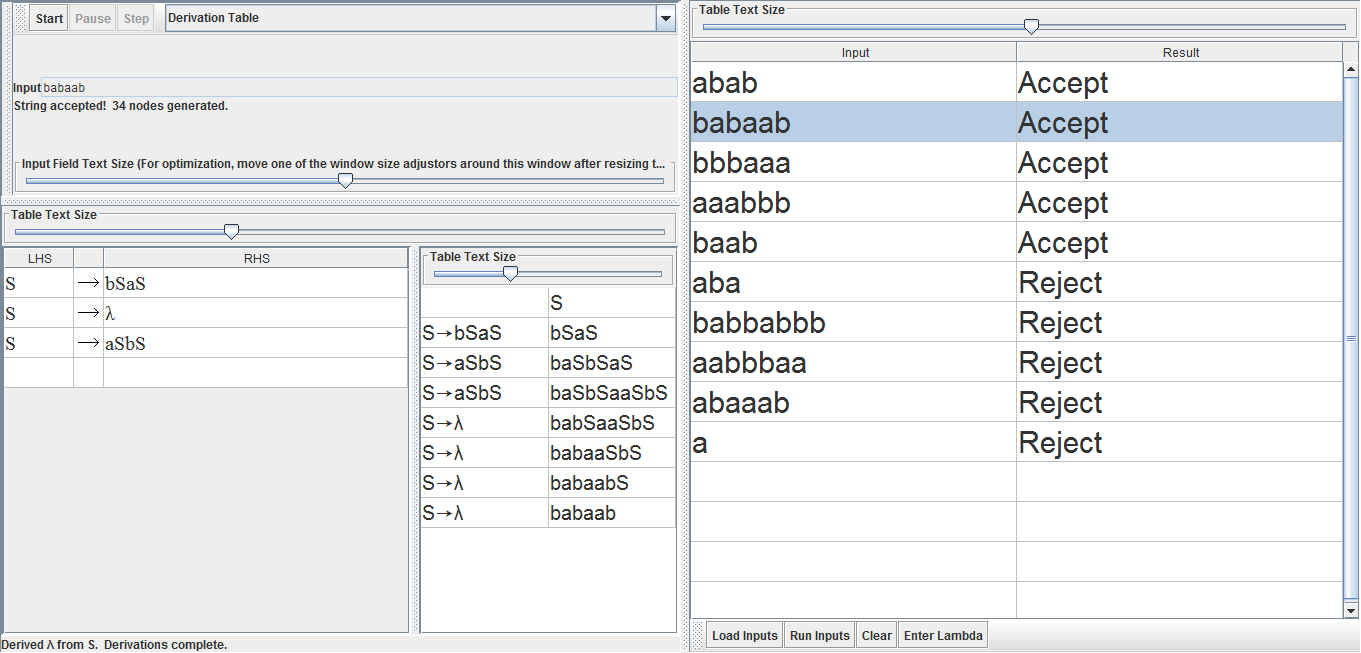
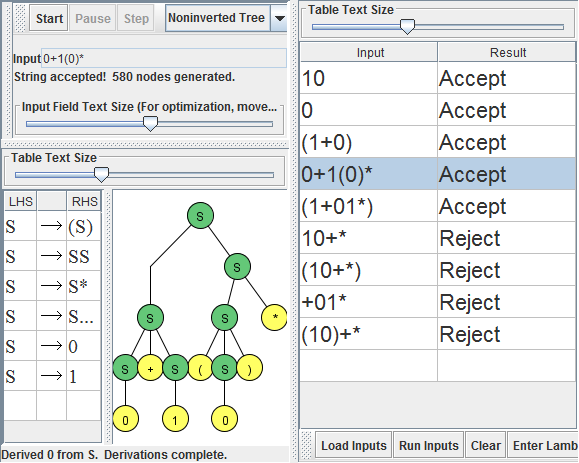
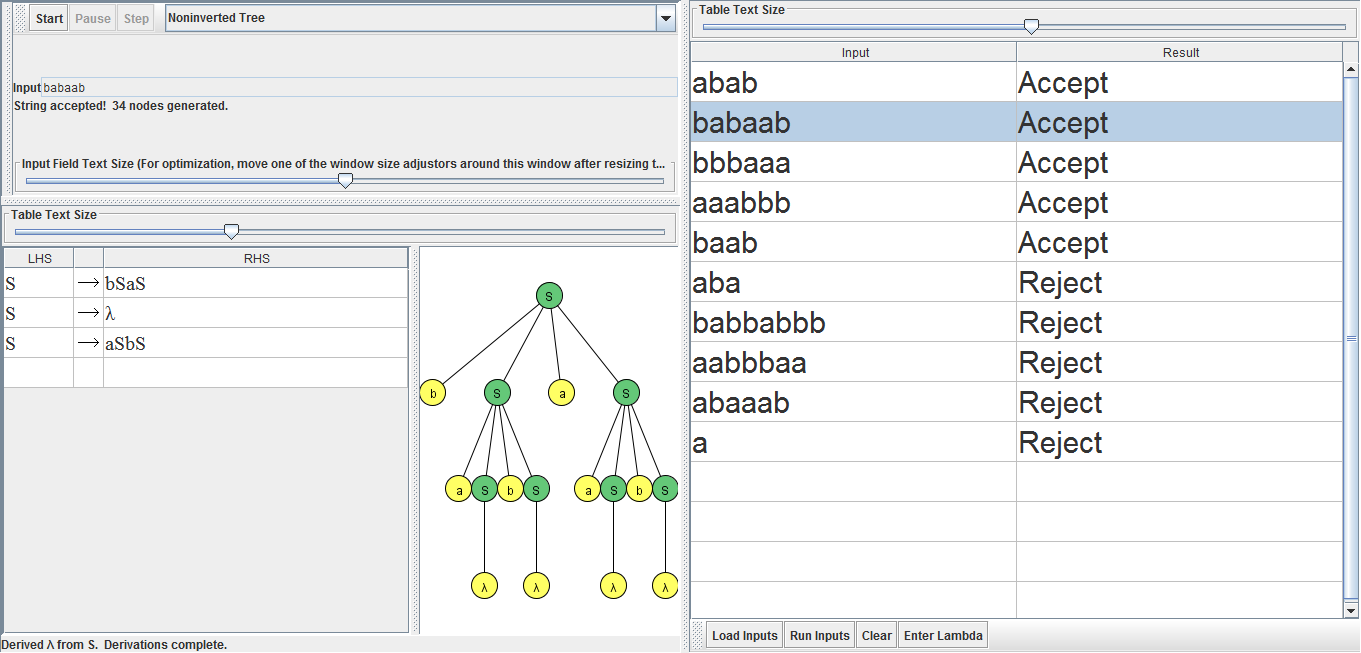
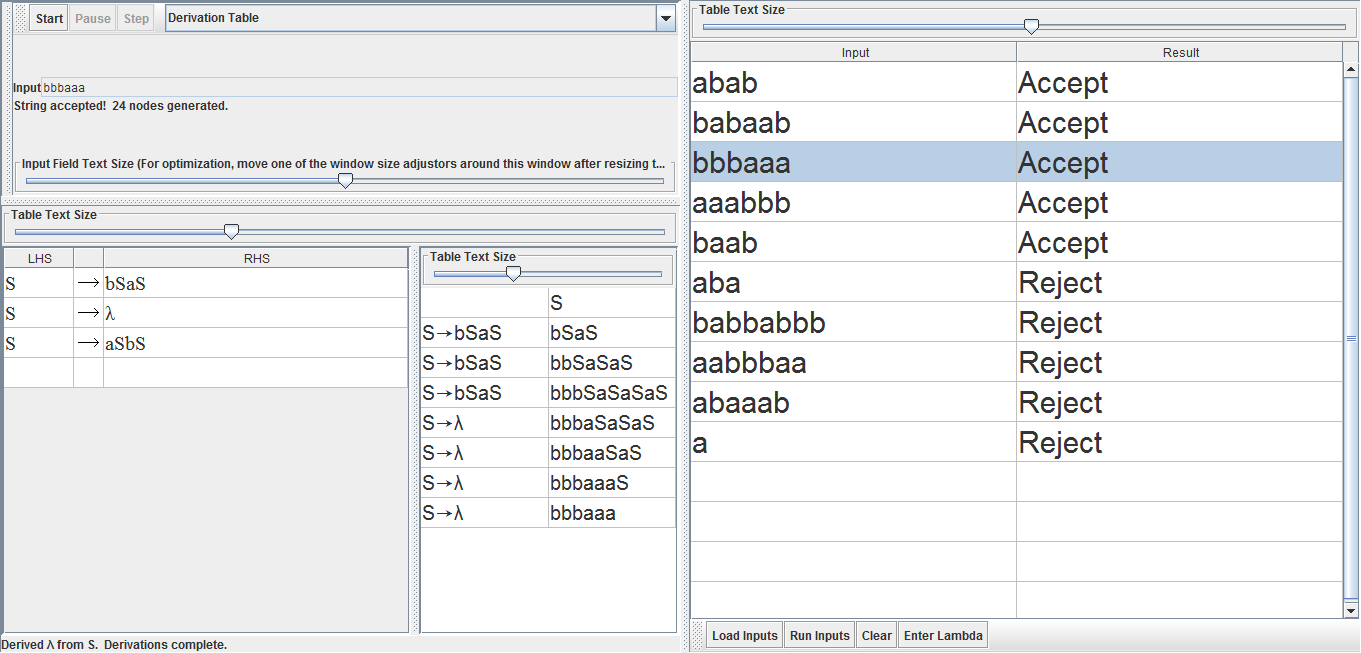
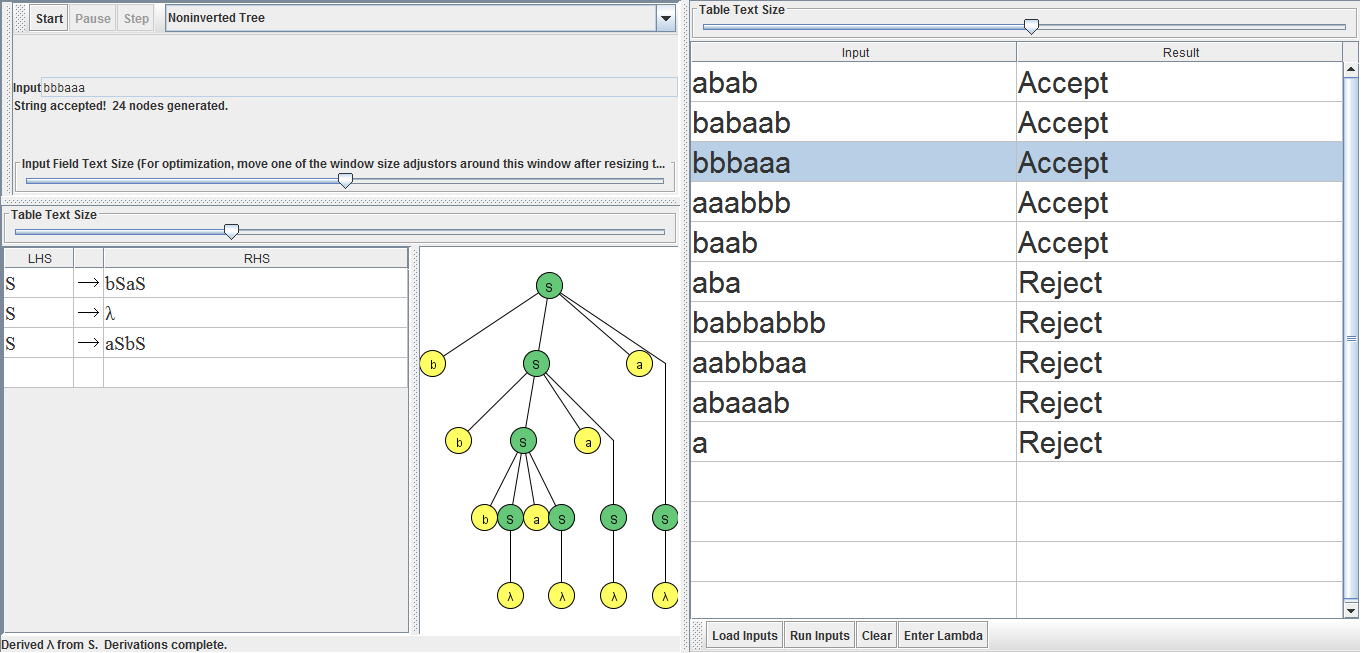
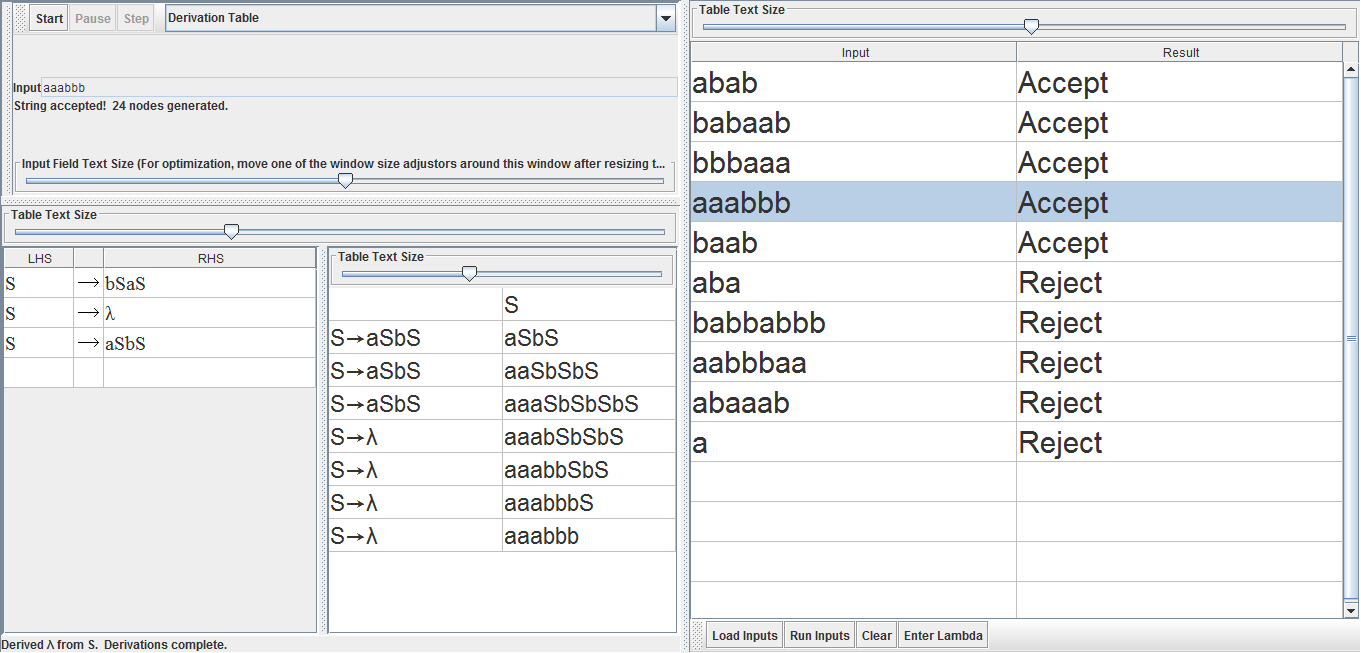
1. L(G3)={(),(()),()(),(())(()),…} Tipo 2, No ambigua



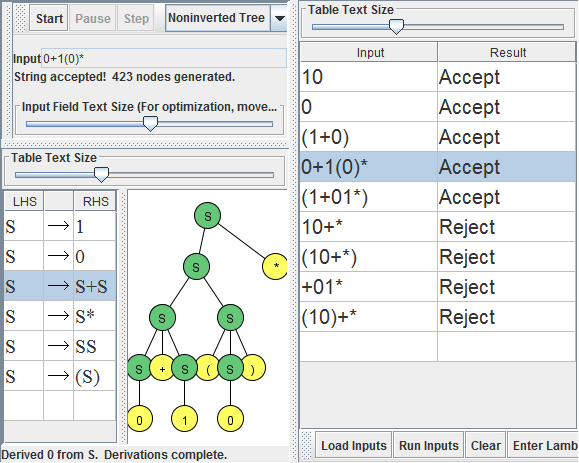
1. L(G4)={ b,a,bdca,cccc…a,bdad…a } Tipo 2, No ambigua

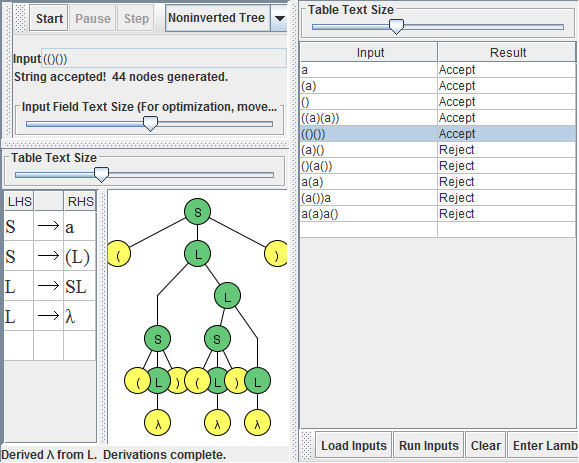


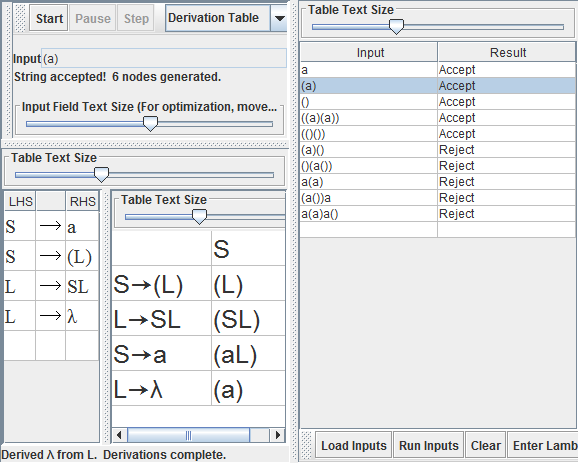
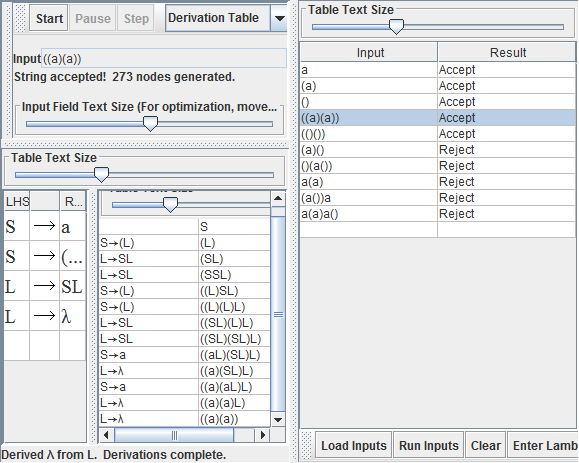
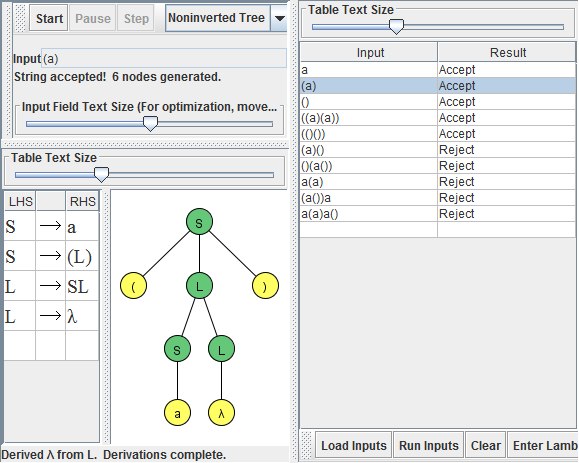
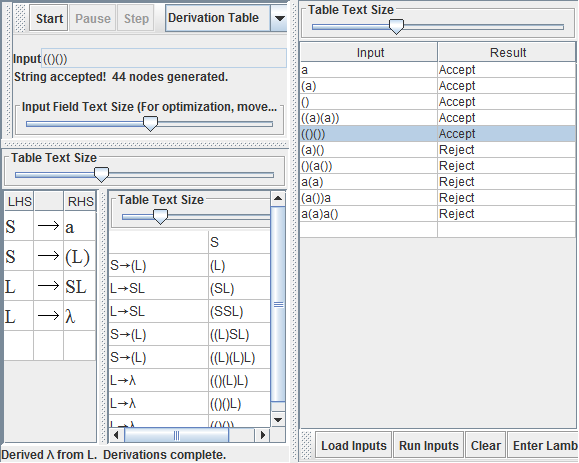
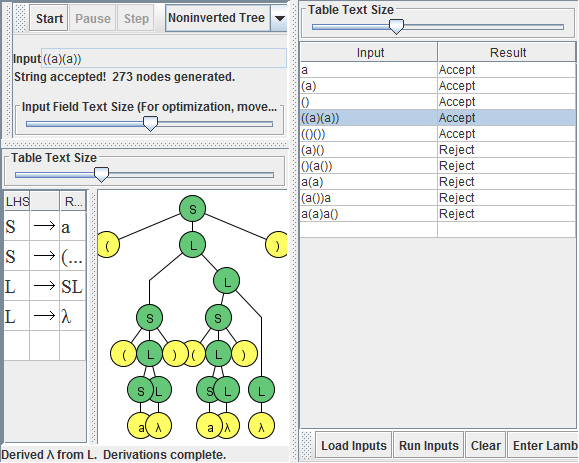
1. L(G5)={ab,abab,ababa,bbaa,…} Tipo 2, No ambigua

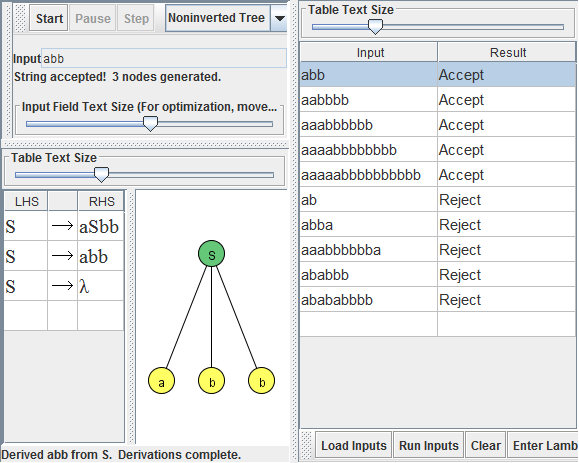


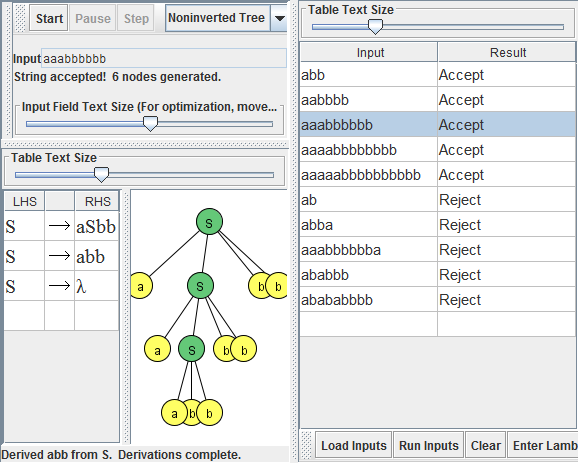
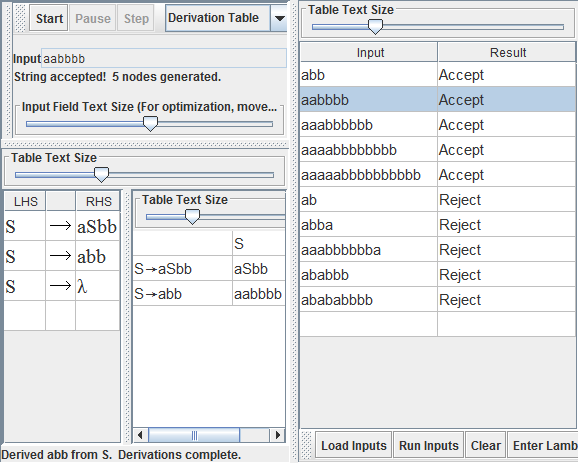
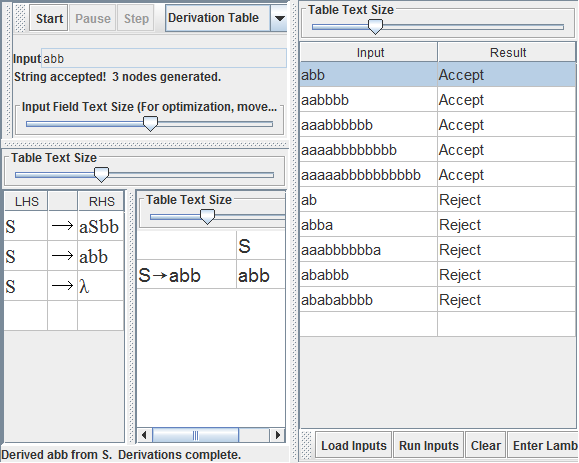
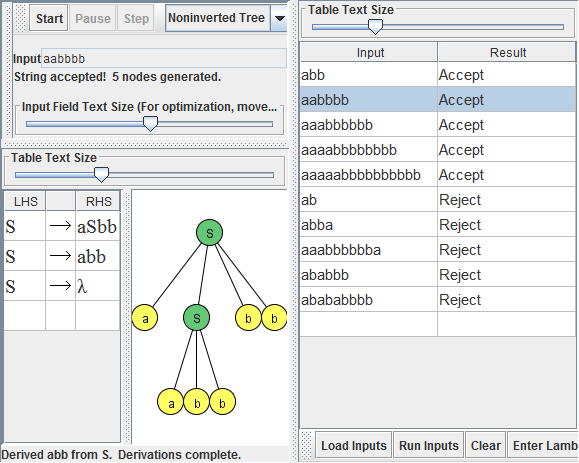
1. L(G6)={0,1,01,011,10,1+0,(0+1),(1+0),(1+01\*)} Tipo 2, Ambigua

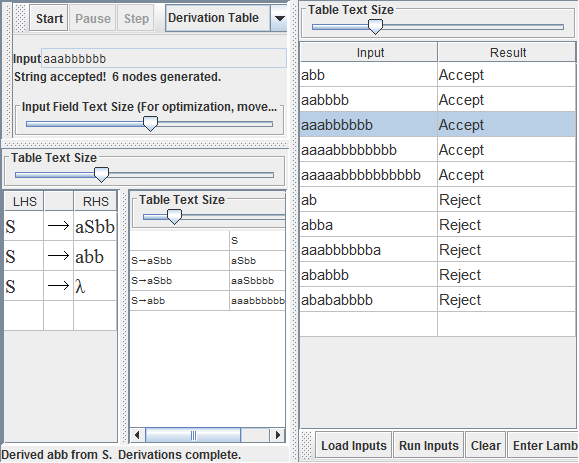


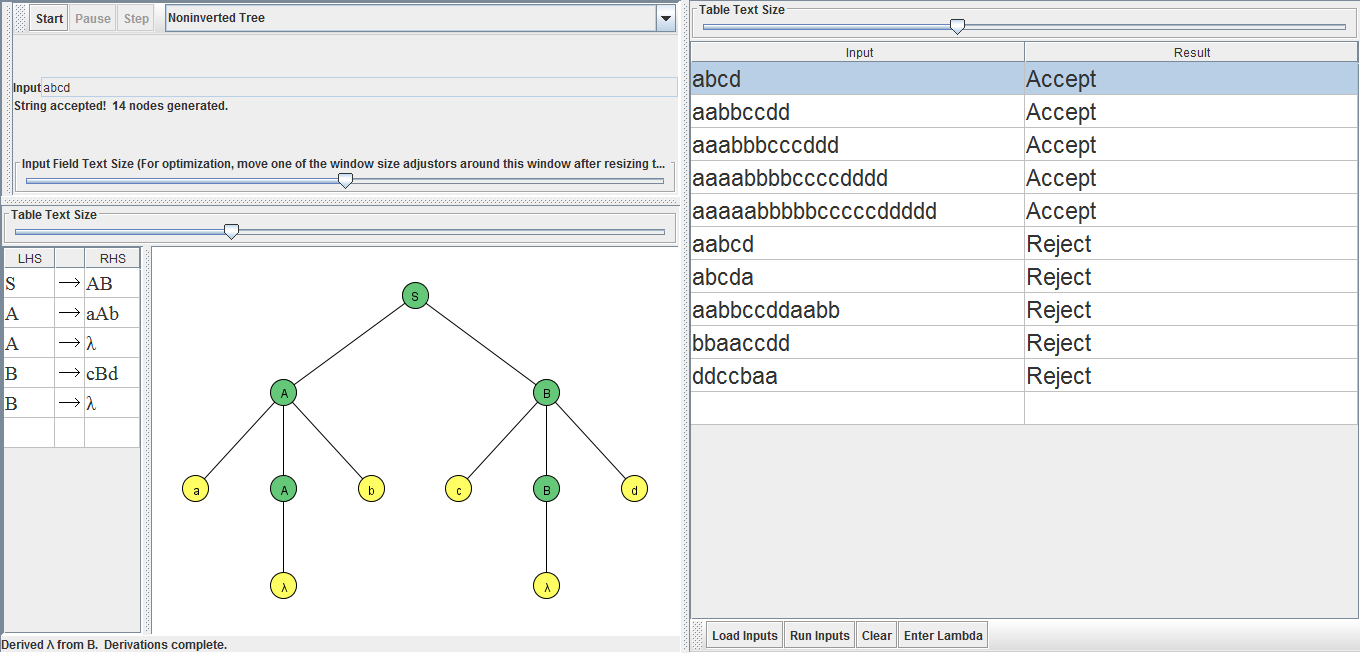
1. L(G7)={(),a,(a),((a)),((a)(a)),… (),(()),…} Tipo 2, No ambigua

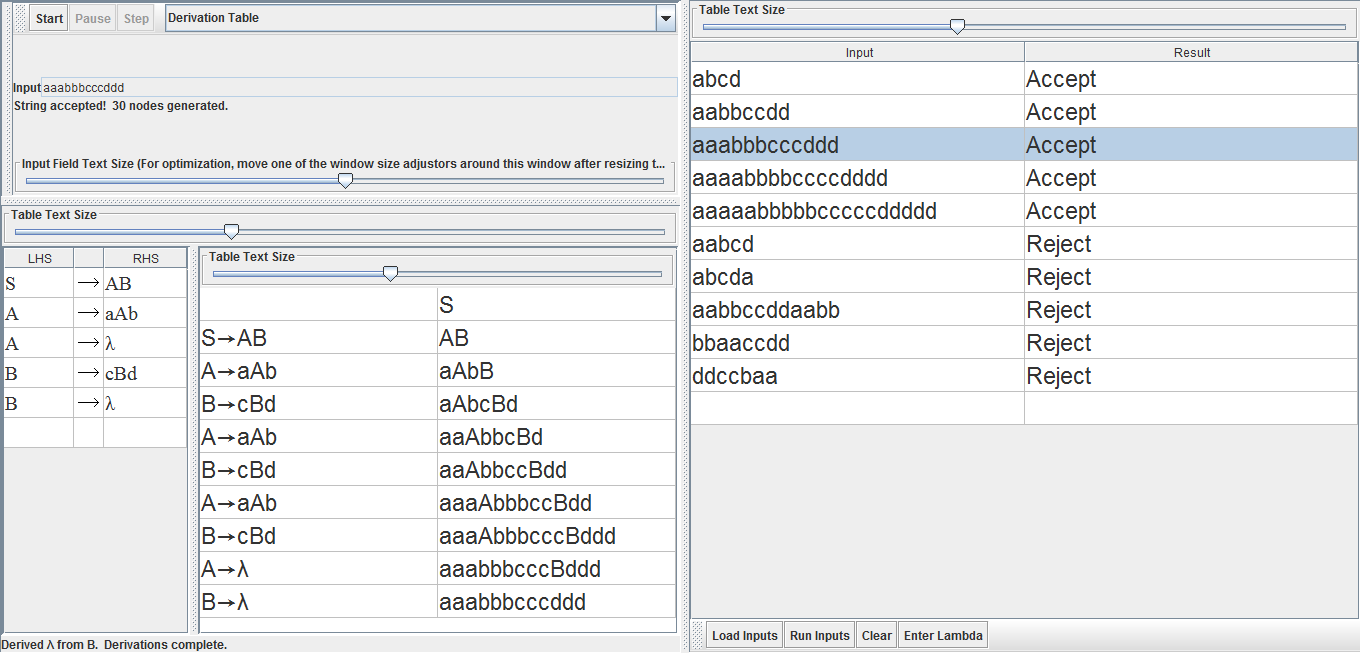
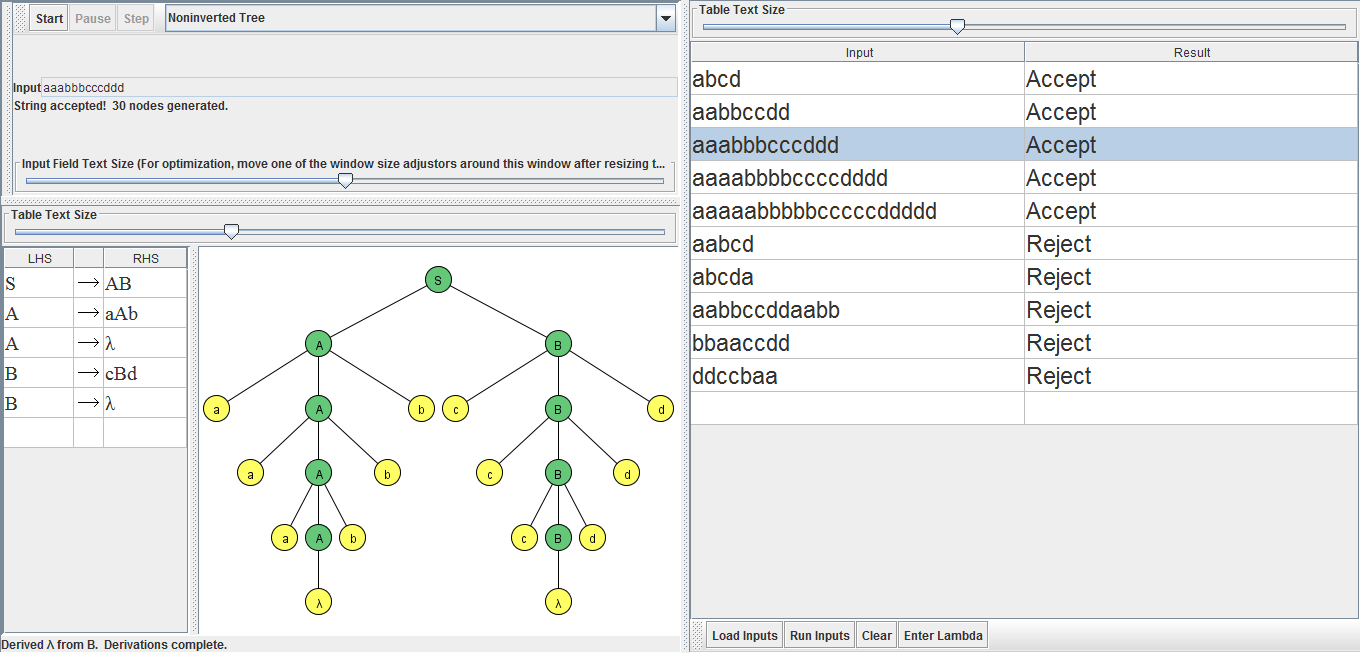
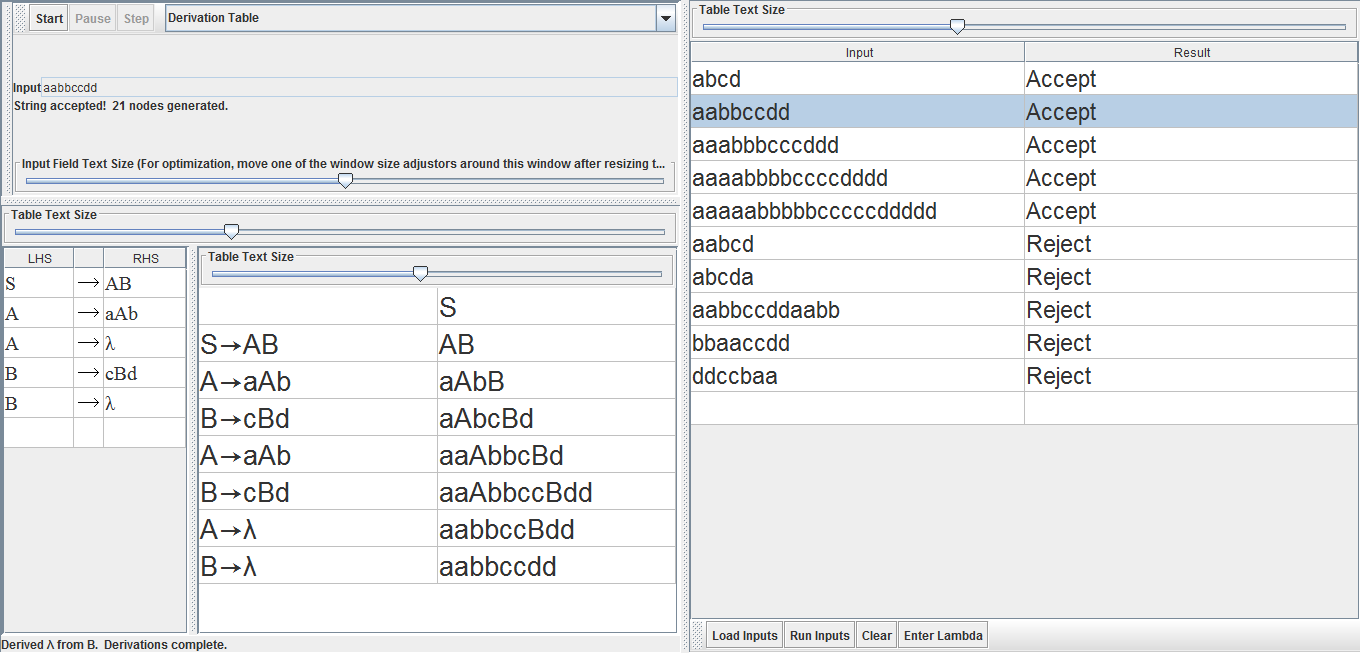
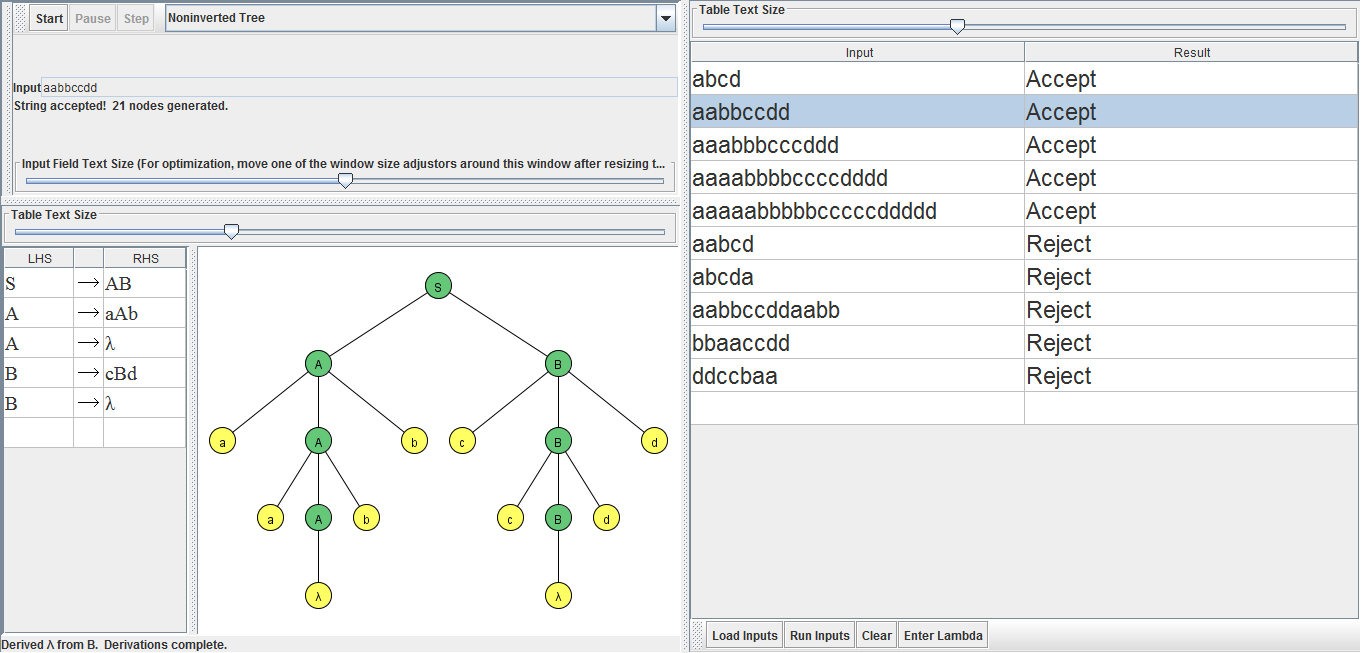
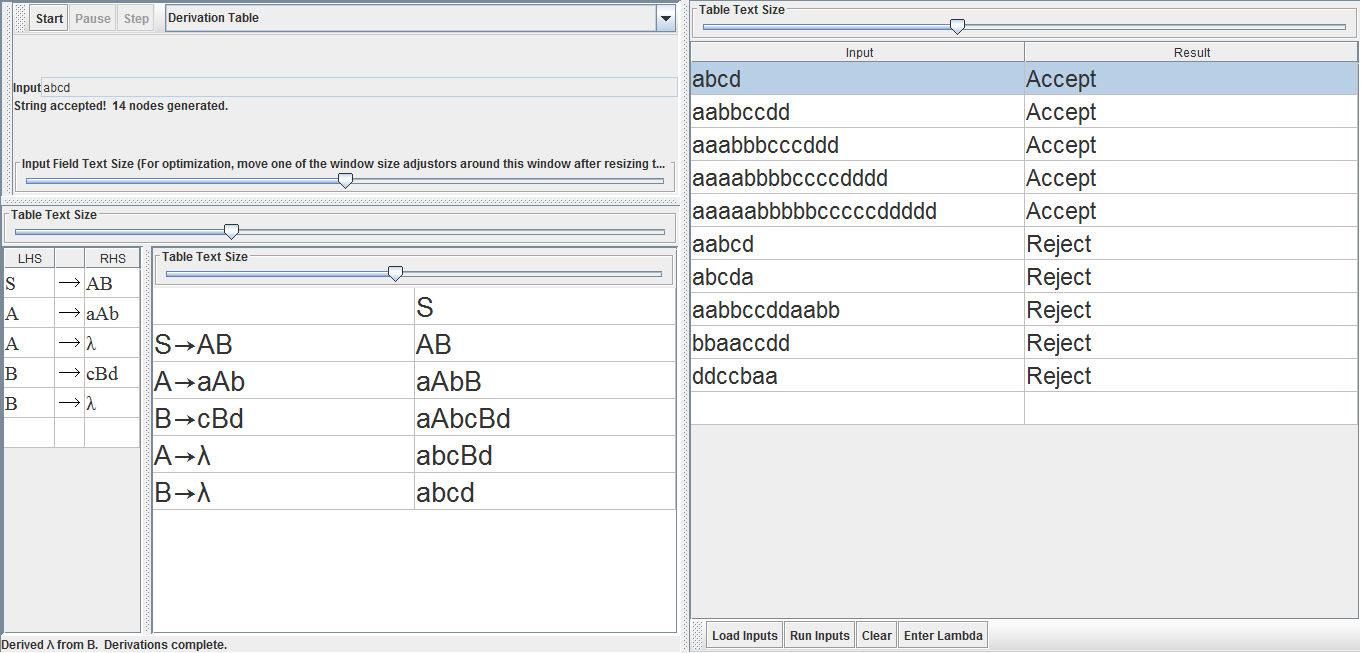


1. L(G8)={anbm | m=2n, m,n=0,1,2,…} Tipo 2, No ambigua

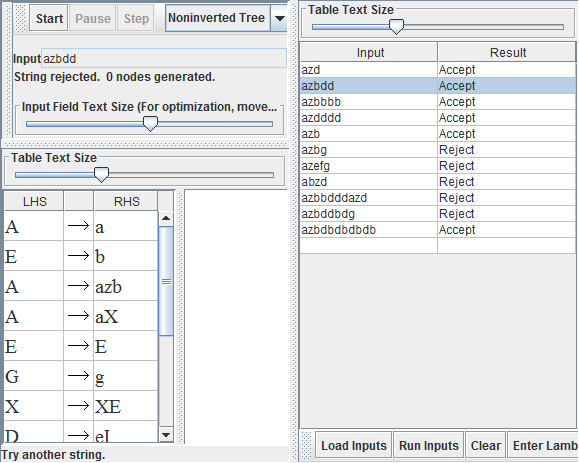


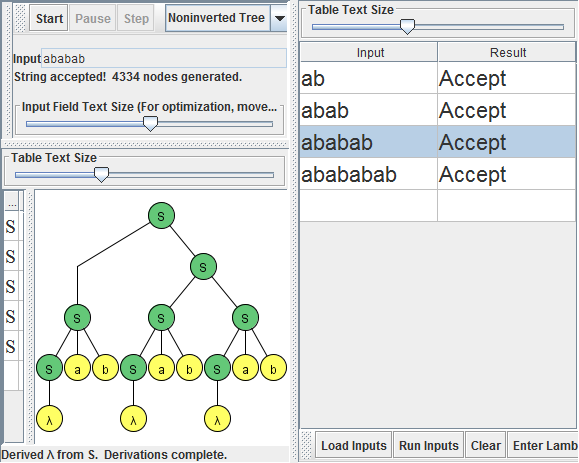
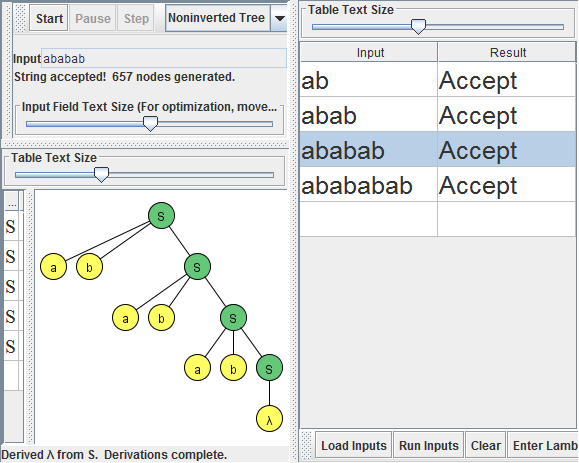
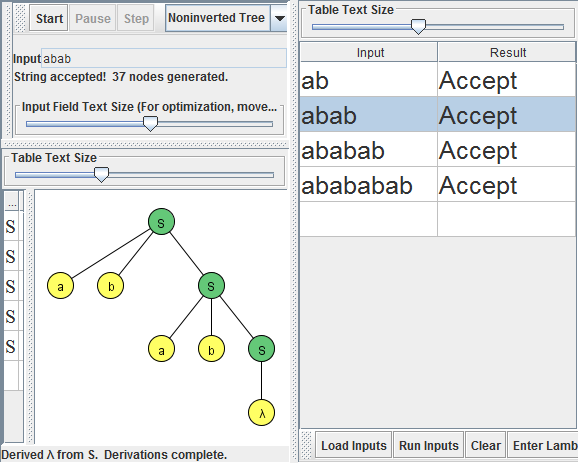
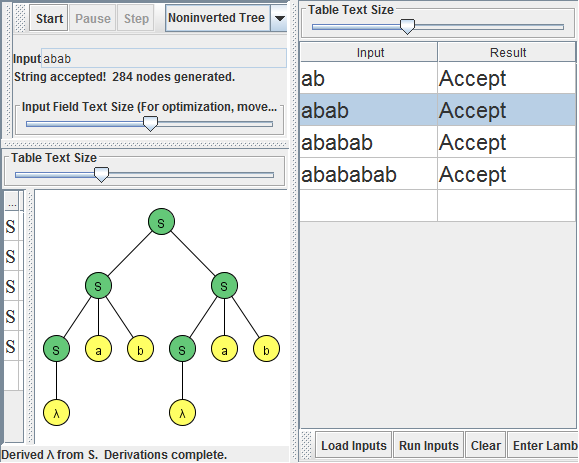


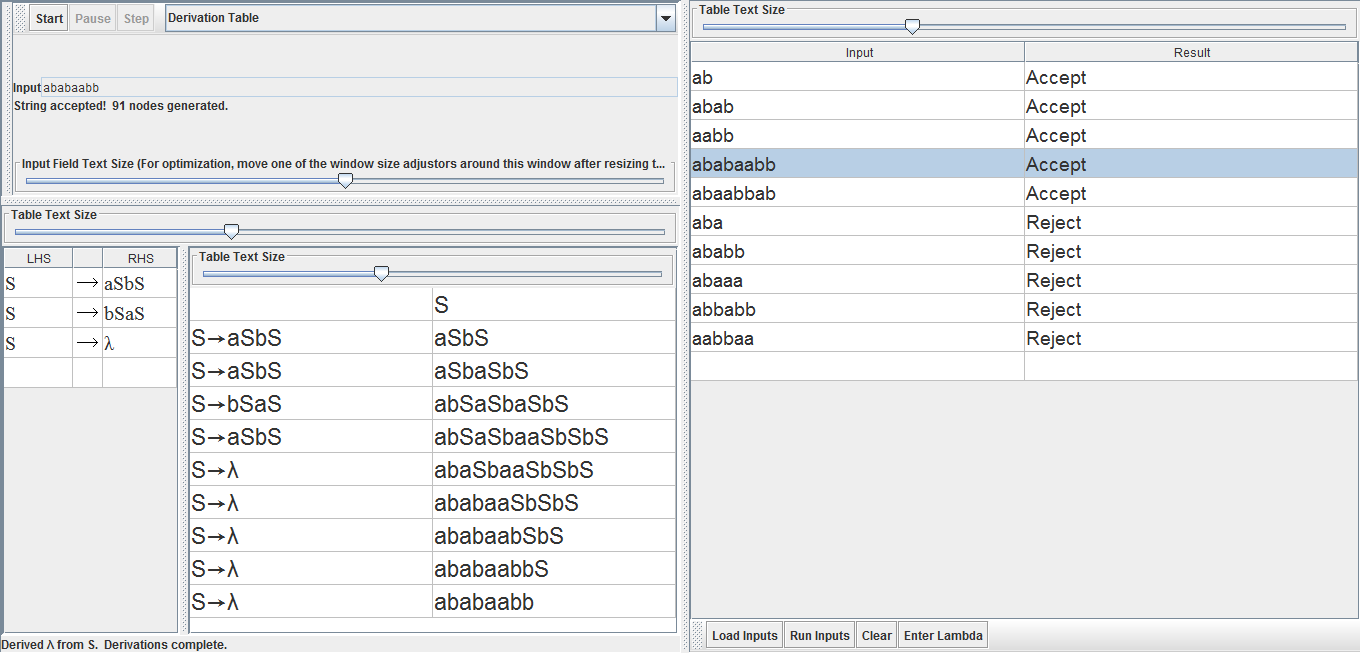
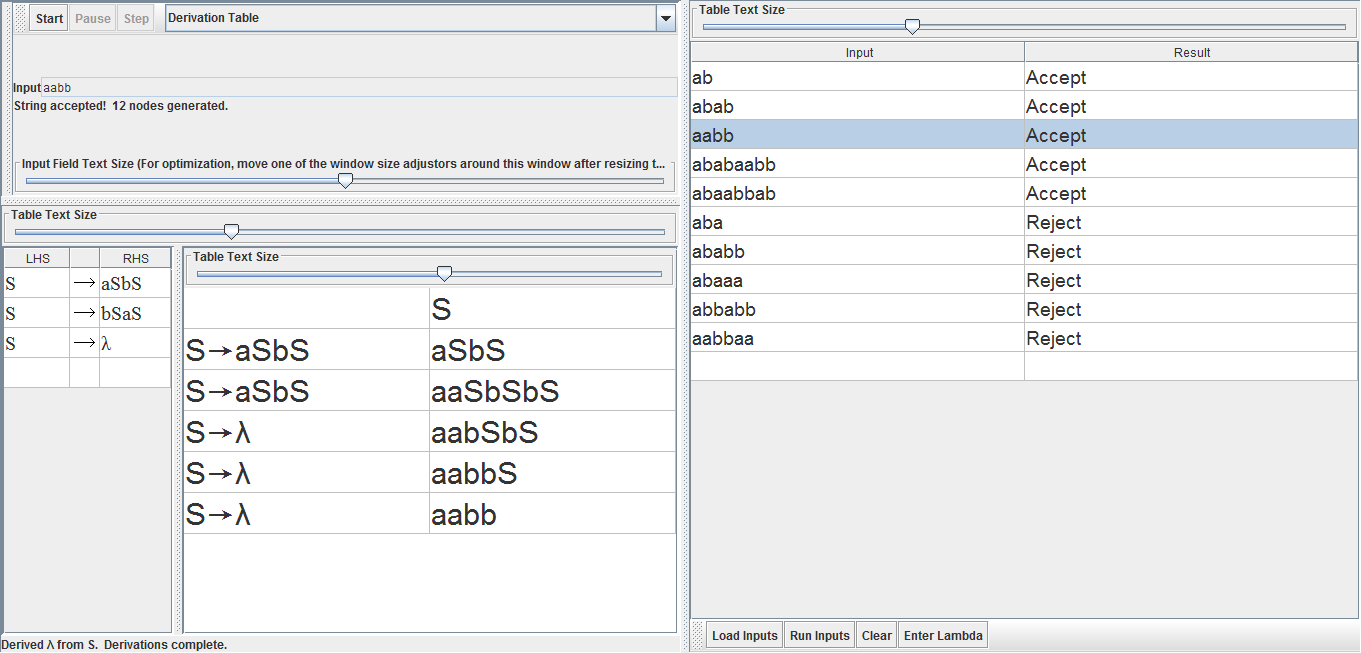
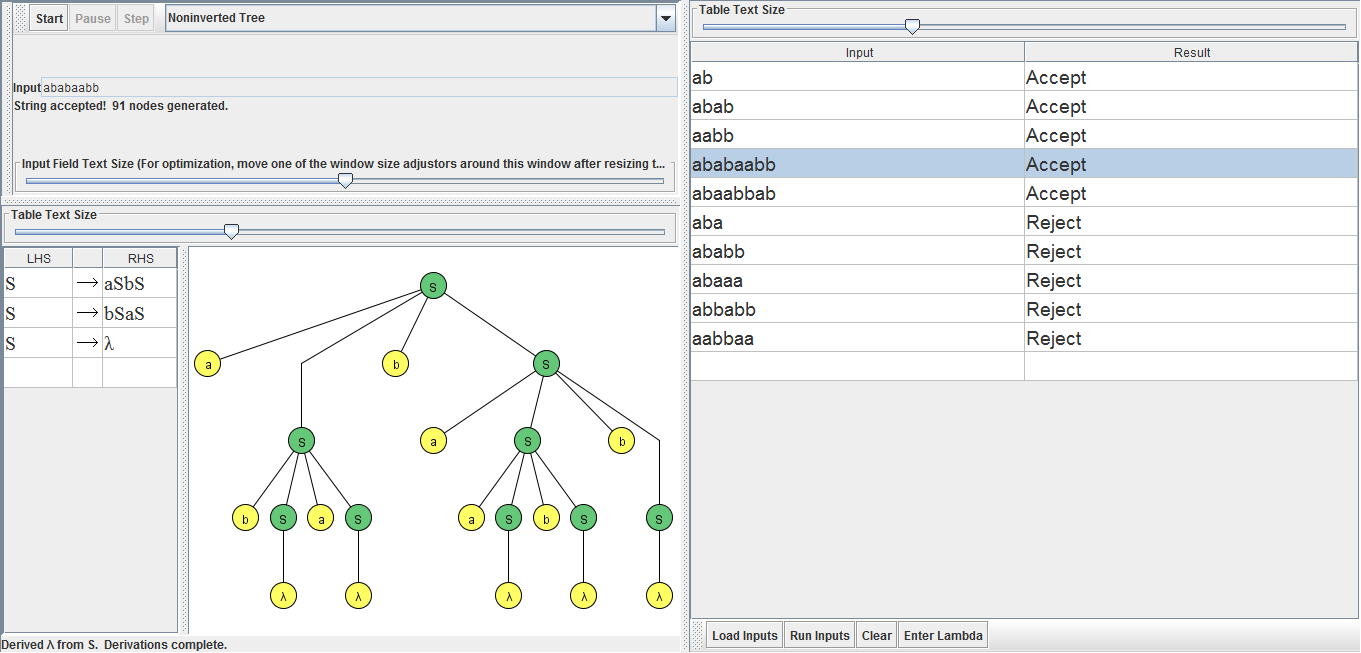
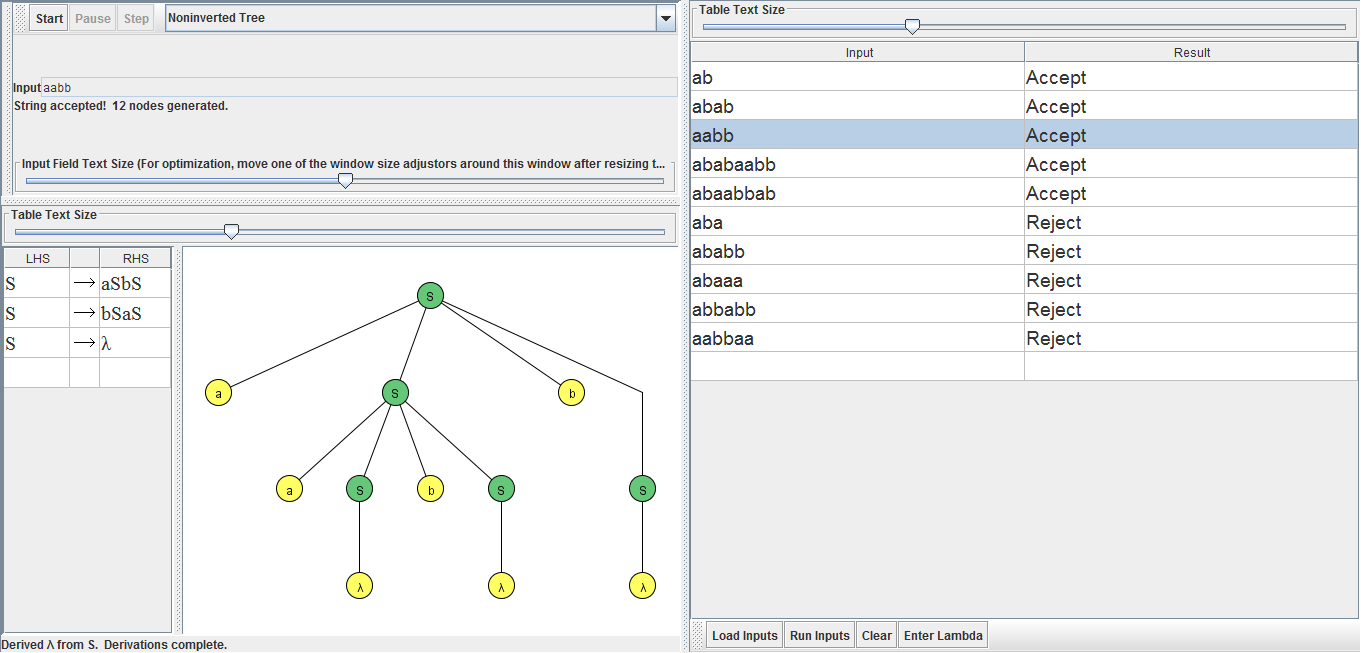
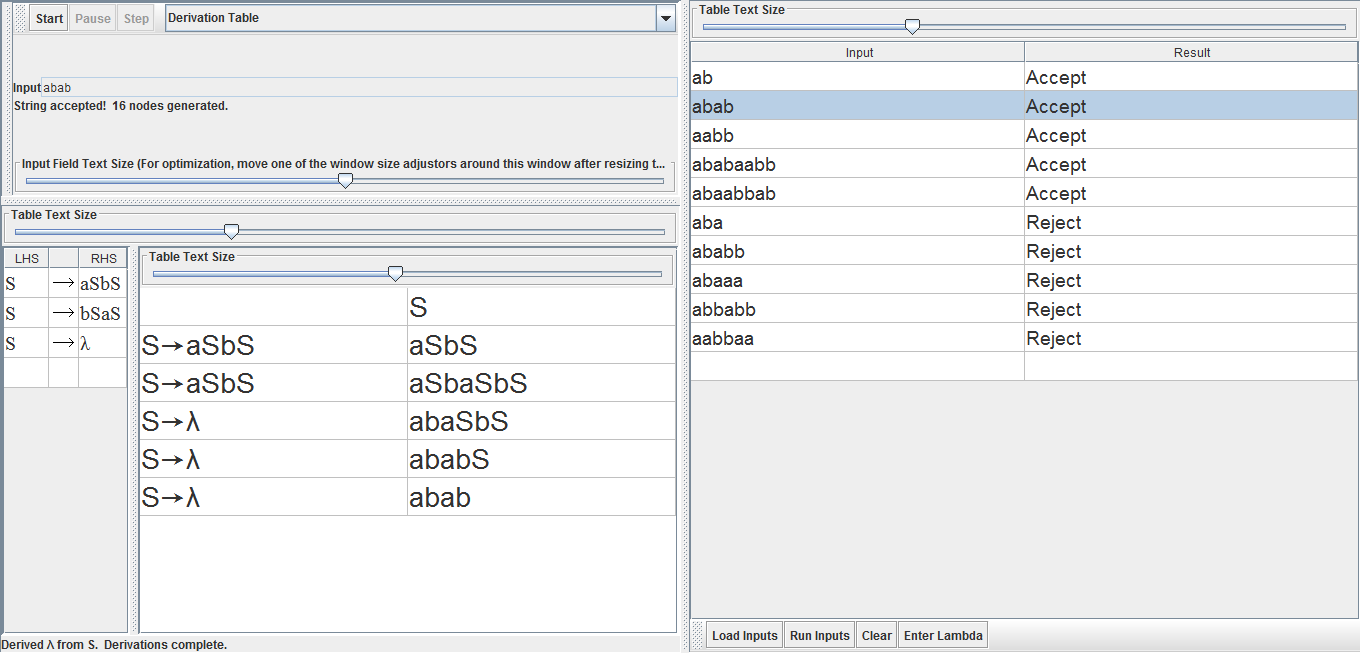
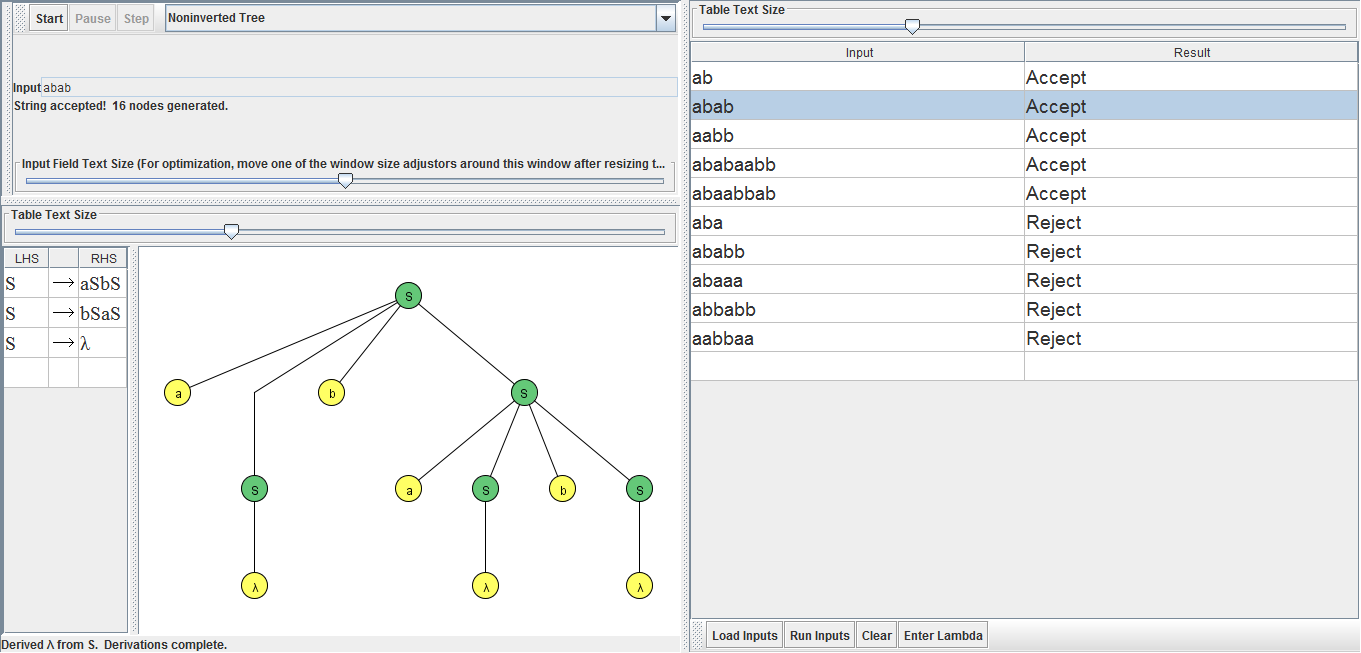
1. L(G9)={ anbm ci,dj | n=m=i=j=0,1.2,…} Tipo 2, No ambigua



1. L(G10)={a,azbn,azdn,azbndm,azbndmbn,… | n diferente de m>0} Tipo 2, Ambigua



1. L(G11)={(ab)n,anbm} Tipo 2, Ambigua
2. L(G12)={anbb,(ab)n} Tipo 2, No ambigua



CONCLUSIÓN

En ésta práctica se realizaron ejercicios con diferentes gramáticas para probar su ambigüedad y definir un lenguaje para estas gramáticas, asignando cadenas válidas y no válidas y comprobar su árbol de derivación para dichas cadenas.

Personalmente ésta práctica fue muy laboriosa porque fueron muchos ejercicios y muchas cadenas a probar, y a mi parecer el propósito de esto es que al momento de hacer cualquier ejercicio nos demos cuenta con facilidad de cada uno de éstos elementos y saberlos identificar con más facilidad en cualquier ejercicio.

BIBLIOGRAFÍA

1. <http://webdiis.unizar.es/asignaturas/LGA/material_2003_2004/4_LIC_GIC_ADPND.pdf>
2. <http://teodelacomp.blogspot.com/2011/03/arboles-de-derivacion.html>
3. <http://ri.uaemex.mx/oca/view/20.500.11799/34043/1/secme-16209.pdf>