**Marco teórico**

**Gramática Independiente del contexto**

* ΣN - un conjunto de símbolos no terminales (variables)
* ΣT - un conjunto de símbolos terminales
* P - un conjunto de reglas de producción
* S∈ΣN un símbolo no terminal de ΣN

La cuaterna (ΣT, ΣN, S, P) se llama gramática independiente del contexto si todas las reglas de producción de P tienen la forma:

A -> β, siendo A∈ΣN y β∈ Σ\*

Todo lenguaje generado por una GIC G (se denota por L(G)) se llama Lenguaje independiente del contexto.

**Gramática ambigua**

La ambigüedad es un problema muy importante en informática. Una gramática es ambigua si existe alguna cadena de terminales que pueda obtenerse mediante árboles de derivación distintos (dos árboles distintos dan la misma cadena)

**PLY (Python Lex-Yacc)**

PLY es una implementación de herramientas de análisis gramatical “lex” y “yacc” para Python.

En resumen, PLY no es nada más que “una forma sencilla de implementar lex/yacc”. Estas son algunas de sus características esenciales:

* Está completamente implementada en Python.
* Utiliza el análisis LR que es razonablemente eficiente y adecuado para gramáticas más grandes.
* PLY proporciona la mayoría de las características estándar de lex/yacc, incluida la compatibilidad con producciones vacías, reglas de precedencia, recuperación de errores y compatibilidad con gramáticas ambiguas.
* PLY es fácil de usar y proporciona una comprobación de errores muy extensa.
* PLY no intenta hacer nada más o menos que proporcionar la funcionalidad lex/yacc básica.

**Lex**

El lex es un generador de programas diseñado para el proceso léxico de cadenas de caracteres de input. El programa acepta una especificación, orientada a resolver un problema de alto nivel para comparar literales de caracteres, y produce un programa C que reconoce expresiones regulares. Estas expresiones las especifica el usuario en las especificaciones fuente que se le dan al lex. El código lex reconoce estas expresiones en una cadena de input y divide este input en cadenas de caracteres que coinciden con las expresiones. En los bordes entre los literales, se ejecutan las secciones de programas proporcionados por el usuario. El fichero fuente lex asocia las expresiones regulares y los fragmentos de programas. Puesto que cada expresión aparece en el input del programa escrito por el lex, se ejecuta el fragmento correspondiente.

El usuario proporciona el código adicional necesario para completar estas funciones, incluyendo código escrito por otros generadores. El programa que reconoce las expresiones se genera en forma de fragmentos de programa C del usuario.

El lex no es un lenguaje completo sino un generador que representa una cualidad de un nuevo lenguaje que se añade al leguaje de programación C. El lex convierte las expresiones y acciones del usuario (llamadas fuente en este capítulo) en un programa C llamado yylex. El programa yylex reconoce expresiones en un flujo (llamado input en este capítulo) y lleva a cabo las acciones especificadas para cada expresión a medida que se va detectando.

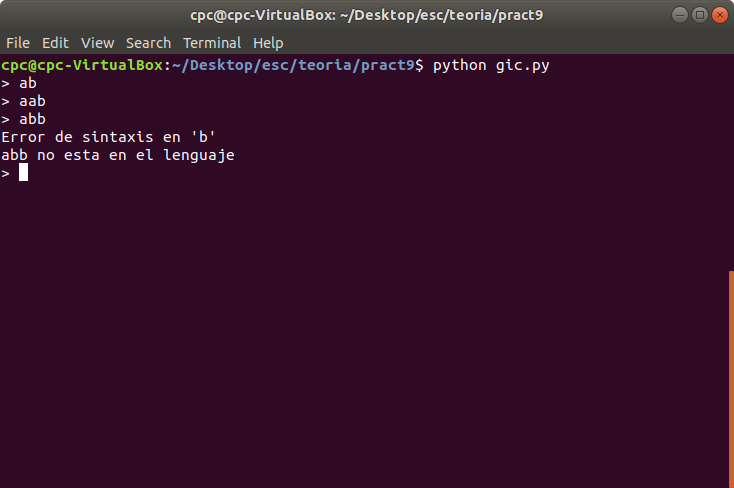
**Yacc**

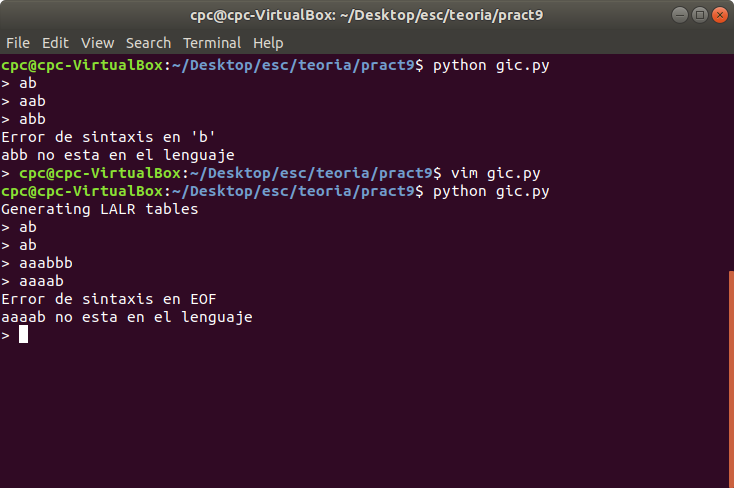
Yacc es un programa para generar analizadores sintácticos. Las siglas del nombre significan Yet Another Compiler-Compiler, es decir, "Otro generador de compiladores más". Genera un analizador sintáctico (la parte de un compilador que comprueba que la estructura del código fuente se ajusta a la especificación sintáctica del lenguaje) basado en una gramática analítica escrita en una notación similar a la BNF. Yacc genera el código para el analizador sintáctico en el Lenguaje de programación C.

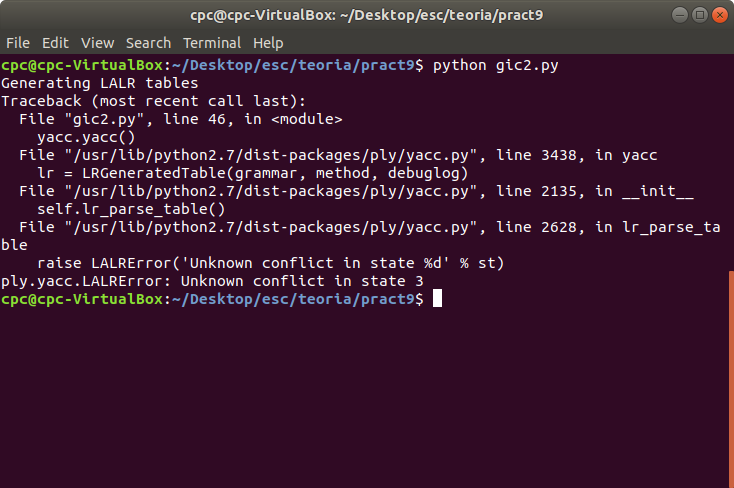
**Material y equipo**

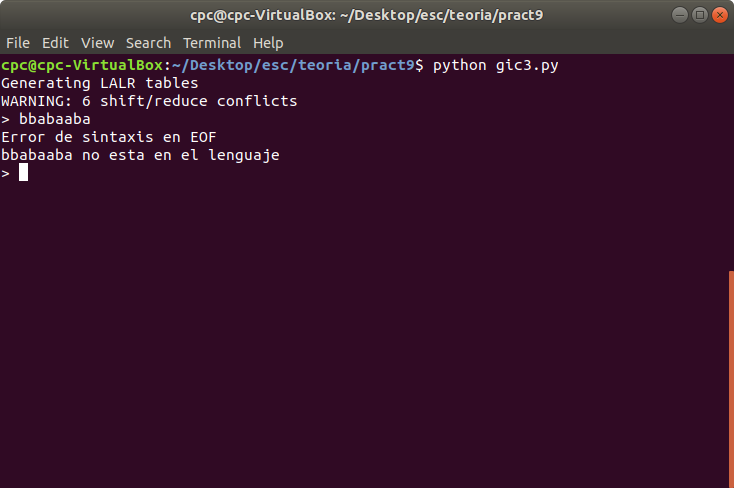
1. Computadora con sistema operativo Windows o cualquier distribución de Linux.
2. Python previamente instalado

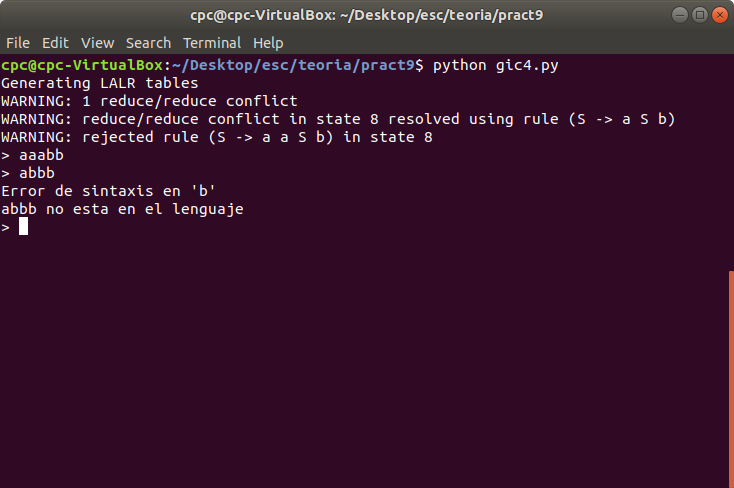
**Desarrollo**









1. 

**Conclusión**

La práctica más complicada que he realizado hasta el momento en laboratorio, no logré comprender la precedencia que se utiliza en el ejemplo de la calculadora hecha con PLY, la ambigüedad es un problema que nos complicó mucho el trabajo en esta práctica.

**Bibliografía**

Ambigüedad. Árboles de derivación. Gramáticas ambiguas

http://antares.sip.ucm.es:8180/webtalf/index.jsp?submenu=temas/submenuIncontextuales&contenido=temas/ambiguedad

Capítulo 10.

Gramáticas independientes del contexto

<http://www.ia.urjc.es/grupo/docencia/automatas_itis/apuntes/capitulo10.pdf>