Proyecto Final: Analizadores LL(1) y SLR(1)

Universidad EAFIT – Lenguajes Formales

Mariamny del Valle Ramírez Telles
Paulina Velásquez Londoño

Mayo de 2025

1. Introducción

Este documento describe la implementación de un sistema que permite analizar gramáticas libres de contexto mediante los algoritmos de análisis sintáctico LL(1) y SLR(1), como parte del proyecto final de la materia Lenguajes Formales. El objetivo fue construir herramientas que permitan, a partir de una gramática arbitraria válida, calcular automáticamente los conjuntos **FIRST** y **FOLLOW**, así como construir las tablas predictivas LL(1) y las tablas de análisis SLR(1). El sistema fue desarrollado en Python, haciendo uso de estructuras de datos como listas, conjuntos y diccionarios, y ejecutado desde terminal en entornos locales.

2. Algoritmos implementados

2.1. Conjuntos FIRST y FOLLOW

Se implementaron los algoritmos descritos por Aho, Lam, Sethi y Ullman en el libro *Compilers: Principles, Techniques and Tools* (2da edición), específicamente de las secciones 4.4 y 4.6. La estructura principal fue una clase llamada `FirstFollow` que calcula los conjuntos **FIRST** y **FOLLOW** mediante iteraciones sucesivas hasta que estos conjuntos dejan de cambiar.

- **FIRST**: contiene todos los terminales que pueden iniciar una cadena derivada de un no terminal.
- **FOLLOW**: contiene los terminales que pueden seguir a un no terminal en alguna derivación.

2.2. Tabla LL(1)

Con base en los conjuntos FIRST y FOLLOW, se construyó la tabla de análisis LL(1), asociando cada producción a su posición correspondiente (no terminal, terminal).

2.3. Autómata LR(0) y Tabla SLR(1)

Se construyó el conjunto de elementos LR(0) mediante los algoritmos de `closure` y `goto`. Luego, se generó la tabla SLR(1) con las acciones de desplazamiento (shift), reducción (reduce), aceptación y transiciones GOTO, usando los conjuntos FOLLOW.

2.3. Autómata LR(0) y Tabla SLR(1)

Se construyó un autómata LR(0) generando los conjuntos de items utilizando los algoritmos `closure` y `goto`. A partir de estos estados, se crearon las tablas ACTION y GOTO siguiendo las reglas de los analizadores SLR(1):

- **SHIFT**: se agrega una transición cuando el punto precede a un terminal.
- **REDUCE**: si el punto está al final, se generan reducciones con base en FOLLOW.
- **ACCEPT**: se reconoce la aceptación cuando se encuentra la producción aumentada completa y el símbolo $.

3. Problemas encontrados

Durante el desarrollo del proyecto se presentaron varios retos importantes:

- **Algoritmo SLR(1)**: Tuvimos muchas dificultades para diseñarlo y probarlo correctamente. Se nos presentaban errores lógicos, errores de estructura de datos y errores en la construcción de los estados. Fue necesario verificar cuidadosamente con ejemplos del libro y de la guía para validar que cada paso de la construcción era correcto.
- **Visualización de tablas**: Al principio, las tablas de LL(1) y SLR(1) se veían desorganizadas en consola. Las columnas no estaban alineadas, lo que dificultaba su lectura. Tuvimos que rediseñar la impresión con espacios fijos y formateo personalizado para lograr una presentación clara y ordenada.
- **Entrada de la gramática**: Uno de los obstáculos fue encontrar un formato de entrada para la gramática que no generara errores. Notamos que escribir las producciones sin espacios hacía que la tabla se construyera mal. Al ingresar cada símbolo separado por espacio, se logró una lectura correcta y una tabla bien organizada. Por otra parte, también notamos que era muy necesario, al momento de ingresar la gramatica, escribir '—' (línea vertical que separa las diferentes producciones), ya que si no escribiamos esto, al momento de validar las cadenas, todas nos decían que no eran válidas, cuando en la guía aparecían que sí eran válidas.
- **Problemas con 'e'**: Otro gran problema que tuvimos fue el manejo del e (épsilon). El símbolo no se reconocía correctamente porque el programa lo trataba como un carácter literal 'e' en lugar de interpretarlo como una producción vacía. Lo pudimos arreglar ajustando la lógica para que ['e'] se ignore en el análisis (parse) y se maneje correctamente en la construcción de los conjuntos FIRST, FOLLOW y tablas LL(1)/SLR(1).

4. Instrucciones de ejecución

1. Instalar Python 3 (si no está instalado).
2. Clonar o descargar el proyecto.
3. Ejecutar el archivo principal desde consola con el siguiente comando:

```
python3 main.py  o  Python–m main
```

4. La salida mostrará en consola:
    - Los conjuntos FIRST y FOLLOW.
    - La tabla LL(1) organizada por filas y columnas.
    - La tabla SLR(1) con las acciones `shift`, `reduce`, `accept` y transiciones GOTO.

5. Conclusiones

Este proyecto nos permitió aplicar de forma práctica los algoritmos de análisis sintáctico estudiados en clase. Pese a las dificultades iniciales con la gramática aumentada, en el parsing y en la validación de las cadenas, logramos construir analizadores funcionales que cumplen con los requerimientos del trabajo.

Se logró una implementación clara y funcional. Las mejoras en la experiencia del usuario, para que el trabajo se viera más presentable, y la validación de errores hacen que el sistema sea intuitivo para pruebas. Este proyecto refuerza la importancia de la teoría formal en la implementación de lenguajes de programación

6. Referencias

- Aho, A.V., Lam, M.S., Sethi, R., Ullman, J.D. (2006). *Compilers: Principles, Techniques, and Tools*. Pearson, 2da edición.