Lenguajes Formales

Proyecto Final

Estudiantes: Hever Andre Alfonso Jimenez y Moises Vergara Garces

Profesor: Adolfo Andres Castro Sanchez

Mayo 2025

Introducción

Durante el desarrollo del proyecto, uno de los mayores retos fue entender y aplicar correctamente los algoritmos para calcular los conjuntos **FIRST** y **FOLLOW**, así como implementar un **parser LL(1)** basado en tabla. Aunque las definiciones teóricas son claras, llevarlas a código y aplicarlas a gramáticas reales nos generó confusión en algunos puntos. Aquí explicamos cómo resolvimos estos problemas.

Cálculo del conjunto FIRST

El conjunto **FIRST**(α) de una cadena α es el conjunto de terminales que pueden aparecer como primer símbolo en alguna derivación de α. Si α puede derivar ε, entonces $\varepsilon \in$ FIRST(α).

Regla práctica:

$$\text{FIRST}(\alpha\beta) = (\text{FIRST}(\alpha) \setminus \{\varepsilon\}) \cup \begin{cases} \text{FIRST}(\beta), & \text{si } \varepsilon \in \text{FIRST}(\alpha) \\ \emptyset, & \text{si } \varepsilon \notin \text{FIRST}(\alpha) \end{cases}$$

Implementamos esto con la función `first_of_string` que recorre los símbolos de la cadena y construye el conjunto correspondiente, manejando correctamente la aparición de ε.

Cálculo del conjunto FOLLOW

El conjunto **FOLLOW**(A) de un no terminal A es el conjunto de terminales que pueden aparecer justo después de A en alguna derivación. También se incluye el símbolo \$ en el FOLLOW del símbolo inicial.

Regla práctica: Si hay una producción de la forma:

$$A \to \alpha B \beta$$

Entonces:

- Agregar FIRST(β) $\setminus$ $\{\varepsilon\}$ a FOLLOW(B)
- Si $\varepsilon \in$ FIRST(β) o $\beta = \varepsilon$, entonces agregar FOLLOW(A) a FOLLOW(B)

Esto lo resolvimos iterando hasta que los FOLLOW no cambien más, como sugiere el algoritmo estándar.

Construcción de la tabla LL(1)

La tabla LL(1) indica qué producción usar al ver un par (no terminal, símbolo de entrada). Para construirla:

- Para cada producción $A \to \alpha$, para cada $a \in$ FIRST(α) $- \{\varepsilon\}$, ponemos $A \to \alpha$ en `table[A][a]`.
- Si $\varepsilon \in$ FIRST(α), agregamos la producción en todas las entradas $b \in$ FOLLOW(A).

La tabla se implementa como un diccionario anidado `table[A][a]`.

Parsing con tabla LL(1)

Una vez construida la tabla LL(1), el parsing se hace con una pila. Se inicializa con \$ y el símbolo inicial. Luego:

- Si el tope de la pila es un terminal y coincide con el símbolo de entrada, se consume.
- Si es un no terminal A, se consulta `table[A][a]` y se reemplaza por el lado derecho (en orden inverso).
- Si no hay entrada válida, se rechaza.

Conclusión

Aunque al inicio fue difícil entender la interacción entre FIRST, FOLLOW y la tabla LL(1), al implementar paso a paso y probar con ejemplos concretos logramos una comprensión práctica de cómo funciona un parser predictivo. Este conocimiento nos dio bases más sólidas sobre cómo se construyen compiladores y analizadores sintácticos reales.