Informe: Construcción de un Generador de Lexer

Nombre del Autor

2 de julio de 2025

Resumen

En este informe se describe el proceso sistemático para la construcción de un generador de lexer, detallando cada etapa fundamental y su justificación. El objetivo es proporcionar una guía clara y modular para el desarrollo eficiente de analizadores léxicos.

1. Introducción

El análisis léxico es la primera fase en la construcción de compiladores e intérpretes. Un lexer o analizador léxico identifica los componentes básicos (tokens) de un lenguaje de programación a partir de su representación textual. Este informe presenta una metodología estructurada para construir un generador de lexer, fundamentando cada etapa del proceso.

2. Proceso para Construir un Generador de Lexer

El proceso de construcción de un generador de lexer se puede dividir en varias etapas modulares, cada una con un propósito específico:

2.1. Definición de Tokens mediante Expresiones Regulares

El primer paso consiste en definir los tokens que el lenguaje reconocerá, como identificadores, números y operadores. Cada tipo de token se describe mediante una expresión regular (ER), lo que permite especificar patrones léxicos de forma concisa y precisa.

- **Ejemplo de tokens:**
  - `IDENT` = `[a-zA-Z_][a-zA-Z0-9_]*`
  - `NUM` = `[0-9]+`

2.2. Conversión de ER a AFND (Algoritmo de Thompson)

Cada expresión regular se transforma en un autómata finito no determinista (AFND) utilizando el algoritmo de Thompson. Este algoritmo proporciona reglas mecánicas para construir AFNDs a partir de operaciones básicas de ER (concatenación, alternancia, cierre, etc.), permitiendo transiciones ϵ y múltiples transiciones para el mismo símbolo.

2.3. Combinación de AFNDs

Todos los AFNDs individuales se combinan en un único AFND global. Esto se logra conectando cada AFND a un nuevo estado inicial común mediante transiciones ϵ, facilitando la gestión de múltiples patrones léxicos en un solo autómata.

2.4. Conversión de AFND a AFD (Algoritmo de Subconjuntos)

El AFND combinado se convierte en un autómata finito determinista (AFD) mediante el algoritmo de subconjuntos. En este proceso, cada estado del AFD representa un conjunto de estados del AFND, eliminando el no determinismo y las transiciones ϵ.

2.5. Minimización del AFD (Algoritmo de Hopcroft)

Para optimizar el autómata, se aplica el algoritmo de Hopcroft, que minimiza el número de estados del AFD fusionando aquellos que son equivalentes. Esto resulta en un autómata más eficiente y fácil de implementar.

2.6. Implementación del AFD

El AFD final se implementa mediante:

- Una tabla de transiciones (matriz estado $\times$ carácter $\rightarrow$ nuevo estado).
- Una tabla de estados de aceptación (asociando cada estado final con el token reconocido).

3. Conclusión

La construcción modular y argumentada de un generador de lexer permite obtener analizadores léxicos eficientes y mantenibles. Cada etapa del proceso contribuye a la robustez y claridad del sistema, facilitando su extensión y adaptación a nuevos lenguajes.