Manual Tecnico - DataForge

Universidad de San Carlos de Guatemala

Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Organización de Lenguajes y Compiladores 1
Primer Semestre 2024

Catedrático:

- Ing. Kevin Lajpop

Tutor académico (aux):

- Carlos Acabal

Manual Técnico del Proyecto de Análisis Léxico y Sintáctico

1. Introducción

El presente manual técnico tiene como propósito brindar una guía exhaustiva para la implementación de un sistema de análisis léxico y sintáctico, utilizando las herramientas JFlex y CUP en el lenguaje de programación Java. Este sistema está diseñado para llevar a cabo operaciones aritméticas y estadísticas, así como para generar distintos tipos de gráficos a partir de un conjunto de datos.

El análisis léxico y sintáctico son etapas fundamentales en el proceso de compilación de lenguajes de programación. El análisis léxico se encarga de la identificación y clasificación de los tokens o unidades básicas del lenguaje, mientras que el análisis sintáctico se ocupa de verificar la estructura gramatical de las secuencias de tokens para determinar su corrección según las reglas del lenguaje.

Este manual está dirigido a estudiantes, desarrolladores y cualquier persona interesada en la construcción de analizadores léxicos y sintácticos. A lo largo de este documento, se detallarán los pasos necesarios para configurar el entorno de desarrollo, definir la gramática del lenguaje, implementar el análisis léxico y sintáctico, y desarrollar funcionalidades adicionales como la generación de gráficos y la elaboración de reportes.

Se espera que al finalizar la lectura de este manual, el lector sea capaz de comprender y aplicar los conceptos y técnicas presentadas para la creación de un sistema de análisis léxico y sintáctico funcional, que pueda ser utilizado como base para proyectos más complejos en el ámbito de la teoría de lenguajes y compiladores.

2. Entorno de Desarrollo

Para llevar a cabo el desarrollo del proyecto de análisis léxico y sintáctico, es esencial configurar adecuadamente el entorno de desarrollo. A continuación, se detallan las herramientas y tecnologías necesarias, así como los pasos para su configuración:

Herramientas y Tecnologías:

- **Java:** Es el lenguaje de programación principal utilizado para el desarrollo del proyecto. Java es ampliamente reconocido por su portabilidad, robustez y amplia comunidad de desarrolladores.
- **JFlex:** Es una herramienta para la generación de analizadores léxicos en Java. JFlex facilita la definición de especificaciones léxicas a través de expresiones regulares y acciones que se ejecutan cuando se reconocen patrones específicos en el texto de entrada.
- **CUP:** Conocido como Constructor of Useful Parsers, es una herramienta para generar analizadores sintácticos en Java. CUP utiliza gramáticas especificadas en forma de Backus-Naur para definir la estructura gramatical del lenguaje a analizar.
- **JFreeChart:** Es una librería de código abierto en Java utilizada para la creación de una amplia variedad de gráficos, incluyendo gráficos de barras, de líneas, de torta, entre otros. Esta librería será utilizada para la generación de gráficos a partir de los datos procesados por el sistema.

Configuración:

1. **Instalación de Java JDK:**
    - Descargue e instale la última versión del Java Development Kit (JDK) desde el sitio web oficial de Oracle.
    - Configure la variable de entorno `JAVA_HOME` para apuntar al directorio de instalación del JDK.
    - Asegúrese de que el directorio `bin` del JDK esté incluido en la variable de entorno `PATH`.
2. **Descarga e Integración de JFlex y CUP:**
    - Descargue los archivos JAR de JFlex desde su sitio web oficial.
    - Descargue los archivos JAR de CUP desde su repositorio en SourceForge.
    - Agregue los archivos JAR de JFlex y CUP al classpath de su proyecto. En entornos de desarrollo integrados (IDE) como Eclipse o IntelliJ IDEA, puede hacerlo agregando los JAR como bibliotecas externas en la configuración del proyecto.
3. **Descarga e Integración de JFreeChart:**
    - Descargue el archivo JAR de JFreeChart desde su página oficial.
    - Agregue el archivo JAR de JFreeChart al classpath de su proyecto de la misma manera que se hizo con JFlex y CUP.

Una vez completada la configuración del entorno de desarrollo, estará listo para comenzar con la implementación del análisis léxico y sintáctico, así como con la generación de gráficos utilizando JFreeChart.

3. Análisis Léxico con JFlex

El análisis léxico es el primer paso en el proceso de compilación, donde el texto de entrada se divide en unidades significativas llamadas tokens. JFlex es una herramienta poderosa para generar analizadores léxicos en Java, facilitando la definición de reglas léxicas y patrones.

Creación del Archivo de Especificación de JFlex:

Un archivo de especificación de JFlex se compone de tres secciones principales:

4. **Definiciones de macros:** En esta sección, se definen expresiones regulares reutilizables, conocidas como macros, que simplifican la especificación de patrones complejos. Las macros se definen utilizando la sintaxis `NOMBRE = expresión_regular`.
5. **Reglas léxicas:** Esta sección contiene las reglas que asocian patrones (expresiones regulares) con acciones a ejecutar cuando se reconoce un patrón. Las acciones típicamente involucran la devolución de tokens al analizador sintáctico.
6. **Código de usuario:** Aquí se puede incluir código Java que se copiará directamente al archivo generado por JFlex. Este código puede contener definiciones de clases, métodos y variables adicionales que se utilizarán en el analizador léxico.

Ejemplo de un archivo de especificación de JFlex:

```
%class Lexer
%unicode
%cup

// Definiciones de macros
DIGIT = [0-9]
ID = [a-zA-Z][a-zA-Z0-9]*

%%

// Reglas léxicas
{DIGIT}+   { return symbol(sym.NUMBER, new Integer(yytext())); }
{ID}       { return symbol(sym.ID, yytext()); }

.          { // Código para manejar caracteres no reconocidos }
```

En este ejemplo, se define una clase `Lexer` que generará JFlex. La macro `DIGIT` representa un dígito, y la macro `ID` representa un identificador. Las reglas léxicas asocian secuencias de dígitos con el token `NUMBER` y secuencias de caracteres que conforman un identificador con el token `ID`.

Generación del Analizador Léxico:

Para generar el analizador léxico en Java a partir del archivo de especificación, siga estos pasos:

1. Abra una terminal o línea de comandos.
2. Navegue al directorio donde se encuentra el archivo de especificación de JFlex.
3. Ejecute el `nombre_archivo.flex` con el nombre de su archivo de especificación:

JFlex generará un archivo Java con el nombre especificado en la directiva `%class` del archivo de especificación. Este archivo contiene el código del analizador léxico, que puede integrarse con el analizador sintáctico generado por CUP en los siguientes pasos del proceso de compilación.

4. Análisis Sintáctico con CUP

El análisis sintáctico es el proceso de verificar la estructura gramatical de una secuencia de tokens generada por el analizador léxico. CUP (Constructor of Useful Parsers) es una herramienta que facilita la

generación de analizadores sintácticos en Java mediante la definición de gramáticas en forma de Backus-Naur (BNF).

Creación del Archivo de Especificación de CUP:

Un archivo de especificación de CUP se compone de varias secciones, incluyendo la definición de símbolos terminales y no terminales, las reglas gramaticales y las acciones semánticas asociadas a cada regla.

Ejemplo de un archivo de especificación de CUP:

```
parser code {:
  // Código de usuario para el analizador sintáctico
:}

terminal ID, NUMBER, PLUS;
non terminal expr;

expr ::= expr PLUS expr
         {: RESULT = new Plus($1, $3); :}
       | ID
         {: RESULT = new Identifier($1); :}
       | NUMBER
         {: RESULT = new Number($1); :};
```

En este ejemplo, se definen los símbolos terminales `ID`, `NUMBER` y `PLUS`, y el símbolo no terminal `expr`. Las reglas gramaticales especifican cómo se pueden combinar los símbolos para formar estructuras válidas en el lenguaje. Las acciones entre `{:` y `:}` se ejecutan cuando se reconoce una regla gramatical, y típicamente involucran la construcción de nodos en un árbol de sintaxis abstracta (AST).

Generación del Analizador Sintáctico:

Para generar el analizador sintáctico en Java a partir del archivo de especificación, siga estos pasos:

1. Abra una terminal o línea de comandos.
2. Navegue al directorio donde se encuentra el archivo de especificación de CUP.
3. Ejecute el `nombre_archivo.cup` con el nombre de su archivo de especificación:

Este generará dos archivos Java:

- `Parser.java`: Contiene el código del analizador sintáctico.
- `Sym.java`: Define las constantes para los símbolos terminales y no terminales utilizados en la gramática.

Puede integrar estos archivos con el analizador léxico generado por JFlex para completar el proceso de análisis léxico y sintáctico de su compilador o intérprete.

5. Autómatas y Gramáticas

En la teoría de la computación, los autómatas son estructuras abstractas que se utilizan para modelar sistemas computacionales. En el contexto de la compilación y el análisis de lenguajes, los autómatas desempeñan un papel fundamental en el análisis léxico y sintáctico:

- **Autómatas Finitos:** Se utilizan en el análisis léxico para reconocer patrones en el texto de entrada y generar los tokens correspondientes. Los autómatas finitos pueden ser deterministas (DFA) o no deterministas (NFA) y son capaces de reconocer lenguajes regulares.
- **Autómatas de Pila (Gramáticas):** Se emplean en el análisis sintáctico para verificar la estructura gramatical de las secuencias de tokens generadas por el analizador léxico. Los autómatas de pila pueden reconocer lenguajes libres de contexto, que son más expresivos que los lenguajes regulares y permiten describir la sintaxis de la mayoría de los lenguajes de programación.

La definición de la gramática del lenguaje en forma de Backus-Naur (BNF) o su variante extendida (EBNF) es esencial para el diseño del analizador sintáctico. La gramática define las reglas que rigen la estructura del lenguaje y se utiliza para generar el analizador sintáctico con herramientas como CUP.

6. Desarrollo del Lenguaje

El desarrollo del lenguaje implica la implementación de las estructuras de datos y las operaciones que forman parte de la especificación del lenguaje. En Java, esto se traduce en la creación de clases y métodos que representen las entidades del lenguaje y realicen las operaciones correspondientes, como operaciones aritméticas y funciones estadísticas.

7. Funciones de Graficación

La librería JFreeChart se utiliza para agregar capacidades de visualización de datos al sistema. Mediante la creación de métodos en Java, se pueden generar gráficos de barras, gráficos de pie, gráficos de líneas y histogramas basados en los datos procesados por el sistema. Estos gráficos pueden ser utilizados para representar los resultados de las operaciones estadísticas o para visualizar cualquier otro tipo de datos relevantes.

8. Reportes y Salida

La generación de reportes es un aspecto importante del sistema, ya que proporciona información útil sobre el proceso de análisis. Los reportes pueden incluir una tabla de tokens reconocidos por el analizador léxico, una tabla de errores encontrados durante el análisis y una tabla de símbolos que muestre las variables y constantes utilizadas en el código. Estos reportes se pueden generar en formato HTML para su fácil visualización en un navegador web.

Además, se debe implementar una consola de salida en la interfaz gráfica del sistema para mostrar los resultados de las operaciones realizadas y los mensajes generados durante el análisis. Esta consola puede ser una ventana de texto donde se muestren los resultados y mensajes en tiempo real.
