Pontificia Universidad JAVERIANA Bogotá	Pontificia Universidad Javeriana Departamento de Ingeniería de Sistemas Teoría de la Computación – 2023-30 Taller 1. Aplicación de lenguajes y gramáticas tipo 3 y 2 (trabajo en grupo) Profesor Jorge Valenzuela

1. Motivación

La motivación de este taller es aplicar de forma práctica la teoría de lenguajes formales, en particular los lenguajes tipo 3 (lenguajes regulares) y tipo 2 (lenguajes libres de contexto) con sus correspondientes autómatas finitos y de pila con el propósito de cubrir las etapas de análisis léxico, análisis sintáctico y análisis semántico para construir un lenguaje evaluador de expresiones matemáticas y lógicas desde la definición de las respectivas gramáticas formales y la alternativa de expresiones regulares para lenguajes tipo 3.

2. Lenguaje de expresiones matemáticas y lógicas

En el lenguaje que se pretende construir los programas están compuestos de bloques de sentencias que terminan con punto y coma ";". Maneja tipos de datos Booleano, Entero, Decimal y Cadena, los identificadores se deben nombrar comenzando con una letra (minúscula o mayúscula) y pueden estar seguidos de una combinación de números y letras (minúsculas o mayúsculas).

Las sentencias pueden ser de asignación, escritura, condición o ciclo, el lenguaje es débilmente tipado esto significa que no es necesario indicar el tipo a un identificador o variable, se infiere por el valor de asignación, lo anterior significa que la sintaxis de la sentencia de asignación es: <Identificador> <operadorAsignar> <expresión> <;>, el operador de asignar es el símbolo "=" y <expresión> es un valor de cualquier tipo de dato o una operación matemática o lógica, ejemplos de sentencias de asignación:

- n = 10;
- Saludo = "Hola";
- Prueba1 = (n > 0);

Para el primer ejemplo se infiere que el identificador "n" es de tipo **Entero**, Saludo es de tipo **Cadena** y Prueba1 de tipo **Booleano.**

La sintaxis de la sentencia de escritura es la siguiente:

"escribe" <expresión> <;>

Ejemplos:

- *Escribe Saludo + " mundo";*
- *Escribe (Saludo + " mundo");*

Las dos variantes de las sentencias anteriores dejan explicito que una expresión puede estar entre paréntesis.

La sentencia de condición es la típica de los lenguajes de programación, si la evaluación de una expresión devuelve verdadero se ejecuta un bloque sentencias, se puede opcionalmente evaluar la condición que evalúa a falso, pero en este lenguaje se va a manejar en español, ejemplo:

```
si ( numero % 2 == 0) {
  escribe ( numero + " es  par");
}
sino {
  escribe (numero + " es impar");
}
```

Las sentencias de condición se pueden anidar. La sentencia de ciclo también es la típica de los lenguajes de programación; mientras una expresión lógica evalúe a verdadero se ejecuta un bloque de sentencias; también aquí la palabra clave del ciclo es en español, “mientras”, ejemplo:

```
resultado = (numero > 0);
mientras resultado {
resto = numero / 2;
numero = numero – resto;
sí (residuo >= 1)
  resultado = false;
}
```

Una expresión matemática se compone de operandos y operaciones sobre estos. Las operaciones que se van a evaluar con nuestro lenguaje son binarias, esto es, requieren dos valores o variables. Las operaciones son: suma, resta, producto, división, módulo y potenciación, cada operación se va a representar como un operando a través de un símbolo en el lenguaje:

- Símbolo más (+) para sumar dos valores,
- Símbolo menos (-) para restar un valor de otro,
- Símbolo asterisco (*) para multiplicar dos valores,
- Símbolo diagonal (/) para dividir un valor entre otro.
- Símbolo porcentaje (%) para el módulo entre dos números.
- Símbolo de exponente (^) para elevar un numero a una potencia dada.

En la sintaxis de nuestro lenguaje las expresiones se escriben en notación infija, es decir, el operador que representa la operación se escribe entre dos operandos, que pueden ser valores numéricos constantes o identificadores de variables, ejemplos; “a + b” ó “7 * 8” o “2 ^ 3”. Las operaciones solo aplican a valores numéricos, salvo la suma que equivale, semánticamente a concatenar dos cadenas.

Una sentencia puede incluir una o más expresiones en una sola cadena que se leerá de izquierda a derecha y cuya prioridad en la evaluación será más alta para potenciación, luego para producto, división y módulo y luego para la suma y resta. Para variar ese orden o reafirmarlo se usarán paréntesis para asociar expresiones, los paréntesis, el de apertura ‘(‘ y el de cierre ‘)’, también son símbolos del lenguaje.

Las operaciones lógicas incluyen mayor que “>”, menor que “<”, mayor o igual “>=”, menor o igual “<=”, igual “==” y no igual “!=” y los conectores lógicos de conjunción “&&” y de disyunción “||”.

Se debe incluir el operador unario que le da el signo a un valor numérico.

3. descripción del taller

El objetivo del taller es definir las gramáticas formales (regular y libre de contexto) para el lenguaje descrito en el numeral 2, trasladar esa gramática a la notación BNF extendida que entiende ANTLR 4 y escribirla en un archivo con extensión .g4, luego, a partir de la definición de la gramática en el archivo .g4 se debe utilizar ANTLR 4 para generar automáticamente los programas en java con el analizador léxico y el analizador sintáctico.

Se debe utilizar el patrón “visitor” para recorrer el árbol sintáctico y codificar un programa java para evaluar las expresiones y así producir los resultados esperados de calcular el valor de la expresión o un mensaje de error en el caso que las sentencias o las expresiones no estuvieran correctamente escritas.

3.1 Herramienta para el taller

La base de construcción, prueba y desarrollo del taller es la herramienta ANTLR acrónimo de (ANother Tool for Language Recognition) – (Otra herramienta para el reconocimiento de lenguaje); la herramienta ANTLR (https://www.antlr.org/) es un conjunto de bibliotecas y herramientas en lenguaje Java para, leer, procesar, ejecutar o traducir desde gramáticas formales generativas (regulares o de libre contexto) escritas en texto estructurado y generar analizadores léxicos y sintácticos cuyos árboles se pueden recorrer para procesarlos.

Con base en las gramáticas definidas en una archivo de texto con extensión ‘.g4’, ANTLR genera lexers y parsers para generar tablas de tokens y árboles sintácticos que se pueden recorrer con el patrón visitor (o con el patrón listener pero no aplica para el taller)

La idea para el taller es usar ANTLR versión 4 en modo línea de comando y desde ahí implementar las 2 etapas requeridas (análisis léxico, análisis sintáctico) con la generación automática dada por ANTLR 4. Para el análisis semántico se debe codificar un programa en java que implemente el patrón “visitor” para recorrer el árbol y hacer la evaluación de las expresiones, mostrando el resultado en pantalla.

Si los estudiantes se sienten más cómodos utilizando un IDE, pueden hacerlo y deben usar el plug-in correspondiente, no obstante, lo único que deben entregar los estudiantes es el archivo .g4 con las dos gramáticas, el programa que implementa el patrón visitor evaluando semánticamente el lenguaje y un programa principal que lea los programas en el lenguaje creado y los interprete.

Se recomienda revisar el sitio https://riptutorial.com/antlr , donde se encuentra un tutorial sobre ANTLR 4 y documentación asociada

Pontificia Universidad JAVERIANA Bogotá	Pontificia Universidad Javeriana Departamento de Ingeniería de Sistemas Teoría de la Computación – 2023-30 Taller 1. Aplicación de lenguajes y gramáticas tipo 3 y 2 (trabajo en grupo) Profesor Jorge Valenzuela

3 evaluación

Las entregas se harán en la correspondiente actividad del campus Virtual, hasta la media noche del día indicado en la actividad publicada en la plataforma virtual y se sustentarán en la sesión del curso del día siguiente.

La evaluación del taller tendrá la siguiente escala:

- Excelente (5.0/5.0): Se desarrollan todos los puntos, se entrega la definición de las gramáticas formales que representa el lenguaje y se presentan el código generado en java junto con el código que implementa el patrón visitor, Al ejecutar el código con unos programas de prueba genera resultados correctos y esperados.
- Bueno (3.5/5.0): Se entrega la definición de las gramáticas formales que representa el lenguaje y se presentan el código generado en java junto con el código que implementa el patrón visitor, Al ejecutar el código con unos programas de prueba genera resultados correctos y esperados pero también resultados incorrectos.
- Aceptable (2.5/5.0): El código tiene problemas de compilación o no genera los resultados esperados y descritos en el numeral 2.
- Necesita mejoras sustanciales (1.0/5.0): El código tiene problemas de compilación o no genera resultados correctos en los puntos descritos en el numeral 2.
- No entregó (0.0/5.0): No entrego lo requerido o la entrega no sigue las instrucciones de este documento.