Nombres:

Emilio José

Apellidos:

De los Santos Sánchez

Matricula:

21-0843

Materia:

Teoria de autómatas y compiladores

Tema:

Actividad 15: Cuestionario en base a lecturas

Profesor

Rina Maria Familia

1.- ¿Por qué una Gramática proporciona las especificaciones sintácticas que se requieren para que un compilador trabaje eficientemente?

Un analizador sintáctico o "parser" toma como entrada la salida de un analizador léxico en la forma de streams de tokens. El parser compara el código fuente (stream de tokens) contra las reglas de producción de la gramática para detectar cualquier error en el código.

2.- ¿Por qué un Analizador Sintáctico puede ser construido automáticamente a partir de determinadas clases de gramática?

Las gramáticas libres de contexto nos proporcionarán un modo para diseñar lenguajes de programación ofreciendo las siguientes ventajas:

1. Una gramática es una especificación sintáctica precisa y fácil de entender de un lenguaje de programación.
2. A partir de algunas clases de gramáticas se puede construir automáticamente un analizador sintáctico que determine si un programa fuente está bien construido.
3. Los cambios a la gramática son más fáciles de realizar cuando se tiene una descripción gramatical del lenguaje.

3.- ¿Por qué una gramática diseñada adecuadamente que impone una estructura a un lenguaje de programación es útil para la traducción de programas fuente a código objeto correcto para un lenguaje de programación y para detectar errores?

Si un compilador tuviera que procesar sólo programas correctos, su diseño e implantación se simplificará mucho. Pero los programadores a menudo escriben programas incorrectos, y un buen compilador debería ayudar al programador a identificar y localizar errores.

4.- Establezca tres relaciones funcionales entre el Análisis Léxico y el Análisis Sintáctico.

1. Se mejora la portabilidad del compilador. Las peculiaridades del alfabeto de entrada y otras anomalías propias de los dispositivos pueden limitarse al analizador léxico. La representación de símbolos especiales o no estándares, como en Pascal, pueden ser aisladas en el analizador léxico.
2. Otra razón por la que se separan los dos análisis es para que el analizador léxico se centre en el reconocimiento de componentes básicos complejos. Por ejemplo, en FORTRAN, existen el siguiente par de proposiciones: DO 5 I = 2.5 (Asignación de 2.5 a la variable DO5I) DO 5 I = 2,5 (Bucle que se repite para I = 2, 3, 4, 5).
3. Se mejora la eficiencia del compilador. Un analizador léxico independiente permite construir un procesador especializado y potencialmente más eficiente para esa función. Gran parte del tiempo se consume en leer el programa fuente y dividirlo en componentes léxicos. Con técnicas especializadas de manejo de buffers para la lectura de caracteres de entrada y procesamiento de componentes léxicos se puede mejorar significativamente el rendimiento de un compilador.

5.- Describa brevemente los métodos generales de Análisis Sintáctico:

- **Métodos Universales (Algoritmo Cocke-Younger-Kasami, Algoritmo Early)**

  Pueden analizar cualquier gramática. Estos métodos, sin embargo, son demasiado ineficientes para usarlos en la producción de compiladores.

- **Métodos Ascendentes**

  Los analizadores sintácticos ascendentes comienzan en las hojas y suben hacia la raíz. En ambos casos, se examina la entrada al analizador sintáctico de izquierda a derecha, un símbolo a la vez.

- **Métodos Descendentes**

  Los analizadores sintácticos descendentes construyen árboles de análisis sintáctico desde arriba (la raíz) hasta abajo (las hojas).

6- ¿Cómo manejan los errores sintácticos?

La precisión de los métodos modernos de análisis sintácticos, que pueden detectar la presencia de errores dentro de los programas de una forma muy eficiente. La detección exacta de la presencia de errores semánticos y lógicos en el momento de la compilación es mucho más difícil.

7.- ¿Establezca diferencias entre los errores léxicos, los sintácticos, los semánticos y los lógicos?

- **Léxicos:** como escribir mal un identificador. palabra clave u operador.
- **Sintácticos:** como una expresión aritmética con paréntesis no equilibrados.
- **Semánticos:** como un operador aplicado a un operando incompatible.
- **Lógicos:** como una llamada infinitamente recursiva.

8.- Describa tres razones de por qué se debe separar el análisis léxico del sintáctico.

1. Se mejora la eficiencia del compilador. Un analizador léxico independiente permite construir un procesador especializado y potencialmente más eficiente para esa función.
2. Un diseño sencillo es quizás la consideración más importante. Separar el análisis léxico del análisis sintáctico a menudo permite simplificar una u otra de dichas fases.
3. Se mejora la portabilidad del compilador. Las peculiaridades del alfabeto de entrada y otras anomalías propias de los dispositivos pueden limitarse al analizador léxico.

9.- Mencione cuatro objetivos del manejador de errores en un Analizador Sintáctico.

1. Debe informar de la presencia de errores con claridad y exactitud.
2. Se debe recuperar de cada error con la suficiente rapidez como para detectar errores posteriores.
3. No debe retrasar de manera significativa el procesamiento de programas correctos.
4. En los casos difíciles, el manejador de errores quizá tenga que adivinar qué tenía en mente el programador cuando escribió el programa.

10.- ¿Cómo debe informar un manejador de errores de la presencia de la ocurrencia de un error?

Al menos debe informar del lugar en el programa fuente donde se detecta el error. Porque es muy probable que el error real se haya producido en alguno de los componentes léxicos anteriores. Una estrategia común empleada por muchos compiladores es imprimir la línea errónea con un apuntador a la posición donde se detecta el error. Si hay una posibilidad razonable de saber cuál es realmente el error. También se incluye un mensaje de diagnóstico informativo y comprensible: por ejemplo. "falta punto y coma en esta posición".

11.- Describa varias estrategias generales que puede emplear un analizador sintáctico para recuperarse de un error sintáctico.

Recuperación en modo de pánico: Este es el método más sencillo de implantar y pueden utilizarlo la mayoría de los métodos de análisis sintáctico. Al descubrir un error, el analizador sintáctico desecha símbolos de entrada, de uno en uno, hasta que encuentra uno perteneciente a un conjunto designado de componentes léxicos de sincronización.

Recuperación a nivel de frase: Al descubrir un error, el analizador sintáctico puede realizar una corrección local de la entrada restante; es decir, puede sustituir un prefijo de la entrada restante por alguna cadena que permita continuar al analizador sintáctico.

Producciones de error: Si se tiene una buena idea de los errores comunes que pueden encontrarse, se puede aumentar la gramática del lenguaje con producciones que generen las construcciones erróneas. Entonces se usa esta gramática aumentada con las producciones de error para construir el analizador sintáctico.

Corrección global: Idealmente, sería deseable que un compilador hiciera el mínimo de cambios posibles al procesar una cadena de entrada incorrecta. Existen algoritmos para elegir una secuencia mínima de cambios para obtener una corrección global de menor costo. Dada una cadena de entrada incorrecta x y la gramática G. estos algoritmos encontrarán un árbol de análisis sintáctico para una cadena relacionada y, tal que el número de inserciones, supresiones y modificaciones de componentes léxicos necesario para transformar x en y sea el mínimo posible.

12.- De un ejemplo de uso de la estrategia conocida como "modo de pánico".

Son generalmente delimitadores, como el punto y coma o la palabra clave end, cuyo papel en el programa fuente está claro.

13.- De un ejemplo de uso de la estrategia conocida como "a nivel de frase".

Sustituir una coma por un punto y coma, suprimir un punto y coma sobrante, o insertar un punto y coma que falta.

14.- De un ejemplo de uso de la estrategia conocida como "de producción de error".

Si el analizador sintáctico usa una producción de error, se pueden generar diagnósticos de error apropiados para indicar la construcción errónea reconocida en la entrada.

15.- De un ejemplo de uso de la estrategia conocida como "de corrección global".

La noción de corrección de costo mínimo proporciona una escala para evaluar las técnicas de recuperación de errores, y se ha usado para encontrar cadenas de sustitución óptimas para la recuperación a nivel de frase.

16.- ¿Qué es un árbol de análisis sintáctico? Proporcione ejemplos.

Un árbol de parseo es la representación gráfica de una derivación. Es conveniente ver cómo las cadenas son derivadas a partir del símbolo de inicio. El símbolo de inicio de la derivación se convierte en la raíz del árbol de parseo.

Ejemplo:

Tomaremos la derivación más a la izquierda de: **a + b * c**

E -> E * E

E -> E + E * E

E -> id + E * E

E -> id + id * E

E -> id + id * id

17.- ¿Cómo se trata la ambigüedad en un árbol sintáctico?

Ningún método puede detectar y remover la ambigüedad automáticamente, pero puede ser removida re-escribiendo toda la gramática sin ambigüedades, o estableciendo y siguiendo las siguientes restricciones de precedencia y asociatividad.

18.- ¿Qué es el análisis sintáctico descendente?

Los analizadores sintácticos descendentes son lo que construyen el árbol sintáctico de la sentencia a reconocer de una forma descendente, comenzando por el símbolo inicial o raíz, hasta llegar a los símbolos terminales que forman la sentencia. El análisis descendente es un procedimiento que crea objetivos y subobjetivos al intentar relacionar una sentencia con su entorno sintáctico.

19.- ¿Qué es el análisis sintáctico descendente recursivo?

Se puede considerar el análisis sintáctico descendente como un intento de encontrar una derivación por la izquierda para una cadena de entrada También se puede considerar como un intento de construir un árbol de análisis sintáctico para la entrada comenzando desde la raíz y creando los nodos del árbol en orden previo.

20.- ¿Qué es un análisis sintáctico predictivo?

En muchos casos, escribiendo con cuidado una gramática, eliminando su recursión por la izquierda y factorizando por La izquierda la gramática resultante, se puede obtener una gramática analizable con un analizador sintáctico por descenso recursivo que no necesite retroceso, es decir, un analizador sintáctico predictivo.

21.- ¿Cómo se crea un diagrama de transición para un análisis sintáctico predictivo?

Todo lenguaje de programación puede especificarse de una manera completa y rigurosa mediante un metalenguaje, tal como BNF o diagramas sintácticos. Las reglas para la construcción de diagramas sintácticos a partir de las producciones de una gramática son: secuencial, alternativa y repetitiva.

22.- ¿Qué es el análisis sintáctico predictivo no recursivo?

Un analizador sintáctico predictivo guiado por tablas tiene un buffer de entrada, una pila, una tabla de análisis sintáctico y una cadena de salida. El buffer de entrada contiene la cadena que se va a analizar, seguida de un símbolo utilizado como delimitador derecho para indicar el fin de la cadena de entrada. La pila contiene una secuencia de símbolos gramaticales con un símbolo en la parte de abajo, que indica la base de la pila. Al principio, la pila contiene el símbolo inicial de la gramática encima del símbolo delimitador.

23.- ¿Cómo se construye una tabla de análisis sintáctico?

Existen tres técnicas para construir una tabla de análisis sintáctico LR para una gramática. El primer método, llamado LR sencillo (SLR, en inglés) es el más fácil de implantar, pero el menos poderoso de los tres. Puede que no consiga producir una tabla de análisis sintáctico para algunas gramáticas que otros métodos sí consiguen. El segundo método, llamado LR canónico, es el más poderoso y costoso. El tercer método, llamado LR con examen por anticipado (LALR, en inglés), está entre los otros dos en cuanto a poder y costo. El método LALR funciona con las gramáticas de la mayoría de los lenguajes de programación y, con un poco de esfuerzo, se puede implantar en forma eficiente.

24.- ¿Cómo se realiza la recuperación de errores en el análisis sintáctico predictivo?

Cuando un compilador analiza un programa fuente, no debe finalizar su ejecución cada vez que se encuentra un error. Lo ideal sería que informará de todos los errores contenidos en él en una sola compilación. Normalmente cuando encuentra un error, yyparse llama a la rutina yyerror y finaliza su ejecución. Pero existen algunos mecanismos que pueden

hacer que el analizador se recupere de los errores y sea capaz de seguir adelante con el proceso.

25.- Detalle cómo se comporta el análisis sintáctico ascendente.

Análisis sintáctico ascendente, conocido como análisis sintáctico por desplazamiento y reducción. El análisis sintáctico por desplazamiento y reducción intenta construir un árbol de análisis sintáctico para una cadena de entrada que comienza por las hojas (el fondo) y avanza hacia la raíz (la cima). Se puede considerar este proceso como el de "reducir" una cadena w al símbolo inicial de la gramática.

26.- Cómo se hace la implantación por medio de pila del análisis sintáctico por desplazamiento y reducción?

Ejemplo Considérese la gramática

S-> aABe

A-> Abc | b

B->d

La frase abbcde se puede reducir a S por los siguientes pasos:

abbcde

aAbcde

aAde

aABe

S

Se examina abbcde buscando una subcadena que concuerde con el lado derecho de alguna producción. Las subcadenas b y d sirven. Elijase la b situada más a la izquierda y sustitúyase por A, el lado izquierdo de la producción A-> b; así se obtiene la cadena aAbcde. A continuación, las subcadenas Abc, b y d concuerdan con el lado derecho de alguna producción. Aunque b es la subcadena situada más a la izquierda que concuerda con el lado derecho de una producción, se elige sustituir la subcadena Abc por A, que es el lado derecho de la producción A->Abc. Se obtiene ahora aAde. Sustituyendo después d por B, que es el lado izquierdo de la producción B-> d, se obtiene aABe. Ahora se puede sustituir toda esta cadena por S. Por tanto, mediante una secuencia de cuatro reducciones se puede reducir abbcde a S. De hecho, estas reducciones trazan la siguiente derivación por la derecha en orden inverso:

S =>md aABe =>md aAde => aAbcde =>md abbcde

27.- ¿Qué es el análisis sintáctico por precedencia de operadores?

Si dos operadores diferentes comparten un mismo operando, la precedencia de los operadores decide cual tomará el operando. Esto es, 2+3*4 puede tener 2 árboles de parseo, uno que corresponde a (2+3)*4 y otro que corresponda a 2+(3*4). Estableciendo la precedencia entre operadores, este problema puede ser resuelto con facilidad. Como en este ejemplo, matemáticamente la multiplicación tiene precedencia sobre la suma, entonces la expresión 2+3*4 siempre será interpretada de la misma manera.

28.- ¿Cómo se hace la recuperación de errores en el análisis sintáctico anterior?

Hay dos maneras habituales de determinar qué relaciones de precedencia deben cumplirse entre un par de terminales. El primer método que se estudia es intuitivo y se basa en las nociones tradicionales de asociatividad y precedencia dé operadores. Por ejemplo, si * tiene mayor precedencia que +, se hace + <* * y **> +. El segundo método para seleccionar las relaciones de precedencia de operadores consiste en construir primero una gramática no ambigua para el lenguaje, una gramática que refleje la asociatividad y precedencia correctas en sus árboles de análisis sintáctico.

29.- ¿Cómo funcionan los generadores de análisis sintáctico?

YACC (YET ANOTHER COMPILER COMPILER): provee una herramienta general para describir la entrada de un programa de computación. El usuario de YACC especifica las estructuras de su entrada, junto con el código que será invocado en la medida en que cada una de esas estructuras es reconocida.

30.- Describa cómo funciona YACC. ¿Cómo se hace la recuperación de errores en YACC?

- **YACC (YET ANOTHER COMPILER COMPILER):** provee una herramienta general para describir la entrada de un programa de computación. El usuario de YACC especifica las estructuras de su entrada, junto con el código que será invocado en la medida en que cada una de esas estructuras es reconocida.
- **Yacc genera una función llamada yyparse**() que contiene el analizador sintáctico. Esta función se comporta como una máquina de estados cuya misión es intentar reducir todo el fichero de entrada al símbolo inicial de la gramática (el primero que se haya definido). Si yacc lo consigue, el analizador sintáctico volverá sin error, y en caso contrario, se invocará a la función y yerror(), que debe estar definida también en algún sitio.