UNIVERSIDAD AUTÓNOMA DE CHIAPAS

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN
CAMPUS 1

Licenciatura en Ingeniería en Desarrollo y Tecnologías de Software

Actividad. 2.1 Realizar la siguiente investigación de analizador sintáctico en formato APA

6 “M”

Materia: Compiladores

Docente: Luis Gutiérrez Alfaro
ALUMNO:
A 211387
Steven de Dios Montoya Hernández
GIT:

TUXTLA GUTIÉRREZ, CHIAPAS
Viernes, 8 De Septiembre de 2023, 23:59

Reconocer el concepto de recursividad.

Es un método fácil de implementar.

Genera una función por cada símbolo no terminal.

Si las producciones son recursivas, entonces la función correspondiente es recursiva.

Es llamado análisis descendente recursivo.

Sólo funciona con gramáticas LL(1)

Construir un analizador descendente recursivo para la siguiente gramática:
Input → Input Expr ‘\n’ | Expr ‘\n’
Expr → Expr ‘+’ Term | Expr ‘-’ Term | Term
Term → Term ‘**’** S | Term ‘/’ S | S
S → Factor ‘^’ S | FactorFactor → (Expr) | CHAR

Solución: La gramática anterior no es adecuada debido a que tiene recursión izquierda.
Por lo tanto, eliminar recursión izquierda:
Input → Expr ‘\n‘ {Expr ‘\n‘}
Expr → Term {(‘+’|’-‘) Term}
Term → S {(‘*’|’/’) S}
S → Factor ‘^’ S | FactorFactor → ‘(‘ Expr ‘)’ | CHAR

Crear una función por cada símbolo no terminal.

Cada función debe llamar a las funciones de los símbolos que componen cada regla en el orden en que aparecen.

La función retorna cuando logra verificar por completo la regla.

Los lenguajes de programación son naturalmente recursivos.

Notación de gramática que es ideal para el diseño de analizadores sintácticos descendentes recursivos: EBNF

El análisis sintáctico descendente recursivo es uno de los más fáciles de implementar

Operador que permite definir secuencias sin recursividad a la izquierda: Repetición

El analizador descendente recursivo no funciona para todas las gramáticas

Cómo se representa el operador de opción: Con corchetes []

Los analizadores descendentes recursivos son más lentos que otros algoritmos

Cómo se representa el operador agrupamiento: Con paréntesis ()

El analizador descendente recursivo requiere que la gramática sea modificada para eliminar la recursividad izquierda.

Existen dos tipos principales de recursividad en los analizadores sintácticos:

Recursividad izquierda: En este caso, una regla puede hacer referencia a sí misma antes de hacer referencia a otras reglas. Por ejemplo, en una gramática que define expresiones aritméticas, una expresión podría estar definida en términos de sí misma y de operadores.

Recursividad derecha: Aquí, una regla se refiere a sí misma después de hacer referencia a otras reglas. Siguiendo con el ejemplo de expresiones aritméticas, una expresión podría estar definida en términos de operadores y números antes de hacer referencia a sí misma.

Es importante tener en cuenta que la recursividad en los analizadores sintácticos debe ser manejada cuidadosamente para evitar bucles infinitos durante el proceso de análisis. Los analizadores sintácticos utilizan técnicas como la recursividad descendente y la recursividad ascendente para gestionar la recursividad de manera eficiente y correcta.

Definir los conceptos de árbol de análisis sintáctico, derivación, inferencia, inferencia recursiva y de un ejemplo.

Árbol de Análisis Sintáctico:

Un árbol de análisis sintáctico, también conocido como árbol de análisis gramatical o árbol de derivación, es una estructura jerárquica que representa cómo una cadena de símbolos en un lenguaje es analizada y descompuesta en sus componentes léxicos y gramaticales según las reglas de una gramática formal. Cada nodo del árbol representa un símbolo gramatical, y las aristas representan las derivaciones entre estos símbolos, mostrando cómo las reglas gramaticales se aplican para construir la cadena original.

Los árboles de análisis sintáctico son utilizados en el proceso de análisis sintáctico para representar la estructura jerárquica de una cadena de símbolos en un lenguaje.

Cada nodo del árbol representa un símbolo gramatical, ya sea un terminal (como una palabra clave o un identificador) o un no terminal (un símbolo que se expande en más símbolos según las reglas gramaticales).

Las aristas del árbol representan las aplicaciones de reglas gramaticales y cómo los símbolos se derivan entre sí.

El nodo raíz del árbol representa el símbolo inicial de la gramática, y las hojas del árbol representan la cadena de entrada que se está analizando.

Derivación:

En el contexto de la teoría de lenguajes formales y la gramática, la derivación se refiere al proceso de aplicar las reglas de una gramática a una cadena de símbolos para construir una estructura válida de acuerdo con esa gramática. Es un paso a paso que muestra cómo se forman las cadenas dentro del lenguaje definido por la gramática.

La derivación describe cómo una cadena de símbolos se genera siguiendo las reglas gramaticales de una gramática formal.

Comienza con un símbolo inicial y aplica sucesivamente las reglas de la gramática para reemplazar símbolos no terminales por sus expansiones.

Cada paso en la derivación representa una aplicación de una regla gramatical, y la secuencia completa de pasos conduce a la formación de la cadena de símbolos final.

La derivación describe cómo una cadena de símbolos se genera siguiendo las reglas gramaticales de una gramática formal.

Comienza con un símbolo inicial y aplica sucesivamente las reglas de la gramática para reemplazar símbolos no terminales por sus expansiones.

Cada paso en la derivación representa una aplicación de una regla gramatical, y la secuencia completa de pasos conduce a la formación de la cadena de símbolos final.

Inferencia:

La inferencia es un proceso de deducción o razonamiento a partir de premisas o información disponible. En el contexto del análisis sintáctico, la inferencia se refiere a la capacidad de determinar cómo se construye una cadena de símbolos en función de las reglas de una gramática dada.

La inferencia implica llegar a una conclusión a partir de premisas o información disponible.

En el contexto del análisis sintáctico, la inferencia implica la capacidad de deducir cómo se construye una cadena de símbolos en función de las reglas gramaticales.

Inferencia Recursiva:

La inferencia recursiva es un concepto que implica la aplicación repetida de reglas o pasos de inferencia a medida que se avanza en el análisis. En el contexto del análisis sintáctico, la inferencia recursiva ocurre cuando una regla gramatical se aplica de manera repetitiva para construir una estructura más compleja, y esta regla puede hacer referencia a sí misma, lo que resulta en la creación de una estructura jerárquica.

La inferencia recursiva ocurre cuando una regla gramatical se aplica repetidamente a sí misma o a otras reglas para construir estructuras más complejas.

En el análisis sintáctico, las reglas recursivas pueden ser útiles para describir estructuras anidadas, como las expresiones aritméticas o las listas en un lenguaje de programación.

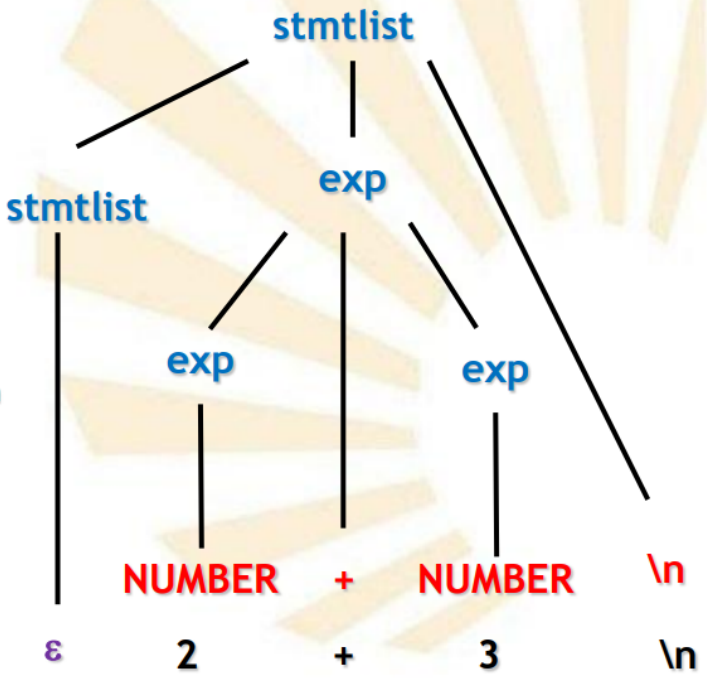
Explicar el proceso de construcción de árboles de análisis sintáctico y de un ejemplo.

Ejemplo 1: análisis de 2 + 3

Derivación
por la derecha
en orden inverso

=> stmtlist
=> stmtlist exp \n
=> stmtlist exp + exp \n
=> stmtlist exp + NUMBER \n
=> stmtlist NUMBER + NUMBER \n
=> ε NUMBER + NUMBER \n
= NUMBER + NUMBER \n

Árbol sintáctico



Tokenización: El proceso comienza con la tokenización, donde la cadena de entrada se divide en unidades más pequeñas llamadas tokens, como palabras clave, identificadores, operadores y símbolos.

Análisis Sintáctico: El analizador sintáctico procesa los tokens de acuerdo con las reglas gramaticales de la lengua para construir un árbol de análisis sintáctico. Esto implica identificar la estructura jerárquica de la cadena de entrada.

Construcción del Árbol: El analizador comienza con el símbolo inicial de la gramática y aplica las reglas gramaticales para derivar gradualmente los símbolos no terminales en la cadena de entrada. Cada regla aplicada se representa como una arista en el árbol.

Recursión: Si las reglas gramaticales son recursivas, el analizador puede aplicar repetidamente una regla para descomponer una estructura en subestructuras más pequeñas. Esto se refleja en la estructura jerárquica del árbol.

Árbol Resultante: Una vez que todas las reglas se han aplicado y todos los símbolos se han derivado, se obtiene un árbol de análisis sintáctico que representa la estructura de la cadena de entrada de acuerdo con las reglas gramaticales.

Ejemplo: Árbol de Análisis Sintáctico para Expresión Aritmética:

Consideremos la siguiente gramática simplificada para expresiones aritméticas:

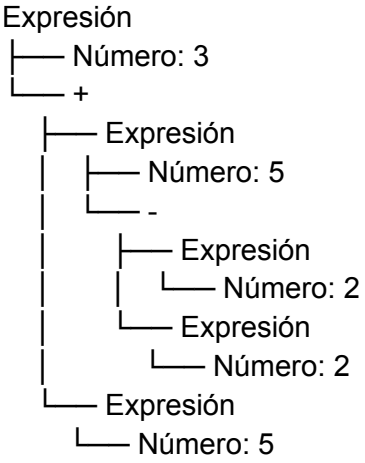
Expresion → Expresion + Expresion
| Expresion - Expresion
| Numero

Y la cadena de entrada: "3 + 5 - 2"

El proceso de construcción del árbol de análisis sería:

Tokenización: Los tokens serían "3", "+", "5", "-", "2".

Construcción del Árbol:



Explicar el proceso de construcción de árboles sintácticos de acuerdo a inferencias y de un ejemplo

Proceso de Construcción de Árboles Sintácticos mediante Inferencias:

Inicio: Se comienza con el símbolo inicial de la gramática.

Aplicación de Reglas: Se aplican las reglas gramaticales a medida que se recorre la cadena de entrada. Cada regla puede generar uno o más símbolos y posiblemente llevar a la expansión recursiva.

Deducir Jerarquía: A medida que se aplican las reglas, se deduce la estructura jerárquica. Los nodos del árbol representan los símbolos generados y las aristas representan las aplicaciones de reglas.

Terminación: El proceso continúa hasta que todos los símbolos de la cadena de entrada se han generado y se ha construido el árbol sintáctico completo.

Concepto de Inferencia en la Construcción de Árboles Sintácticos:

La inferencia en la construcción de árboles sintácticos implica el proceso de deducir la estructura jerárquica de una cadena de entrada utilizando las reglas gramaticales de un lenguaje. A medida que se aplican las reglas, se infiere cómo los símbolos se relacionan entre sí para formar la estructura sintáctica.

Pasos en la Construcción del Árbol Mediante Inferencias:

Aplicación de Reglas Gramaticales: Se comienza con el símbolo inicial de la gramática y se aplican las reglas gramaticales para expandir los símbolos en la cadena de entrada. Cada regla representa una inferencia de cómo se relacionan los símbolos.

Inferencia Recursiva: Si la gramática tiene reglas recursivas, se aplican de manera repetitiva para construir estructuras anidadas. Esta inferencia recursiva es crucial para manejar expresiones complejas y jerarquías en el lenguaje.

Deducción de la Estructura: A medida que se aplican las reglas, se deduce la estructura jerárquica del árbol. Los nodos del árbol representan los símbolos generados, y las aristas representan las inferencias realizadas mediante las reglas.

Construcción del Árbol: La deducción acumulada a lo largo de la aplicación de las reglas y las inferencias resulta en la construcción del árbol sintáctico completo, que representa la estructura de la cadena de entrada.

Importancia de la Inferencia en el Análisis Sintáctico:

La inferencia es esencial en el análisis sintáctico ya que permite transformar una cadena de símbolos en una representación estructurada y comprensible. Los árboles sintácticos generados mediante inferencias son herramientas valiosas para comprender cómo las cadenas de símbolos en un lenguaje se organizan y se ajustan a las reglas de la gramática.

Beneficios de los Árboles Sintácticos Mediante Inferencias:

Ayudan a visualizar la estructura de una cadena de entrada y su relación con las reglas gramaticales.
Facilitan la detección de errores sintácticos al resaltar discrepancias entre la estructura real y la esperada.
Sirven como base para la generación de código y el análisis semántico en etapas posteriores del procesamiento del lenguaje.

Ejemplo: Árbol Sintáctico para una Expresión Aritmética:

Gramática simplificada:

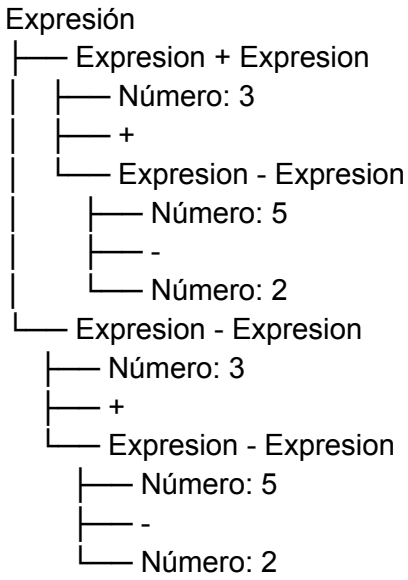
Expresion → Numero
| Expresion + Expresion
| Expresion - Expresion

Cadena de entrada: "3 + 5 - 2"

Inicio: Se comienza con el símbolo Expresion.

Aplicación de Reglas:

Expresión se expande en Expresion + Expresion.
La primera Expresión se expande en Número con el valor 3.
La segunda Expresión se expande en Expresion - Expresion.
La primera Expresión en la segunda regla se expande en Número con el valor 5.
La segunda Expresión en la segunda regla se expande en Número con el valor 2.
Deducir Jerarquía:



Explicar el proceso de construcción de derivaciones de acuerdo a árboles y de un ejemplo

¿Que es un Árbol de Derivación?

Un árbol es un conjunto de puntos, llamados nodos, unidos por líneas llamadas aristas o arcos. Una arista conecta dos nodos distintos. Este árbol muestra claramente cómo se agrupan los símbolos de una cadena terminal en subcadenas, que pertenecen al lenguaje de una de las variables de la gramática.Pero lo más importante es que el árbol, conocido como "árbol de derivación",cuando se emplea en un compilador, es la estructura de datos que representa el programa fuente. En un compilador, la estructura del árbol del programa fuente facilita la traducción del programa fuente a código ejecutable permitiendo que el proceso de traducción sea realizado por funciones naturales recursivas. El árbol de derivación permite mostrar gráficamente como se puede derivar cualquier cadena de un lenguaje a partir del símbolo distinguido de una gramática que genera ese lenguaje. Ciertas gramáticas permiten que una cadena terminal tenga más de un árbol de análisis. Esta situación hace que esa gramática sea inadecuada para un lenguaje de programación, ya que el compilador no puede decidir la estructura sintáctica de determinados programas fuentes y, por tanto, no podría deducir con seguridad cuál será el código ejecutable apropiado correspondiente al programa

Terminología de árboles

Los árboles son colecciones de nodos, que mantienen una relación padre-hijo. Un nodo tiene como máximo un padre, que se dibuja por encima del mismo, y cero o más hijos, que se dibuja por debajo.

Existe un nodo, el nodo raíz, que no tiene padre; es nodo es el que posee un grado de entra igual a cero y su grado de salida puede ir de 1 a más.Los nodos sin hijos se denominan hojas. Los nodos que no tienen hojas son nodos interiores.

El hijo de un hijo… de un nodo es un descendiente de dicho nodo. Un padre de un padre de un… es un ancestro. Evidentemente, cualquier nodo es ancestro y descendiente de sí mismo.

Los hijos de un nodo se ordenan de "Izquierda a derecha" y se dibujan así.Si el nodo N está al a izquierda del nodo M, entonces todos los descendientes de N son los que están a la izquierda de todos los descendientes de M

Proceso de Construcción de Derivaciones a partir de Árboles Sintácticos:

La construcción de derivaciones a partir de árboles sintácticos implica reconstruir la cadena de entrada original siguiendo las reglas gramaticales que se aplicaron en la creación del árbol. Aquí está el proceso junto con un ejemplo:

Inicio: Comienza con el nodo raíz del árbol sintáctico.

Recorrido del Árbol: Realiza un recorrido del árbol en profundidad (DFS) siguiendo las aristas, es decir, las aplicaciones de reglas gramaticales.

Construcción de la Derivación: A medida que avanzas por el árbol, anota los símbolos generados y las reglas aplicadas en el orden en que aparecen.

Terminación: Al llegar a las hojas del árbol (nodos que representan símbolos terminales), habrás construido la derivación completa de la cadena original.

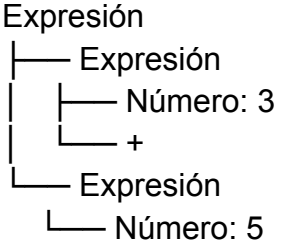
Ejemplo: Construcción de Derivación a partir de Árbol Sintáctico:

Supongamos la siguiente gramática y árbol sintáctico:

Gramática:

Expresion → Numero
| Expresion + Expresion

Árbol Sintáctico:



Proceso:

1. Comienzas en el nodo raíz Expresión.
2. Avanzas por la rama izquierda y llegas a Expresión.
3. Sigues hacia abajo y llegas a Número: 3.
4. Vuelves al nodo padre Expresión y avanzas a través de +.
5. Te desplazas a la rama derecha y llegas a Expresión.
6. Llegas a Número: 5.

Derivación Construida:
Expresion → Expresion + Expresion → Numero + Expresión → 3 + Expresión → 3 + Número → 3 + 5

En este ejemplo, siguiendo el proceso de construcción de derivaciones a partir del árbol sintáctico, hemos reconstruido la derivación original que generó la cadena "3 + 5".

La construcción de derivaciones a partir de árboles sintácticos es fundamental para comprender cómo las reglas gramaticales se aplican para generar una cadena de entrada específica y cómo se descompone en subexpresiones según la estructura jerárquica.

Explicar el proceso de construcción de inferencias recursivas de acuerdo a derivaciones y de un ejemplo

Proceso de Construcción de Inferencias Recursivas a partir de Derivaciones:

La construcción de inferencias recursivas a partir de derivaciones implica aplicar reglas gramaticales que contienen recursividad, lo que resulta en la generación repetida de una misma regla en la derivación. Aquí está el proceso junto con un ejemplo:

Inicio: Comienzas con una regla gramatical que contiene recursividad.

Aplicación de Reglas: Aplicas la regla a sí misma o a otra regla, generando una estructura más compleja.

Recursión: El proceso de aplicar la regla de manera repetida crea una serie de expansiones anidadas, cada una basada en la misma regla.

Construcción de Derivación: A medida que avanzas en la derivación, anotas las expansiones recursivas y cómo se relacionan con la regla original.

Terminación: Al llegar a expansiones que no pueden seguir aplicándose debido a la terminación recursiva, habrás construido la derivación completa.

Ejemplo: Construcción de Inferencias Recursivas a partir de Derivaciones:

Supongamos la siguiente gramática y regla gramatical recursiva:

Gramática:

Expresion → Numero
| Expresion + Expresion

Regla Gramatical Recursiva:

Expresion → Expresion + Numero

Derivación Recursiva:

Comienzas con la regla Expresion → Expresion + Numero.

Aplicas Expresión en el lado izquierdo de la regla: Expresion → Expresion + Numero + Numero.

Continúan aplicando Expresión en el lado izquierdo: Expresion → Expresion + Numero + Numero + Numero.

Derivación Construida:

Expresion → Expresion + Número → Expresion + Numero + Numero → Expresion + Numero + Numero + Numero

En este ejemplo, la regla Expresión se aplica recursivamente en la derivación, lo que resulta en una cadena de expresiones anidadas. Cada vez que se aplica la regla, se añade un nuevo Número, creando una estructura similar a la suma repetida.

La construcción de inferencias recursivas a partir de derivaciones es esencial para comprender cómo las reglas gramaticales pueden generar estructuras complejas y anidadas en un lenguaje. Este proceso también es fundamental en la construcción de árboles sintácticos con expansiones recursivas.

Bibliografía

Introduction to the Theory of Computation por Michael Sipser

Sipser, M. (Año de publicación). Introduction to the Theory of Computation. Editorial.
Parsing Techniques: A Practical Guide por Dick Grune y Ceriel J.H. Jacobs

Grune, D., & Jacobs, C. J. H. (Año de publicación). Parsing Techniques: A Practical Guide. Editorial.
Programming Language Pragmatics por Michael L. Scott

Scott, M. L. (Año de publicación). Programming Language Pragmatics. Editorial.
Introduction to Automata Theory, Languages, and Computation por John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman

Hopcroft, J. E., Motwani, R., & Ullman, J. D. (Año de publicación). Introduction to Automata Theory, Languages, and Computation. Editorial.

Unidad I

En Fundamentos generales de programación, de Luis Joyanes Aguilar, 33-37. México: Mc Graw-Hill, 2013.

Aquilar, Luis joyanes. En Metodologia de la programación Diagramas de flujo, Algoritmos y programación Estructurada, 66-88. Mexico: Mc Graw-Hill, 1987.

En Fundamentos de Programación , de Manuel Santos. Ismael Patiño. Raul Carrasco, 37-40. México : AlfaOmega-Rama, 2006.

Chorda, Gloria de Antonio. Ramon. «Metodologia de la Programación.» 39-41. España: Ra-ma, s.f.

En Lenguaje Ensamblador para Mircrocomputadoras IBM, de J. Terry Godfrey, 73-76. México: Prentice Hall, 1991.

En Introducción a la computación y a la programación Estructurada, de Guillermo Levine, 115-118. Mexico, 1989.

En Sistema Operativos y Compiladores, de Jesus Salas Parilla, 149,150,173,174. México : Mc Graw-Hill, 1992.

Unidad II

En Compiladores, Principios, Técnicas y Herramientas , de Sethi, R, Ullman, J Aho A, 86-89,94-108,115-131,164-187,190-193,200-201,209-212,221-226,264-274,443-444. U.S.A: Adisson-Wesley Iberoamericana , 1990.

En Compiladores e Interpretes. Teoria y Practica, de M Moreno, M de la cruz, A Ortega y E Pulido, 78-85, 191-194, 199-204, 221-223. España: Pearson- Prentice Hall, 2006.

Unidad III

En Compiladores, Principios, Técnicas y Herramientas , de Sethi, R, Ullman, J Aho A, 86-89,94-108,115-131,164-187,190-193,200-201,209-212,221-226,264-274,443-444. U.S.A: Adisson-Wesley Iberoamericana , 1990.

En Compiladores e Interpretes. Teoria y Practica, de M Moreno, M de la cruz, A Ortega y E Pulido, 78-85, 191-194, 199-204, 221-223. España: Pearson- Prentice Hall, 2006.

Unidad IV

En Compiladores, Principios, Técnicas y Herramientas , de Sethi, R, Ullman, J Aho A, 86-89,94-108,115-131,164-187,190-193,200-201,209-212,221-226,264-274,443-444. U.S.A: Adisson-Wesley Iberoamericana , 1990.

En Compiladores e Interpretes. Teoria y Practica, de M Moreno, M de la cruz, A Ortega y E Pulido, 78-85, 191-194, 199-204, 221-223. España: Pearson- Prentice Hall, 2006.

En Sistema Operativos y Compiladores, de Jesus Salas Parilla, 149,150,173,174. México : Mc Graw-Hill, 1992.