zalez Morales Victor Eduardo

Fecha 14-nov

d IV - "Análisis Sintático"

del analizador sintática
independientes de contexto libre
ático descendente
ico ascendente
por precedencia de operadores
ticos LR
ambiguas
adores sintáticas

Unidad #4

o de errores

programación

sas. Por ej.
cias, expresiones

máticas, de contexto

ñadores y desarrolladores

precisas
alizadores sintáticos
tección de ambigüedades y errores
ncilla la modificación del lenguaje.

analizador léxica

secuencia de tokens según una gramática.
programa es válida, genera árbol sintáctico.

Gonzalez Morales Victor Eduardo

Fecha 14-nov

Unidad IV - "Análisis Sintático"

4.1 - El papel del analizador sintático
4.2 - Gramáticas independientes de contexto libre
4.3 - Análisis sintático descendente
4.4 - Análisis sintático ascendente
4.5 - Análisis sintático por precedencia de operadores
4.6 - Analizadores Sintáticos LR
4.7 - Uso de gramáticas ambiguas
4.8 - Generadores de analizadores sintáticos

Análisis Sintático: Conceptos, manejo de errores

Visión general de la sintaxis de lenguajes de programación

- Los lenguajes siguen reglas sintáticas precisas. Por ej. un programa tiene: Blaques, sentencias, expresiones y componentes léxicos.
La sintaxis se describe con: Gramáticas, de contexto libre o Notación BNF.

La gramática ayudan a diseñadores y desarrolladores de compiladores:
- Son especificaciones precisas
- Permiten generar analicadores sintáticos
- Facilitan la detección de ambigüedades y errores
- Hacen más sencilla la modificación del lenguaje.

Concepto de analicador léxico

- Verifica secuencia de tokens según una gramática.
- Si programa es válida, genera árbol sintático.

Gonzalez Morales Victor

Fecha 14-nov

- En práctica, dirige el proceso de compilación
- Incluye acciones semánticas para resto de fases.
- Detecta errores sintácticos y se recupera de ellos.
- Controla el flujo de tokens del analizador
- Se designa compilación dirigida por sintaxis.

Manejo de errores en compiladores.

Clasificaciones de errores:
- Léxicos: errores en identificadores, palabras clave, operadores.
- Sintácticos: paréntesis o expresiones mal estructurados.
- Semánticos: operadores en operandos incompatibles.

Errores lógicos y de corrección:
- No siempre pueden detectarse
- Requieren análisis de intenciones o flujo de programa

Los compiladores no se fijan en esos errores:

- Errores de sintaxis impiden crear el árbol sintáctico.
- El manejo de errores desde el inicio mejora la estructura y respuesta del compilador.
- Un buen manejo ayuda a localizar y describir errores.

Manejo de errores de sintaxis en compiladores:

Objetivo: Recuperarse de l error y continuar compilación.
El analizador debe:

- Indicar errores clara y precisamente
- Recuperarse para seguir analizando entrada
- Distinguir errores y advertencias
- Evitar ralentizar compilación.

Gonzalez Morales Victor Eduardo

Fecha 14-11

Estrategias de manejo de errores

Ignorar el problema (Panic Mode).
- Ignora tokens hasta una condición segura (';' o 'End')
- Desecha tokens entre el error y token seguro
- Continúa análisis desde condición segura

Recuperación a nivel de frase
- Corrige error insertando tokens (';').
- Cuidado: podría provocar recuperaciones infinitas si la corrección introduce errores nuevos.

Reglas de producción adicionales
- Agrega reglas gramaticales para errores comunes.
- Permite dirección y corrección automática de errores.
- Emite advertencias en vez de errores Si es posible.

Corrección global
- Genera árbol sintáctico completo a partir de secuencia
- Devuelve versión corregido de entrada inicial.
- Crea árbol sintáctico para secuencia sin errores

Gramáticas en un analizador sintáctico:
- Utiliza gramáticas de contexto libre para reconocer sentencias.
- Definición de una gramática G.
  N: no terminales
  T: terminales
  P: Reglas de producción
  S: Axioma inicial

Pueden existir múltiples derivaciones en una pseudocadena:
- Izq: reescribe el no terminal más a la izquierda
- Der: reescribe el no terminal más a la derecha.

Rayter

Gonzalez Morales Victor

Fecha 14-11

Derivaciones en gramática:
- Una regla de producción es una regla de reescritura.
- Un no terminal se reemplaza por la pseudocadena en el lado derecho.
- Pseudocadena: Secuencia de terminales y/o no terminales.
- Notación de derivación: $\alpha \Rightarrow \beta$.
- Las pseudocadenas derivadas del axioma inicial son formas sentenciales.

Árbol sintáctico en lenguajes de programación:
- Un árbol sintáctico representa la estructura de una sentencia.
- La raíz es el axioma inicial de la gramática.
- Los nodos internos son los no terminales de las reglas de producción.
- Cada nodo tiene tantos hijos como símbolos en la regla aplicada.
- Si la sentencia es incorrecta, no se genera el árbol.
- Ambigüedad: Una sentencia admite múltiples árboles.
- La ambigüedad se resuelve cambiando la gramática o aplicando reglas.

Gramática y Derivaciones 16-Nov

Gramática no ambigua que reconoce expresiones aritméticas.

$N = \{E, T, F\}$
$T = \{id, num, +, *, (,)\}$
$S = E$

$P = \{E \to E + T \mid T$
$T \to T * F \mid F$
$F \to id \mid num \mid (E)\}$

Objetivo: Construir la cadena de tokens: $id_1 * id_2 + id_3$

Gonzalez Morales Victor Eduardo

Fecha 16-nov

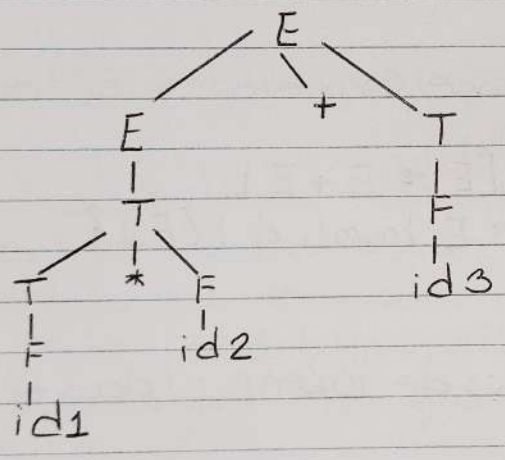
Derivaciones a izquierda

$E \Rightarrow E + T \Rightarrow T + T \Rightarrow T * F + T \Rightarrow F * F + T \Rightarrow id_1 * F + T \Rightarrow id_1 * id_2 + T \Rightarrow id_1 * id_2 + F \Rightarrow id_1 * id_2 + id_3$

Derivaciones a derecha

$E \Rightarrow E + T \Rightarrow E + F \Rightarrow E + id_3 \Rightarrow T * F + id_3 \Rightarrow T + id_2 * id_3 \Rightarrow F + id_2 * id_3 \Rightarrow id_1 + id_2 * id_3$

Árboles sintácticos



Gramática y derivaciones

Gramática ambigua que reconoce expresiones aritméticas

$N = \{E\}$

$T = \{id, num, +, *, (,)\}$

$S = E$

$P = \{E \rightarrow E + E \mid E * E \mid num \mid id \mid (E)\}$

Objetivo: construir la cadena de tokens: $id_1 * id_2 + id_3$

Derivaciones a izq: $E \Rightarrow E + E \Rightarrow E * E + E \Rightarrow id_1 * E \Rightarrow id_2 + E \Rightarrow id_1 * id_2 + id_3$

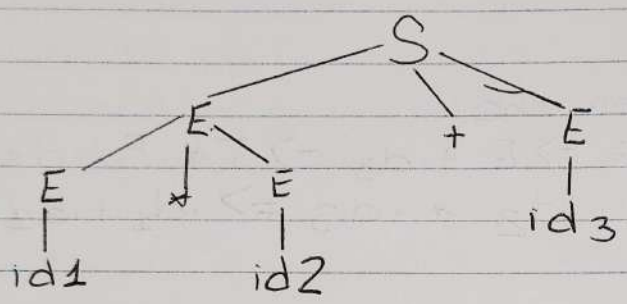
Gonzalez Morales Victor

Fecha 16-nov

Derivaciones a derechas

$E \Rightarrow E + E \Rightarrow E + id_3 \Rightarrow E * E + id_3 \Rightarrow E * id_2 + id_3 \Rightarrow id_1 * id_2 + id_3$

Árboles Sintácticos



Gramática y derivaciones

Gramática ambigua que reconoce expresiones aritmeticas

$N = \{E\}$

$T = \{id, num, +, *, (,)\}$

$S = E$

$P = \{E \rightarrow E + E \mid E * E \mid num \mid id \mid (E)\}$

Objetivo construir la cadena de tokens: idaux+ idcont +id1

Derivaciones a derecha

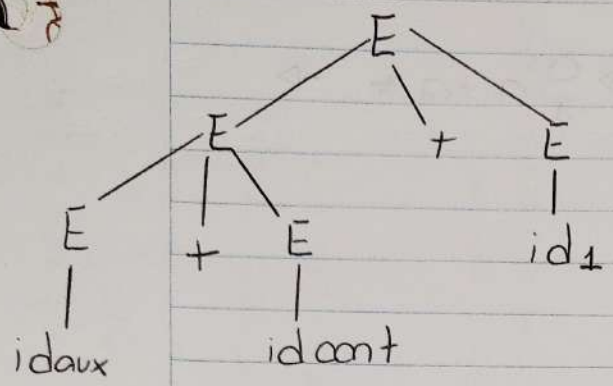
$E \Rightarrow E + E \Rightarrow E + id_1 \Rightarrow E + E + id_1 \Rightarrow E + idcont + id_1 \Rightarrow idaux + idcont + id_1$

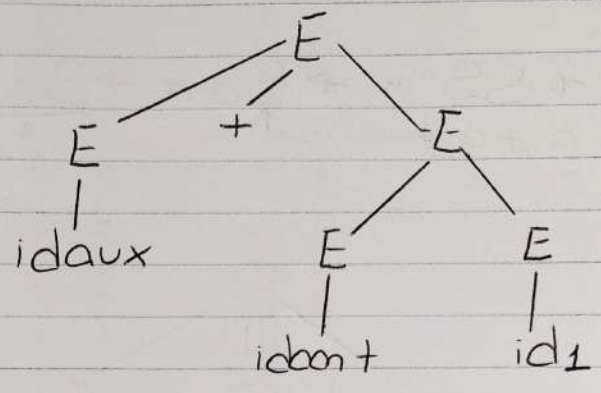
Derivaciones a izquierda

$E \Rightarrow E + E \Rightarrow idaux + E \Rightarrow idaux + E + E \Rightarrow idaux + idcont + E \Rightarrow idaux + idcont + id_1$

Gonzalez Morales Victor

Fecha 16-nov





Considerando la siguiente gramática libre de contexto:

$S \rightarrow SS+ \mid SS* \mid a$

Sea la cadena: aa +a *

a) Realice la derivación a derecho para la cadena
b) Realice la derivación a izquierda para la cadena
c) Diseñe un árbol sintáctico para la cadena
¿Es ambigua esta gramática? Justifica la respuesta

Considerando la gramática libre de contexto.
$S \rightarrow 0S1 \mid 01$
Sea la cadena 000111
Responda los incisos y pregunta del problema 1

Considerando la gramática libre de contexto
$S \rightarrow S+S \mid SS \mid (S) \mid S*a$
Sea la cadena (a +a)+a
Responda los incisos y preguntas del problema 1

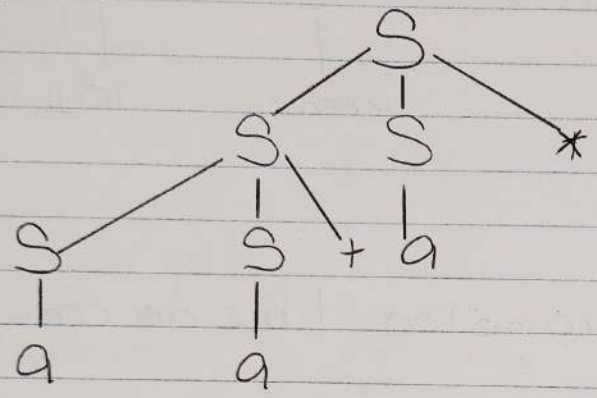
Gonzalez Morales Victor

Fecha 19-nov

a)

$S \rightarrow SS* \rightarrow Sa* \rightarrow SS+a* \rightarrow Sa+a* \rightarrow$

aa+a*



b) $S \rightarrow SS* \rightarrow SS+S* \rightarrow aS+S*$

$\rightarrow aa+S* \rightarrow aa+a*$