UNIVERSIDAD TECNOLÓGICA DE SANTIAGO, UTESA

SISTEMA CORPORATIVO
FACULTAD DE INGENIERIA Y ARQUITECTURA
CARRERA DE INGENIERIA EN SISTEMAS COMPUTACIONALES

ASIGNATURA:

Compiladores
INF-920-001

TAREA SEMANA 5:

Análisis Sintáctico y Semántico

PRESENTADO A:

Iván Mendoza

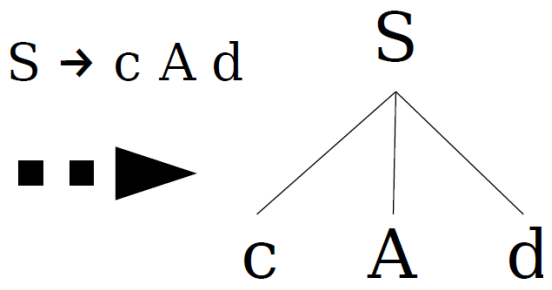
PRESENTADO POR:

Liván Herrera (2-16-0686)

Santiago de los Caballeros
República Dominicana
Febrero, 2022

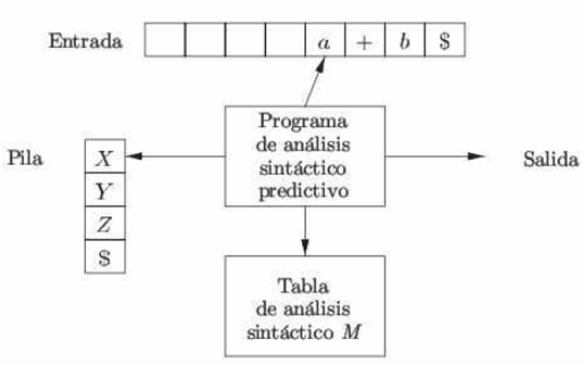
Investigar

- **Análisis Sintáctico Descendente**



El análisis sintáctico descendente, intenta descubrir la ramificación entre las producciones de gramática por la izquierda del símbolo inicial para una cadena de entrada. Para poder elaborar este tipo se realizan desde un inicio algunas operaciones para que la gramática se considere LL1, las cuales son: Eliminar la ambigüedad y a su vez eliminar la recursividad por la izquierda.

- **Analizador recursivo predictivo.**



Existen varios métodos utilizados para el análisis sintáctico. La mayoría pueden ser ascendentes o descendentes. Los ascendentes construyen el árbol desde las hojas hacia la raíz y los descendentes van desde la raíz hasta llegar las hojas. Este método contiene algo denominado las “no terminales de la gramática” que poseen un procedimiento asociado cada una. El símbolo de preanálisis determina “sin ambigüedad” el procedimiento que ha sido elegido para cada no terminal y el procedimiento adicional parea.

- **Analizador con tablas (LL).**

2. Para cada terminal *a* del First (α), añádase A → α en la posición M[A , *a*].

Símbolo No Terminal	First
S	(
A	x , (
B	; , ε
C	x , (

S → '(' A ')'
A → C B
B → ';' A | ε
C → 'x' | S

	;	x	(	)	$
S			S → (A)		
A		A → CB	A → CB		
B	B → ; A				
C		C → x	C → S		

Una tabla para el analizador sintáctico de tipo LL es bidimensional. Las filas de la misma se etiquetas con los no terminales y las columnas con los terminales de la gramática sobre la cual se basa el analizador sintáctico. Adicional a esto, se le añade la columna FDC, la cual es el fin de cadena.

- **Análisis Semántico.**

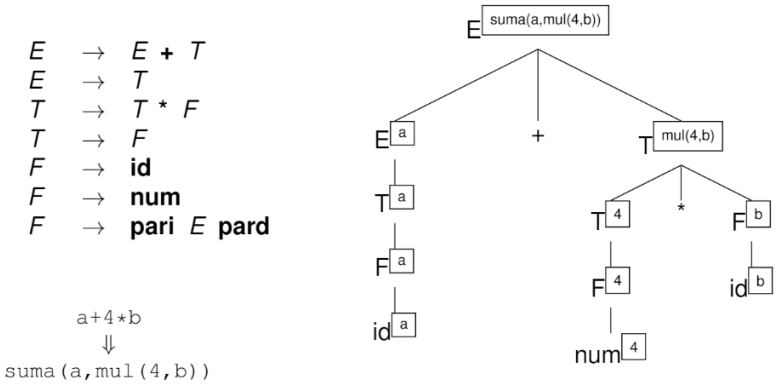
Es la fase del compilador en donde se comprueba la corrección semántica del programa, además de que posee subrutinas que son independientes. Estas pueden ser invocadas por los analizadores morfológicos y sintácticos.

- **Gramáticas de atributos.**

REGLAS GRAMATICALES	REGLAS SEMÁNTICAS
decl → type var-list	var-list.$dtype$ = type.$dtype$
type → int	type.$dtype$ = integer
type → float	type.$dtype$ = real
var-list$_1$ → id, var-list$_2$	id.$dtype$ = var-list$_1$.$dtype$
	var-list$_2$.$dtype$ = var-list$_1$.$dtype$
var-list → id	id.$dtype$ = var-list.$dtype$

Esta es un tipo de gramática independiente del contexto en la que sus símbolos terminales y no terminales se les dota de unos atributos. El fin de esta es poder conocer un determinado valor de un atributo cualquier parte del árbol de derivación y tomar la mejor decisión.

- **Traducción dirigida por la sintaxis.**



En la traducción dirigida por la sintaxis, se asocia información a la construcción del lenguaje asociando algunos atributos a distintos símbolos de la gramática. De esta manera, los valores de los atributos se calculan mediante reglas semánticas.

- **Comprobaciones Semánticas.**

  Nos permiten comprobar 3 puntos claves a la hora de trabajar nuestro código, como son el uso correcto de las palabras reservadas, que cada variable posea un valor correspondiente a su tipo y que las funciones y ciclos se usen de manera correcta.

Ejercicios:

1) Crear un analizador semántico de un compilador. (Utilizar cualquier herramienta) en el lenguaje de programación de su preferencia.

2) Subir el código a GitHub, enviar el código fuente y el ejecutable del proyecto.

https://github.com/livanh1/AnalizadorSemantico.git