Resumen Clase 1 – Sintaxis

Criterios para evaluar lenguajes de programacion:

-Simplicidad y legibilidad:

-Claridad de los bindings: Los elemento de un lenguaje pueden ligarse a sus atributos o propiedades en diferentes momentos (escritura,compilacion,ejecucion,etc.). Esta ligadura debe ser clara.

-Confiabilidad: Esta esta relacionada con la seguridad. Chequeo de tipos y manejo de excepciones.

-Soporte: Deberia ser accesible para cualquiera y deberia poder ser implementado en diferentes plataformas

-Abstraccion: Es la capacidad de definir y usar estructuras u operaciones complicadas de manera que sea posible muchos de los detalles

-Ortogonalidad: Esta refiere a la posibilidad que ofrece el lenguaje de poder combinar sus elementos sin producir errores

-Eficiencia: Esta se relaciona con: tiempo y espacio, esfuerzo humano y optimizable.

SINTAXIS:

Semantica: Conjunto de reglas para dar significado a los programas sintacticamente validos.

Sintaxis: Conjunto de reglas que definen como componer letras, digitos y otros caracteres para formar los programas.

La sintaxis establece reglas que definen cómo deben combinarse las componentes básicas, llamadas “word”, para formar sentencias y programas.

La sintaxis debe ayudar al programador a escribir programas correctos sintacticamente. Esta establece reglas que sirven para que el programador se comunique con el procesador.

Elementos de la sintaxis:

Alfabeto o conjunto de caracteres: Conjunto finito de simbolos admitidos en el lenguaje. La secuencia de bits que compone cada carácter la determina la implementacion.

Identificadores: Cadena de letras y digitos, que deben comenzar con una letra.

Operadores suma, resta, etc.

Comentarios: hacen los programas mas legibles.

Palabra clave y palabra reservada: array, do, else, if. Una palabra clave tiene un significado dentro de un contexto y una palabra reservada, es una palabra clave que ademas no puede ser usada por el programador como identificador de otra entidad.

Estructura sintactica:

Vocabulario: Conjunto de caracteres y palabras necesarias para construir expresiones, sentencias y programas. Ej: identificadores, operadores, palabras claves, etc.

Expresiones: Construcciones sintacticas compuestas de operadores y operandos, de cuya evaluacion se obtiene un valor.

Sentencias: Componente sintactico mas importante.

Reglas lexicas y sintacticas:

Reglas lexicas: Conjunto de reglas para formar las “word”, a partir de los caracteres del alfabeto.

Reglas sintacticas: Conjunto de reglas que definen como formar a partir de esas palabras, las “expresiones” y “sentencias”

Tipos de sintaxis:

Abstracta: Se refiere basicamente a la estructura

Concreta: Se refiere basicamente a la parte lexica

Pragmatica: Se refiere basicamente al uso practico

Diferentes formas de escribir un while, varian en la parte concreta que seria el lexico pero son iguales respecto a la sintaxis abstracta, ya que son un while.

Como definir la sintaxis: existen diferentes formas para esto. Usar lenguaje natural (Fortran), usar gramaticas libres de contexto (BNF y EBNF), usar diagramas sintacticos (equivalentes a BNF pero mas intuitivos) y usar arboles de analisis sintactico.

BNF: Es una notacion formal para describir la sintaxis, es un metalenguaje que utiliza metasimbolos ( <, >, ::= por ej) este define las reglas por medio de “producciones”, ej: <digito> ::= 0|1|2|3|4|5|6|7|8|9.

Gramatica: Conjunto de reglas finita que define un conjunto infinito de posibles sentencias validas en el lenguaje. Una gramatica esta formada por una 4-tupla.

G = (N,T,S,P)

Donde:

N = Conjunto de simbolos no terminales

T = Conjunto de simbolos terminales

S = Simbolo distinguido de la gramatica que pertenece a N

P = Conjunto de producciones

Gramaticas libre de contexto y sensibles al contexto: Una gramatica libre de contexto es aquella en la que no realiza un analisis del contexto y una sensible al contexto analiza este tipo de cosas.

No todas las oraciones que se pueden armar con los terminales son validas, se necesita de un metodo de analisis (reconocimiento) que permita determinar si un string dado es valido o no en el lenguaje: parsing.

El parse, para cada sentencia construye un “arbol sintactico o arbol de derivacion”

Arboles Sintacticos: Existen dos maneras de construirlo. Metodo botton-up (de izq a der y de der a izq) y Metodo top-down (de izq a der y de der a izq).

{Darle una leida al tema de la gramatica y la 4-tupla es como un concepto no tan complejo pero entiendo que es importante pq como que se forma toda la sintaxis o las posibilidades, esta interesante}

Producciones recursivas: Son las que hacen que el conjunto de sentencias descriptas sea infinito.

GN = ( N, T, S, P) N = { <natural>, <digito> }, T = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }, S = <natural>, P = { <natural> ::= <digito> | <digito> <natural>, <digito> , ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}

Cualquier gramatica que tiene una produccion recursiva describe un lenguaje infinito.

Regla recursiva por la izquierda: La asociatividad es por la izquierda, el simbolo no terminal de la parte izquierda de una regla de produccion aparece al comienzo de la parte derecha

Regla recursiva por la derecha: La asociatividad es por la derecha, el simbolo no terminal de la parte izquierda de una regla de produccion aparece al final de la parte derecha

Gramaticas ambiguas: Una gramatica es ambigua si una sentencia puede derivarse de mas de una forma.

Subgramaticas: “La filosofia de composicion es la forma en que trabajan los compiladores”. Basicamente que una gramatica se puede conformar por otras gramaticas, entiendo.

EBNF: Es otra forma de describir la sintaxis libres de contexto, esta gramatica es la BNF extendida y los meta simbolos incorporados son:

[] = Elemento optativo puede o no estar

(|) = selección de una alternativa

{} = repeticion

\*0 o mas veces +1 o mas veces

Diagramas sintacticos (CONWAY): Es un grafo sintactico, cada diagrama tiene una entrada y una salida, y el camino determina el analisis, cada diagrama representa una regla o produccion. Para que una sentencia sea valida, debe haber un camino desde la entrada hasta la salida que la describa. Este se visualiza mejor que BNF o EBNF