TIPOS DE GRAMÁTICAS

TIPO 0 (sin restricciones)

Axioma o símbolo inicial de la gramática

$G_3 = (\{0,1\}, \{A,B,S\}, S, P)$ (producciones)

(Σ_T: $\{0,1\}$, Σ_N: $\{A,B,S\}$)

donde $P_3 = \{(S \to AO), (AO \to 1B1), (1A \to 0B0), (B \to \lambda), (B \to 1), (B \to 0)\}$

$S \to AO$

$A \to 1B1$

$1A \to 0B0$

$B \to \lambda \mid 1 \mid 0$

TIPO 1 (dependiente del contexto)

$xAy \to xvy$

$G_4 = (\{0,1\}, \{A,B\}, A, P)$

$P = \{(A \to 1B1), (A \to 11), (1B1 \to 101), (1B1 \to 111)\}$

Tipo 2 (Independientes del contexto)

$G_5 = (\{0,1\}, \{A,B\}, A, P)$ ::=

$P = \{(A \to 1B1), (A \to 11), (B \to 1), (B \to \emptyset)\}$

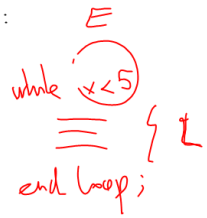
Tipo 3 (Regulares o lineales)

$A \to Ba$

$A \to a$

Ejemplo 1: Los bucles en Ada se describen de la siguiente manera:

[while <exp>]
  for <var> in [reverse] <i>..<j>] loop
    [<sentencia> ;]+
end loop;



Donde <exp> es una expresión booleana
<var> es un identificador de variable o un número
<i> y <j> representan el rango de valores que va a tomar la variable
Sentencia, puede ser cualquier sentencia del lenguaje, incluidos null u otros bucles anidados
Ejemplos de operaciones pueden ser op1, op2, op3, etc.

Escribir una gramática que se corresponda con el lenguaje dado

Sol:

S → W L | F L | E op E
W → while E loop L end loop ;
F → for T in R T..T loop L end loop ;
T → var | num
R → reverse | λ
L → S; | S; L | λ ← null
E → E op E | (E) | num | var

F → E O E
O → < | > | <> | + | * | / | \

ERRORES TIPICOS
1: Definir variable y número (var → a-z A-Z ; num → 0|1|2|3)
2: No definir S, o E
3: Definir op y no tener en cuenta los operandos
4.- No contemplar el ; de separación entre sentencias

Ejemplo 2: La definición de funciones en cierto lenguaje es de la siguiente manera:

tipo function nombre (tipo : nombre [; tipo : nombre]*)
declare {
[tipo : nombre [; nombre]* ; | decl_funcion]+
}
begin {
[sentencia]+
}

Donde
- tipo puede ser integer o char
- nombre comienza por una letra y puede ir seguido por cualquier cantidad de letras o números.
- Sentencia, puede ser op1, op2, op3 ← E op E
- declaración de funciones puede ser recursiva (funciones dentro de funciones) por lo que decl_función representa la declaración de una nueva función dentro de la actual.

Crear una gramática que represente la declaración de funciones en este lenguaje.

Sol:

argumento — Lista de argumentos

S → T function N (A L) declare { D } { begin } E { }

T → integer | char

A → T : N

L → ; A L | λ

¿como contemplo que solo haya un argumento?

D → A L ; | S

N → id

E → E op E | (E) | num | λ