Trabajo Practico de Traductor Dirigido por la Sintaxis

CI.: 5.574.180
Nombre: Andres
Apellido: Heo Kim

Tema: Dado una lista de precios, se pide obtener el valor correspondiente al precio promedio y la cantidad de precios que se tomaron en consideración (elementos en la lista).

Inicialmente planteamos que la lista de números está separado por comas (,) y que los precios son números enteros.

Ejemplo de entrada de la lista de precios valida:
10000 , 40000 , 5000
Serian 3 precios con un promedio de 18333

Empezamos el análisis de como vamos a leer la estructura de lista de precios con su separador la coma (,).

Tenemos un BNF inicial para leer lista de precios:
LISTA -> PRECIO , LISTA
LISTA -> PRECIO

y el BNF de precio seria un numero entero, basicamente una cadena consecutiva de digitos:

PRECIO -> DIGITO PRECIO
PRECIO -> DIGITO
DIGITO -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Escribimos todo el BNF en una columna:

TDS -> LISTA
LISTA -> PRECIO **,** LISTA | PRECIO
PRECIO -> DIGITO PRECIO | DIGITO
DIGITO -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Verificamos que no tenga recursión por la izq., en este caso no tenemos.
Verificamos factores comunes: en este caso tenemos en “lista” y en “precio”

LISTA -> PRECIO **,** LISTA | PRECIO

LISTA -> PRECIO R1
R1 -> **,** LISTA | ε

PRECIO -> DIGITO PRECIO | DIGITO

PRECIO -> DIGITO R2
R2 -> PRECIO | ε

Verificamos el conjunto primero:

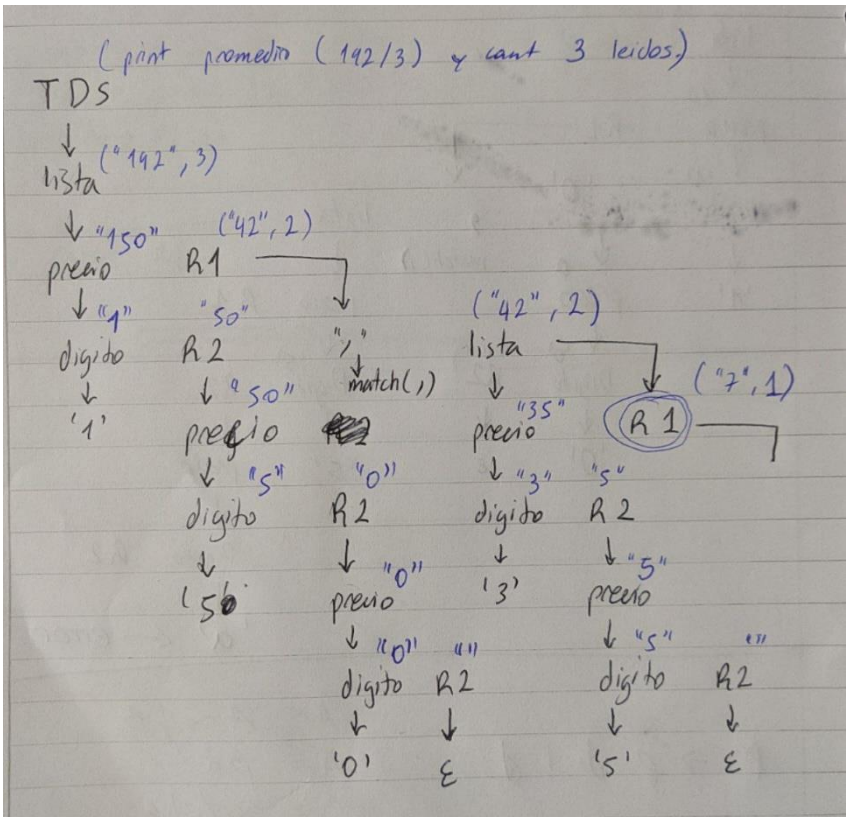
BNF	Conjunto Primero
TDS -> LISTA	P(TDS) = P(LISTA) = { 0 1 2 3 4 5 6 7 8 9 }
LISTA -> PRECIO R1	P(LISTA) = P(PRECIO R1) = { 0 1 2 3 4 5 6 7 8 9 }
R1 -> **,** LISTA	P(R1) = P(**,** LISTA) U P(ε) = { **,** } U { ε }
PRECIO -> DIGITO R2	P(PRECIO) = P(DIGITO R2) = {0 1 2 3 4 5 6 7 8 9 }
R2 -> PRECIO | ε	P(R2) = P(PRECIO) U P(ε) = {0 1 2 3 4 5 6 7 8 9 } U { ε }
DIGITO -> 0 | 1 | … | 9	P(DIGITO) = {0 1 2 3 4 5 6 7 8 9}

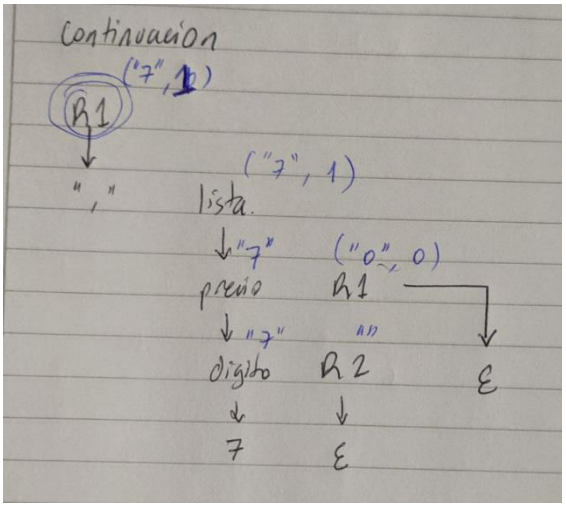
Ya que corroboramos que la gramática es predictiva (viendo solamente la recursión por la izq y conjunto primero), escribimos el BNF junto a las Reglas emanticas:

BNF	Reglas Semanticas
TDS -> LISTA	LINEA < entrada.length() ? error() : print(LISTA.numero / LISTA.contador, LISTA.contador)
LISTA -> PRECIO R1	LISTA.numero = PRECIO.numero1 + R1.numero2 != “” ? R1.numero2 : 0 LISTA.contador = 1 + R1.contador
R1 -> **,** LISTA	R1.numero = LISTA.numero R1.contador = LISTA.contador
R1 -> e	R1.numero= "" R1.contador = 0
PRECIO -> DIGITO R2	PRECIO.numero = DIGITO.numero1 || R2.numero2
R2 -> PRECIO	R2.numero = PRECIO.numero
R2 -> ε	R2.numero = ""
DIGITO -> 0	DIGITO = “0”
…	…
DIGITO -> 9	DIGITO = “9”

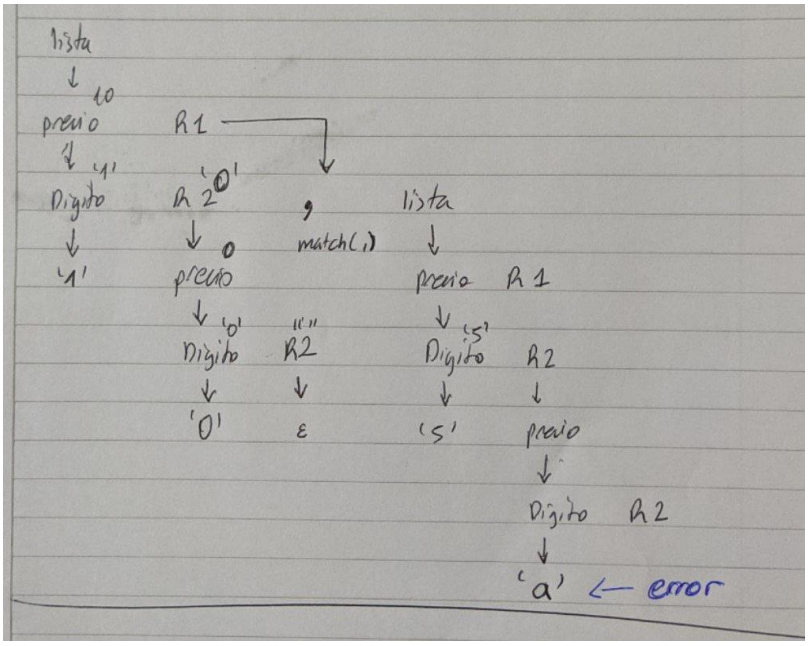
Ejemplos de Arboles Sintacticos

a. para una Entrada Valida: 150, 35, 7





b. para una Entrada Nó Valida: 10, 5a, 90



Ahora que tenemos el BNF, Conjunto Primero y las Reglas Semanticas procedemos a codificar nuestro TDS, para este caso estamos usando el lenguaje Java.

Definimos 3 variables globales, uno para guardar la entrada, otro de utilidad para imprimir LOGs y el puntero a la entrada.

```
public static String entrada = "0000001,0002,003";

// Entrada para habilitar mensaje de logs
public static final boolean PRINT_INFO_LOGS = false;

// Puntero para ir consuimendo la entrada
static int LINEA = 0;
```

Estructura Auxiliar para hacer retorno de variables compuestas:

```
// Clase Auxiliar para retornar 2 variables
public static class Valores {

    String numero;
    int contador;

    @Override
    public String toString() {
        return "n: " + this.numero + ", c:" + this.contador;
    }
}
```

Funciones Match, Input y Error:

```
/** Metodo del match, encargado de machear la entrada y controlar que este ...
static void match(char terminal) {
    if(PRINT_INFO_LOGS) System.out.println("\nMatch();  terminal='"
            + terminal + "', input = '" + input() + "'");
    if (terminal != input()) {
        error();
    }
    LINEA++;
}

static char input() {
    if (LINEA < entrada.length()) {
        if(PRINT_INFO_LOGS) System.out.print(" " + entrada.charAt(LINEA));
        return entrada.charAt(LINEA);
    } else {
        if(PRINT_INFO_LOGS) System.out.print(" $");
        return '$';
    }
}

/** Metodo para imprimir mendaje de "Error de Sintaxis" ...3 lines */
static void error() {
    System.err.println("Error de Sintaxis en la linea " + LINEA
            + " sobre el caracter: '" + input() + "'");
    System.exit(0);
}
```

Metodo Main:

```
// Metodo Main del Traductor Dirigito por la Sintaxis
public static void main(String[] args) {
    TDS();
}
```

Producciones:

```
static void TDS() {
    entrada = entrada.replaceAll(" ", "");
    if(PRINT_INFO_LOGS) System.out.println("entrada sin espacios: " + entrada);

    Valores retorno = LISTA();
    if(PRINT_INFO_LOGS) System.out.println("Procesado hasta la linea: " + LINEA);

    if (LINEA < entrada.length()) {
        error();
    } else {
        if(PRINT_INFO_LOGS) System.out.println("Lista(): " + retorno);
        int promedio = Integer.parseInt(retorno.numero) / retorno.contador;
        System.out.println("TDS();    promedio: " + promedio + "  cant. precios leidos: " + retorno.contador);
    }
}
```

```
static Valores LISTA() {
    Valores lista = new Valores();
    String numero1, numero2;
    int contador;

    if (input() == '0' || input() == '1' || input() == '2' || input() == '3' || input() == '4' ||
            input() == '5' || input() == '6' || input() == '7' || input() == '8' || input() == '9') {
        numero1 = PRECIO();
        Valores r1 = R1();
        numero2 = r1.numero;
        contador = r1.contador;
        lista.numero = String.valueOf(Integer.parseInt(numero1) + (numero2 != "" ? Integer.parseInt(numero2) : 0));
        lista.contador = 1 + contador;
        return lista;
    } else {
        error();
        return null;
    }
}
```

```
static Valores R1() {
    Valores r1 = new Valores();
    if (input() == ',') {
        match(',');
        Valores lista = LISTA();
        if(PRINT_INFO_LOGS) System.out.println("Lista(): " + lista);
        r1.numero = lista.numero;
        r1.contador = lista.contador;
        return r1;
    } else {
        if(PRINT_INFO_LOGS) System.out.println("Fin de Recurcion en R1");
        r1.numero = "";
        r1.contador = 0;
        return r1;
    }
}
```

```
static String PRECIO() {
    String numero, numero1, numero2;
    if (input() == '0' || input() == '1' || input() == '2' || input() == '3' || input() == '4' ||
            input() == '5' || input() == '6' || input() == '7' || input() == '8' || input() == '9') {
        numero1 = DIGITO();
        numero2 = R2();
        numero = numero1.concat(numero2);
        return numero;
    } else {
        error();
        return null;
    }
}
```

```
static String R2() {
    String numero;
    if (input() == '0' || input() == '1' || input() == '2' || input() == '3' || input() == '4' ||
            input() == '5' || input() == '6' || input() == '7' || input() == '8' || input() == '9') {
        numero = PRECIO();
        return numero;
    } else {
        if(PRINT_INFO_LOGS) System.out.println("Fin de Recurcion en R2");
        numero = "";
        return numero;
    }
}
static String DIGITO() {
    if (input() == '0') {
        match('0');
        return "0";
    } else if (input() == '1') {
        match('1');
        return "1";
    } else if (input() == '2') {
        match('2');
        return "2";
    } else if (input() == '3') {
        match('3');
        return "3";
    } else if (input() == '4') {
        match('4');
        return "4";
    } else if (input() == '5') {
        match('5');
        return "5";
    } else if (input() == '6') {
        match('6');
        return "6";
    } else if (input() == '7') {
        match('7');
        return "7";
    } else if (input() == '8') {
        match('8');
        return "8";
    } else {
        match('9');
        return "9";
    }
}
```

Para el caso que se quiera probar el TDS, tendra en un repositorio publico el codigo para probarlo. IDE usado seria NetBeans 8.2 sobre Java 8

Link: https://github.com/heokim/compiladores-tp1-tds/blob/main/src/tds/TDS.java