ISIS-1106 Lenguajes y Máquinas
Proyecto parte 1
POR PAREJAS

Muchos de los lenguajes de programación se pueden definir por medio de una gramática independiente del contexto. Como vimos en clase, los lenguajes generados por estas gramáticas corresponden a los lenguajes reconocidos por los autómatas de pila. Esto quiere decir que podemos utilizar un autómata de pila para reconocer si dado un lenguaje de progrmación específico, un programa es valido o no.

En el taller 1, ustedes definieron una gramática para definir redes de Petri. Un ejemplo de una descripción de una red de Petri se muestra a continuación. (En este caso, vamos a trabajar solo con minúsculas.)

```
p_red nombreejemplo
    var  x = 5
    var     y = 6
    var     p = 1
    var     inf = 10000


        sitio: coladeespera
               capacidad:= inf
               marcacion_i:= 0
        sitio: procesadores
               capacidad:=p+p
               marcacion_i:=capacidad
        sitio: trabajando1
               capacidad:=  1
               marcacion_i:= 0
        sitio: trabajando2
               capacidad:=  2
               marcacion_i:= 0
        sitio: sitiodeconteo
               capacidad:=  inf
               marcacion_i:= 0


        transicion (entrar, exponencial, x/3,  (x+4)/3)

        transicion atender1
        transicion atender2


        transicion (terminar1,deterministico,  y+(-y/2))
        transicion (terminar2,uniforme,1,3)


        arco(entrando:  entrar, coladeespera)
        arco(coladeespera, atender1, 1)
        arco(procesadores, atender1)

        arco(coladeespera, atender2, 1)
```

```
arco(procesadores, atender2, 2)

arco(atender2, trabajando2, 1)
arco(atender1, trabajando1, 1)

arco(trabajando2, terminar2, 1)
arco(trabajando1, terminar1, 1)


arco(terminar1, procesadores, 1)
arco(terminar2, procesadores, 1)

arco(terminar1, conteo, 1)
arco(terminar2, conteo, 2*p)
```

```
fin_red
```

El primer paso para reconocer el lenguaje es tomar un programa LISP y hacer su análisis léxico. El resultado de éste proceso lleva a una cadena de tokens que deben corresponder a los símbolos terminales de la gramática independiente del contexto.

El lexer se puede implementar utilizando un autómata con respuesta (en los estados o transiciones) que genera los tokens a partir de la cadena de caracteres describiendo el programa de entrada. Cada token debe describirse con un único caracter.

n para nombres

para números

P para p_red

S para sitio

C para capacidad

I para marcacion_i

T para transicion

X para exponencial

D para deterministico

A para arco

F para fin_red

Q para :=

Los demás símbolos son de un único carácter y se dejan igual.

Se incluye un proyecto eclipse en el cual se implementa un lexer y un parser para un lenguaje sencillo. En el proyecto hay un directorio docs donde encuentran una presentación que explica el proceso que se siguió para construir el lexer-parser.

En el proyecto eclipse encontraran los siguientes archivos gold:

LExerParser.gold

Z: ignoren ete archivo (es el que causa error en el archov src)

BONO:

Su programa, además de reconocer el lenguaje, debe validar que se están definiendo arcos entre transiciones y sitios previamente definidos y que no se esten relacionando mediante un arco 2 transiciones o 2 sitios. También debe verficar que las variables están definidas.

Para hacer esto, puede ser necesario dividir el trabajo en etapas para ir llenando una tabla de símbolos.