MANUAL DE USUARIO

La herramienta Wison es un analizador LL1 que nos permite crear gramáticas que podemos usar para analizar cadenas de entrada.
Siendo wison una herramienta montada en angular, veremos la sintaxis para poder usarla.
Entonces la herramienta wison no permitirá entrar cadenas que se detecte siendo recursivas hacia la izquierda, de la forma A->Aa|b, o que se puedan factorizar, es decir que de una producción A->aB |aB1, donde ya es un terminal igual. Por lo que si se detecta uno de estos casos la herramienta wison entonces no aceptara la gramática e informará al usuario de los problemas con la gramática.
Al generar la gramática recuerda agregarle tu nombre sino será creado como “defaultParser”, entonces cuando el mensaje de texto de que la gramática fue creada podrás ver en el inferior de la página un selector de parsers y un cuadro donde podras colocar la gramática, dicha gramática maneja errores sintácticos de a 1, y los errores léxicos los Ignorará. Si la cadena es válida saldrá un mensaje mencionando que es válida, de lo contrario indicará que salió mal y cuál puede ser la solución.

GRAMÁTICA WINDIGO

Antes de manejar la sintaxis Windigo es importante que de primero aprendas que en la parte léxica se tomará de preferencia las primeras reglas, por lo que si aplicaste reglas de la forma {$_Terminal} entonces te recomendamos que coloques el terminal en las primeras reglas, sin embargo tras varios terminales detectados la herramienta decidirá la expresión más larga.
Para los No Terminales es importante que durante su producción no que generes gramáticas ambiguas, en este caso se maneja comprobación de recursividad a la izquierda y posible factorización, si la gramática escrita posee alguna de estas cualidades entonces se procederá a mencionar en qué producción se encuentra el error y a denegar el acceso. Los no terminales empiezan de la forma %_No_Terminal .
Datos Extras: Para la concatenación de expresiones se tiene que utilizar entre paréntesis, la estructura [aA-zZ] acepta todas las letras y la estructura [0-9] acepta todos los números, se pueden agregar a cada expresión que no sea (‘Hola’) +,*,?, si se desea agregar se debe agregar el terminal que posee la propiedad (‘Hola’) en ($_Terminal)

La sintaxis entonces para manejar la gramática windigo es la siguiente:

```
#Esto es un comentario de línea

#Estructura Wison

Wison ¿
Lex {:
     /**
```

```
      Esto es un
      Comentario de bloque
    */

    # Declaración de terminales de la forma:
    # Terminal $_NOMBRE <- EXPRESIÓN ;

    Terminal $_Una_A     <- ‘a’ ;    # cualquier carácter alfanumérico por
separado
    Terminal $_Mas       <- ‘+’ ;    # cualquier carácter especial por
separado
    Terminal $_Punto     <- ‘.’ ;    # cualquier carácter especial por
separado
    Terminal $_P_Ab      <- ‘(’ ;    # cualquier carácter especial por
separado
    Terminal $_P_Ce      <- ‘)’ ;    # cualquier carácter especial por
separado
    Terminal $_FIN       <- ‘FIN’;   # cualquier palabra reservada
    Terminal $_Letra     <- [aA-zZ]; # alfabeto completo en mayusculas y
minusculas
    Terminal $_NUMERO    <- [0-9] ;  # Digitos del 0 al 9
    Terminal $_NUMEROS   <- [0-9]* ; # Estrella de Kleene para hacer 0 o n
veces
    Terminal $_NUMEROS_2 <- [0-9]+ ; # Cerradura positiva para hacer 1 on
veces
    Terminal $_NUMEROS_3 <- [0-9]? ; # reconoce la cláusula ? para hacer 0
o 1 vez
    Terminal $_Decimal   <- ([0-9]*)($_Punto)($_NUMEROS_2); # terminal
combinado

:}
```

```
#Esto es un comentario de línea

#Estructura Wison

Wison ¿
Lex {:
     /**
      Esto es un
      Comentario de bloque
    */

    # Declaración de terminales de la forma:
    # Terminal $_NOMBRE <- EXPRESIÓN ;
```

```
    Terminal $_Una_A     <- ‘a’ ;    # cualquier carácter alfanumérico
por separado
    Terminal $_Mas       <- ‘+’ ;    # cualquier carácter especial por
separado
    Terminal $_Punto     <- ‘.’ ;    # cualquier carácter especial por
separado
    Terminal $_P_Ab      <- ‘(’ ;    # cualquier carácter especial por
separado
    Terminal $_P_Ce      <- ‘)’ ;    # cualquier carácter especial por
separado
    Terminal $_FIN       <- ‘FIN’;   # cualquier palabra reservada
    Terminal $_Letra     <- [aA-zZ]; # alfabeto completo en mayusculas y
minusculas
    Terminal $_NUMERO    <- [0-9] ;  # Digitos del 0 al 9
    Terminal $_NUMEROS   <- [0-9]* ; # Estrella de Kleene para hacer 0 o
n veces
    Terminal $_NUMEROS_2 <- [0-9]+ ; # Cerradura positiva para hacer 1
on veces
    Terminal $_NUMEROS_3 <- [0-9]? ; # reconoce la cláusula ? para hacer
0 o 1 vez
    Terminal $_Decimal   <- ([0-9]*)($_Punto)($_NUMEROS_2); # terminal
combinado

:}


Syntax {{:
   # Declaración de no terminales de la forma
   # No_Terminal %_Nombre ;

   No_Terminal %_Prod_A;
   No_Terminal %_Prod_B;
   No_Terminal %_Prod_C;
   No_Terminal %_S;


   # Simbolo inicial de la forma
   # Initial_Sim %_Nombre ;

   Initial_Sim %_S ;

   #Todo símbolo no terminal debe ser declarado antes de usarse en las
producciones
   # Las producciones son de la siguiente forma
   # %_Initial_Sim  <= %_Prod_A ... %_No_terminal_N o $_Terminal_N ... ;
```

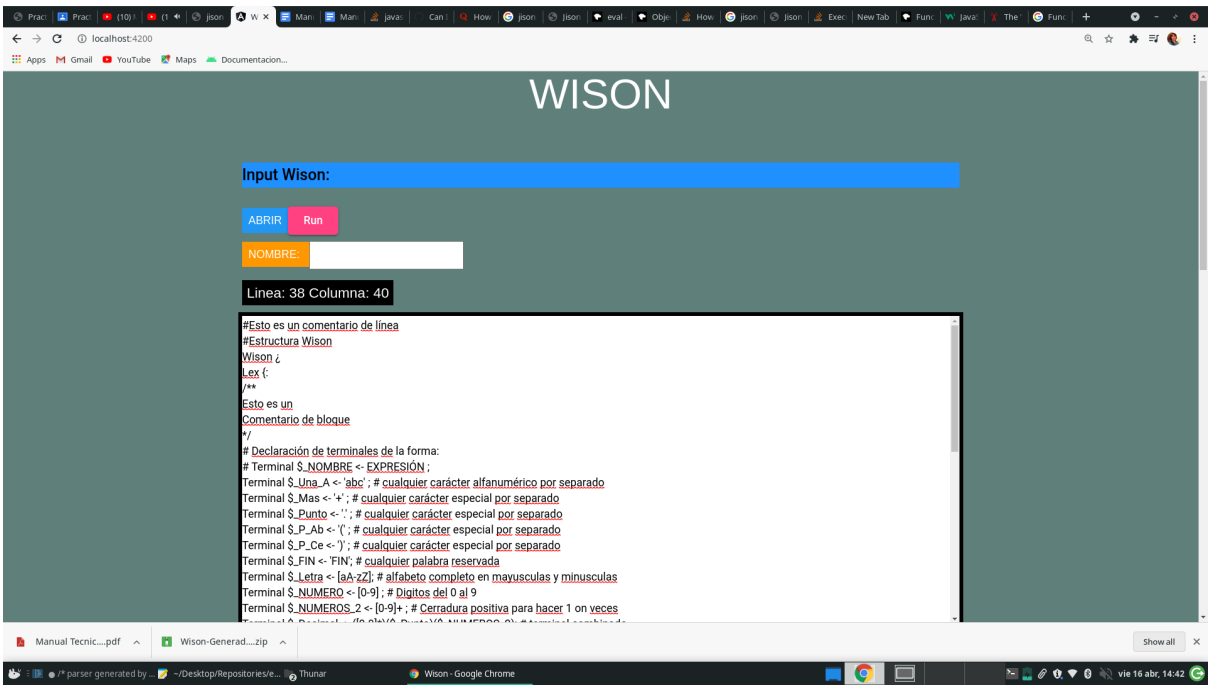
```
   %_S <= %_Prod_A $_FIN ;
   %_Prod_A <= $_P_Ab %_Prod_B $_P_Ce ;
   %_Prod_B <= %_Prod_B %_Prod_C | %_Prod_C ;
   %_Prod_C <= $_Una_A $_Mas $_Una_A ;
:}}
?Wison

# Fin de estructura Wison
```

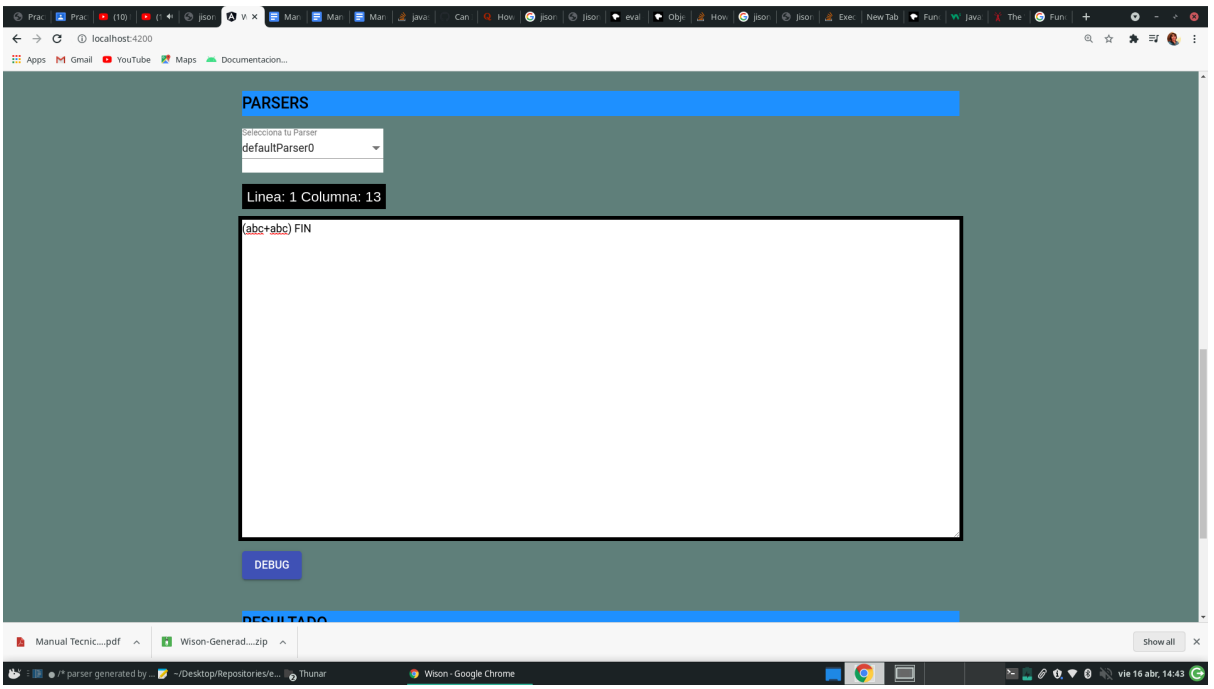
GRAMÁTICAS GENERADAS

Para las gramáticas generadas no se puede decir con exactitud la sintaxis ya que es el usuario quien las crea, por lo tanto se recomienda conocer la gramática que se escribió, y agregar una cadena de entrada al seleccionar una gramática, al seleccionarla y presionar el botón debug entonces procesa la gramática, y devuelve mensajes del resultado del parseo, si fue correcto generará un árbol de derivación.

Requisitos

- Angular Cli V11
- d3 graphviz
- Angular Material
- Node Js V14


localhost:4200
WISON
Input Wison:
ABRIR
Run
NOMBRE:
Linea: 38 Columna: 40
#Esto es un comentario de línea
#Estructura Wison
Wison ¿
Lex {:
/**
Esto es un
Comentario de bloque
*/
Declaración de terminales de la forma:
Terminal $_NOMBRE <- EXPRESIÓN ;
Terminal $_Una_A <- 'abc' ; # cualquier carácter alfanumérico por separado
Terminal $_Mas <- '+' ; # cualquier carácter especial por separado
Terminal $_Punto <- '.' ; # cualquier carácter especial por separado
Terminal $_P_Ab <- '(' ; # cualquier carácter especial por separado
Terminal $_P_Ce <- ')' ; # cualquier carácter especial por separado
Terminal $_FIN <- 'FIN'; # cualquier palabra reservada
Terminal $_Letra <- [aA-zZ]; # alfabeto completo en mayusculas y minusculas
Terminal $_NUMERO <- [0-9] ; # Digitos del 0 al 9
Terminal $_NUMEROS_2 <- [0-9]+ ; # Cerradura positiva para hacer 1 on veces
Manual Tecnic....pdf
Wison-Generad....zip
Show all
/* parser generated by ...
~/Desktop/Repositories/e...
Thunar
Wison - Google Chrome
vie 16 abr, 14:42


localhost:4200
PARSERS
Selecciona tu Parser
defaultParser0
Linea: 1 Columna: 13
(abc+abc) FIN
DEBUG
Manual Tecnic....pdf
Wison-Generad....zip
Show all
/* parser generated by ...
~/Desktop/Repositories/e...
Thunar
Wison - Google Chrome
vie 16 abr, 14:43