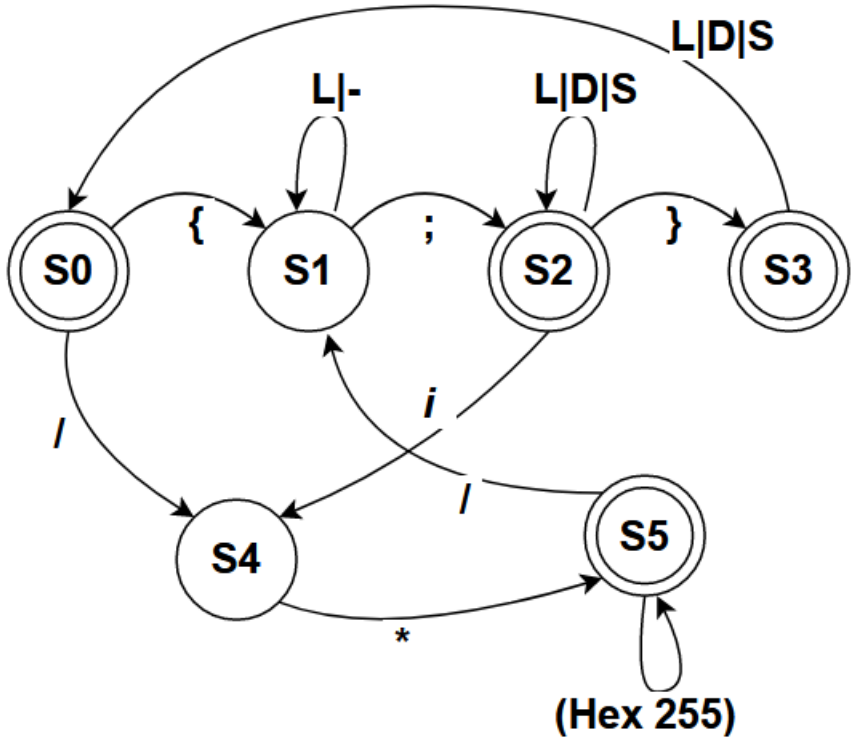
MANUAL TECNICO

Proyecto 1

Autómata css

Este autómata fue utilizado para resolver la gramática de css.
consta de 6 estados.




```
while i < len(entrada):
    cadena = entrada[i]
    if estado is 0:
        consola += 'entradando a estado 0 con lexema: '+lexema+'\n'
        if analizadorCSS.leerLetras(entrada[i]):
            estado =0
            lexema = lexema+ entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]

            columna += 1
        elif entrada[i].isdigit():
            estado = 0
            lexema = lexema + entrada[i]
            columna += 1
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        elif analizadorCSS.signoSelector(entrada[i]):
```

```
            estado =0
            lexema = lexema+ entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            columna += 1
        elif entrada[i] is '{':
            estado = 1
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            listaTokens.append(["Selector: " + lexema, fila, columna])
            consola += 'TOKEN ' + lexema+'= Selector: encontrado con estado 0\n'
            #analizadorCSS.textarea.tag_config('BOOL', foreground="blue")
            listaTokens.append(["Llave Abierta: " + '{', fila, columna])
            consola += 'TOKEN { = Llave Abierta: encontrado con estado 0\n'
            columna += 1
            lexema = ""
        elif entrada[i] is '/':
            estado = 4
            retorno ='A'
            columna += 1
            lexema = lexema + entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        elif entrada[i] == '\t' or entrada[i] ==' ':
            estado = 0
            columna += 1
            lexema = lexema + entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        elif entrada[i] =='\n':
            estado = 0
            fila += 1
            columna =0
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            #Error.append(["Salto de linea no esperado: " + entrada[i], fila,
columna])
        else:
            estado = 0
            fila += 1
            Error.append(["Caracter no valido: " + entrada[i], fila, columna])
            consola += 'ERROR Caracter no valido' + entrada[i] + 'encontrado EN
estado 0\n'

    elif estado is 1:
        consola += 'entradando a estado 1 con lexema: '+lexema+'\n'
        if analizadorCSS.leerLetras(entrada[i]):
            estado =1
            lexema = lexema+ entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            columna += 1
        elif entrada[i] is '-':
            estado = 1
            lexema = lexema + entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            columna += 1
        elif entrada[i] is ':':
            estado = 3;
            #lexema = lexema + entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
```

```
            for token in reservadas:
                if lexema == token:
                    listaTokens.append(["Propiedad: " + lexema, fila, columna])
                    consola += 'TOKEN ' + lexema + '= Propiedad: encontrado con
estado 1\n'
                    lexema = ""
                    estado = 2
            for token in reservadas:
                if lexema != token and lexema != '':
                    estado = 2
                    Error.append(["Propiedad inexistente: " + lexema, fila, columna])
                    consola += 'ERROR ' + lexema + '= Propiedad inexistente:
encontrado con estado 1\n'
                    lexema = ""
            columna += 1
        elif entrada[i] == '\t' or entrada[i] ==' ':
            columna += 1
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            #Error.append(["Espacio no esperado: " + entrada[i], fila, columna])
        elif entrada[i] =='\n':
            fila += 1
            columna =0
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            #Error.append(["Salto de linea no esperado: " + entrada[i], fila,
columna])

        else:
            estado = 1
            fila += 1
            Error.append(["Caracter no valido: " + entrada[i], fila, columna])
            consola += 'ERROR Caracter no valido' + entrada[i] + 'encontrado EN
estado 1\n'

    elif estado is 2:
        consola += 'entradando a estado 2 con lexema: '+lexema+'\n'
        if entrada[i] is '}':
            estado = 0
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            Entrada_Corregida = Entrada_Corregida + ''
            listaTokens.append(["Propiedades: " + lexema, fila, columna])
            consola += 'TOKEN ' + lexema + '= Propiedad: encontrado con estado 2\n'
            listaTokens.append(["Llave Cerrada: " + '}', fila, columna])
            consola += 'TOKEN } = Llave Cerrada: encontrado con estado 0\n'
            columna += 1
            lexema = ""
        elif analizadorCSS.leerLetras(entrada[i]):
            estado =2
            lexema = lexema+ entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            columna += 2
        elif entrada[i].isdigit():
            estado = 2
            lexema = lexema + entrada[i]
            columna += 1
            Entrada_Corregida = Entrada_Corregida + entrada[i]
```

```
        elif analizadorCSS.signoPropiedad(entrada[i]):
            estado = 2
            columna += 1
            lexema = lexema + entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        elif entrada[i] is ';':
            estado = 2
            retorno = 'B'
            lexema = lexema + entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            listaTokens.append(["Propiedades: " + lexema, fila, columna])
            consola += 'TOKEN ' + lexema + '= Propiedad: encontrado con estado 2\n'
            columna += 1
            lexema = ""
        elif entrada[i] is '/':
            estado = 3
            retorno = 'B'
            lexema = lexema + entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            listaTokens.append(["Propiedades: " + lexema, fila, columna])
            consola += 'TOKEN ' + lexema + '= Propiedad: encontrado con estado 2\n'
            columna += 1

        elif entrada[i] == '\t' or entrada[i] ==' ':
            columna += 1
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            #Error.append(["Espacio no esperado: " + entrada[i], fila, columna])
        elif entrada[i] =='\n':
            fila += 1
            columna = 0
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            #Error.append(["Salto de linea no esperado: " + entrada[i], fila,
columna])

        else:
            estado = 2
            fila += 1
            Error.append(["Caracter no valido: " + entrada[i], fila, columna])
            consola += 'ERROR Caracter no valido' + entrada[i] + 'encontrado EN
estado 1\n'

    elif estado is 3:
        consola += 'entradando a estado 3 con lexema: ' + lexema + '\n'

        if entrada[i] is '*':
            estado = 4
            columna += 1
            lexema = lexema + entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        elif entrada[i] == '\t' or entrada[i] ==' ':
            columna += 1
            estado = 3
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            lexema = lexema + entrada[i]
            #Error.append(["Espacio no esperado: " + entrada[i], fila, columna])
```

```
        elif entrada[i] =='\n':
            fila += 1
            columna = 0
            estado = 3
            lexema = lexema + entrada[i]
            #Entrada_Corregida = Entrada_Corregida + entrada[i]
            #Error.append(["Salto de linea no esperado: " + entrada[i], fila,
columna])

        else:
            estado = 2
            fila += 1

    elif estado is 4:
        consola += 'entradando a estado 4 con lexema: ' + lexema + '\n'
        if entrada[i] is '*':
            estado = 5
            columna += 1
            lexema = lexema + entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        elif entrada[i] == '\t' or entrada[i] == ' ':
            columna += 1
            #lexema = lexema + entrada[i]
            #Entrada_Corregida = Entrada_Corregida + entrada[i]
            # Error.append(["Espacio no esperado: " + entrada[i], fila, columna])
        elif entrada[i] == '\n':
            fila += 1
            columna = 0
            #lexema = lexema + entrada[i]
            #Entrada_Corregida = Entrada_Corregida + entrada[i]
            # Error.append(["Salto de linea no esperado: " + entrada[i], fila,
columna])

        else:
            estado = 5
            fila += 1

    elif estado is 5:
        consola += 'entradando a estado 5 con lexema: ' + lexema + '\n'
        if entrada[i] is '*':
            estado =6
            lexema = lexema + entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        elif entrada[i] == '\t' or entrada[i] == ' ':
            columna += 1
            lexema = lexema + entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            # Error.append(["Espacio no esperado: " + entrada[i], fila, columna])
        elif entrada[i] == '\n':
            fila += 1
            columna = 0
            lexema = lexema + entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            # Error.append(["Salto de linea no esperado: " + entrada[i], fila,
columna])
```

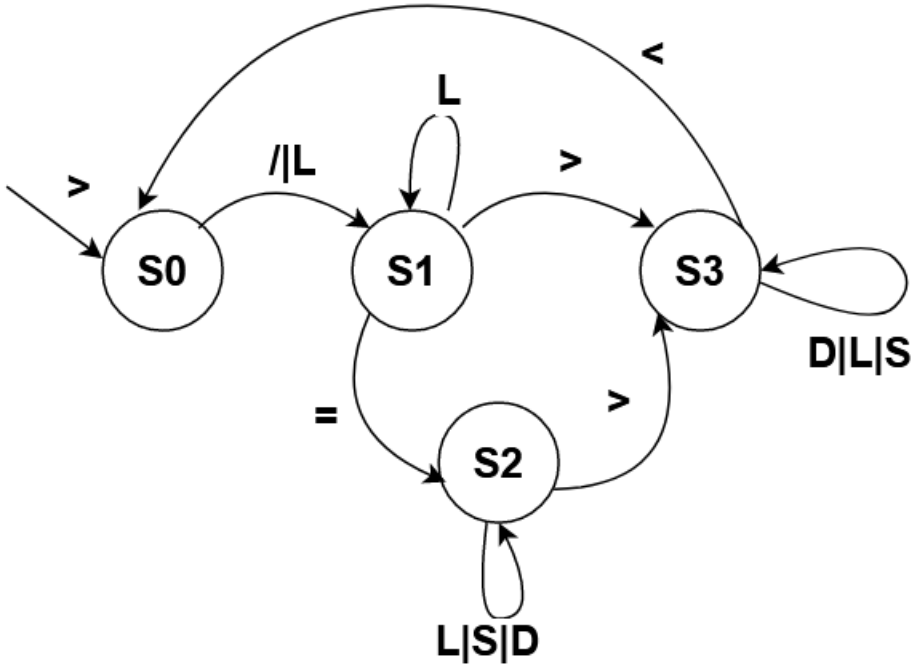
```
        else:
            estado = 5
            fila += 1
            lexema = lexema + entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]

    elif estado is 6:
        consola += 'entradando a estado 6 con lexema: ' + lexema + '\n'
        if entrada[i] is '/':
            if retorno is 'A':
                estado = 0
            elif retorno is 'B':
                estado = 1
            columna += 1
            lexema = lexema + entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            listaTokens.append(["Cometario: " + lexema, fila, columna])
            consola += 'Comentario ' + lexema + ': encontrado con estado 6\n'
            lexema = ""
        elif entrada[i] == '\t' or entrada[i] == ' ':
            columna += 1
            lexema = lexema + entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            # Error.append(["Espacio no esperado: " + entrada[i], fila, columna])
        elif entrada[i] == '\n':
            fila += 1
            columna = 0
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            lexema = lexema + entrada[i]
            # Error.append(["Salto de linea no esperado: " + entrada[i], fila,
columna])
        else:
            estado = 5
            fila += 1
            lexema = lexema + entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]

    i+=1
```

Autómata HTML

Este autómata fue creado para resolver la gramática de HTML consta de 4 estados




```
while i < len(entrada):

    cadena = entrada[i]
    if estado is 0:
        if entrada[i] is '<':
            estado = 0;
            lexema = lexema+ entrada[i]
            columna += 1
            Entrada_Corregida = Entrada_Corregida+entrada[i]

        elif analizador.leerLetras(entrada[i]):
            estado =1
            lexema = lexema+ entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            columna += 1
        elif entrada[i] is '/':
            estado = 1;
            lexema = lexema+ entrada[i]
            columna += 1
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        elif entrada[i] == '\t' or entrada[i] ==' ':
            estado = 0
            columna += 1
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        elif entrada[i] =='\n':
            estado = 0
            fila += 1
```

```
            columna =0
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        else:
            estado = 0
            fila += 1
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            Error.append(["Caracter no valido: "+entrada[i],fila,columna])

    elif estado  is 1:
        if analizador.leerLetras(entrada[i]):
            estado = 1;
            lexema = lexema+ entrada[i]
            columna += 1
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        elif entrada[i] is '=':
            estado = 2;
            lexema = lexema+ entrada[i]
            columna += 1
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        elif entrada[i] is '>':
            estado = 3;
            lexema = lexema+ entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            columna += 1
            #print(entrada[i])

            for token in reservadas:
                if lexema == token:
                    #print(entrada)
                    #print("Etiqueta: ",lexema," Fila: ",fila," Columna: ",columna)
                    listaTokens.append(["Etiqueta: "+lexema,fila,columna])
                    lexema = ""
                    estado = 3
            for token in reservadas:
                if lexema != token and lexema != '':
                    estado = 3
                    Error.append(["Etiqueta inexistente: " + lexema, fila, columna])
                    lexema = ""


        elif entrada[i] == '\t' or entrada[i] ==' ':
            estado = 0
            columna += 1
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        elif entrada[i] =='\n':
            estado = 0
            fila += 1
            columna += 1
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        else:
            estado = 1
            fila += 1
            Error.append(["Caracter no valido: " + entrada[i], fila, columna])

    elif estado is 2:
```

```
        if analizador.leerLetras(entrada[i]):
            estado = 2;
            lexema = lexema + entrada[i]
            columna += 1
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        elif entrada[i].isdigit():
            estado = 2
            lexema = lexema + entrada[i]
            columna += 1
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        elif analizador.signo(entrada[i]):
            estado = 2
            lexema = lexema + entrada[i]
            columna += 1
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        elif entrada[i] is '>':
            estado = 3;
            lexema = lexema + entrada[i]
            columna += 1
            #print("Instruccion Especial: ", lexema, " Fila: ", fila, " Columna: ",
columna)
            listaTokens.append(["Instruccion Especial: " + lexema, fila, columna])
            lexema = ""
            estado = 3
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        elif entrada[i] == '\t' or entrada[i] == ' ':
            estado = 2
            columna += 1
            lexema = lexema + entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        elif entrada[i] == '\n':
            estado = 2
            fila += 1
            columna =0
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        else:
            estado = 2
            fila += 1
            Error.append(["Caracter no valido: "+entrada[i],fila,columna])



    elif estado is 3:
        if analizador.leerLetras(entrada[i]):
            estado = 3;
            lexema = lexema + entrada[i]
            columna += 1
            Entrada_Corregida = Entrada_Corregida + entrada[i]

        elif entrada[i].isdigit():
            estado = 3
            lexema = lexema + entrada[i]
            columna += 1
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        elif entrada[i] == '.'or entrada[i] == ':'or entrada[i] == ';'or entrada[i]
```

```
== '='or entrada[i] == '/'or entrada[i] == '.'or entrada[i] == '-'or entrada[i] ==
'_'or entrada[i] == '\"':
            estado = 3
            columna += 1
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            lexema = lexema + entrada[i]
        elif entrada[i] is '<':
            estado = 0;
            if lexema != "":
                #print("Texto Plano: ", lexema, " Fila: ", fila, " Columna: ",
columna)
                listaTokens.append(["Texto Plano: " + lexema, fila, columna])
            lexema = ""
            lexema = lexema + entrada[i]
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        elif entrada[i] == '\t' or entrada[i] == ' ':
            estado = 3
            columna += 1
            Entrada_Corregida = Entrada_Corregida + entrada[i]
            #lexema = lexema + entrada[i]
        elif entrada[i] == '\n':
            estado = 3
            fila += 1
            columna =0
            Entrada_Corregida = Entrada_Corregida + entrada[i]
        else:
            estado = 3
            fila += 1
            Error.append(["Caracter no valido: "+entrada[i],fila,columna])

    i+=1
```

Autómata de calculadora

Para este analizador léxico se utilizaron, dos tipos de autómatas. autómata AFN y autómata a pila para resolver el análisis sintáctico.

Autómata AFN

```
char = entrada[i]
if estado is 0:
    if char == '(':
        estado = 0
        confirmacion.append([columna, fila, 'PARA', char])
        columna += 1
    elif char == ')':
        estado = 0
        confirmacion.append([columna, fila, 'PARC', char])
        columna+=1
    elif char == '-':
        estado = 0
        confirmacion.append([columna, fila, 'MEN', char])
        columna+=1
    elif char == '+':
        estado = 0
```

```
            confirmacion.append([columna, fila, 'MAS', char])
            columna+=1
        elif char == '/':
            estado = 0
            confirmacion.append([columna, fila, 'DIV', char])
            columna+=1
        elif char == '*':
            estado = 0
            confirmacion.append([columna, fila, 'POR', char])
            columna+=1
        elif char.isdigit():
            estado  = 1
            lexema+=char
        elif analizadorCalculadora.leerLetras(char):
            estado = 2
            lexema+=char
        elif char == '\t' or char ==' ':
            estado = 0
            columna+=1
        elif char == '\n':
            estado = 0
            expresion = ""
            for tokens in confirmacion:
                expresion += str(tokens[3])
            parser = Sintactico
            if len(confirmacion) != 0:
                parseoCorrecto = parser.parse(confirmacion)
                if parseoCorrecto:
                    print("Analisis Sintactico Correcto.")
                    token.append([idE,fila,expresion,"Cadena Correcta"])
                else:
                    token.append([idE,fila,expresion,"Cadena con Errores"])
            confirmacion = []
            id = 1
            fila += 1
            idE += 1
        #agregar parse aqui

        else:
            error.append([idE,lexema,fila, columna])
            idE+=1
    elif estado is 1:
        if char.isdigit():
            lexema+=char
            estado = 1
        elif char =='.':
            lexema+=char
            estado = 3
        else:
            confirmacion.append([columna, fila, 'NUMERIC', lexema])
            columna+=1
            estado = 0
            lexema=""
            i-=1
    elif estado is 2:
```

```
    if char.isdigit() or analizadorCalculadora.leerLetras(char):
        lexema+=char
        estado = 2
    else:
        confirmacion.append([columna, fila, 'ID', lexema])
        columna+=1
        estado = 0
        lexema=""
        i-=1
elif estado is 3:
    if char.isdigit():
        lexema += char
        estado = 1
    else:
        confirmacion.append([columna, fila, 'NUMERIC', lexema])
        columna += 1
        estado = 0
        lexema = ""
        i -= 1

i+= 1
```