

3 grupos de números enteros:

342#A42#123#523



UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Sintaxis y Semántica de los Lenguajes

Trabajo Práctico Autómatas

Curso K2102

Grupo N°6

Integrantes:

Agostina Escobar

Luciano Lobo

Lara Puelman

Valentina Aguilera

Luciano Degasperi

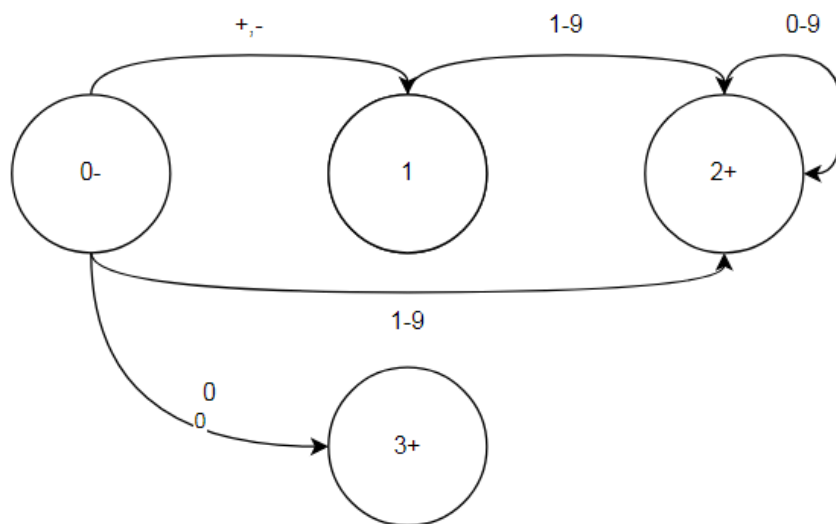
Fecha de Presentación: 07/10

Desarrollo de los puntos

Para el desarrollo del punto 1, en primera instancia propusimos la confección de 3 Lenguajes Regulares para integrarlos en un solo Lenguaje Regular que reconozca los tres tipos de números y el carácter '#' que los separa.

Las **convenciones** para los números decimales son:

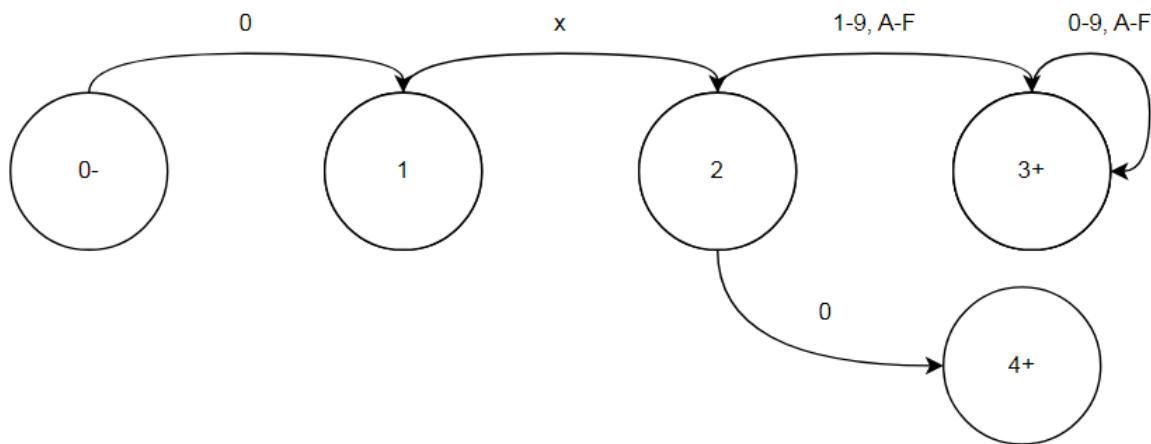
- Los números pueden tener signo o no para ser reconocidos como decimales. Por ejemplo, se reconocen: +12, -4, 67.
- El número cero se reconoce con un simple 0.
- No se reconoce la cadena en caso de repetición de ceros a la izquierda, como por ejemplo: 000.



El segundo reconoce números hexadecimales identificando con el prefijo '0x' comparable al método de identificación de hexadecimales en el lenguaje de programación C.

Las **convenciones** son:

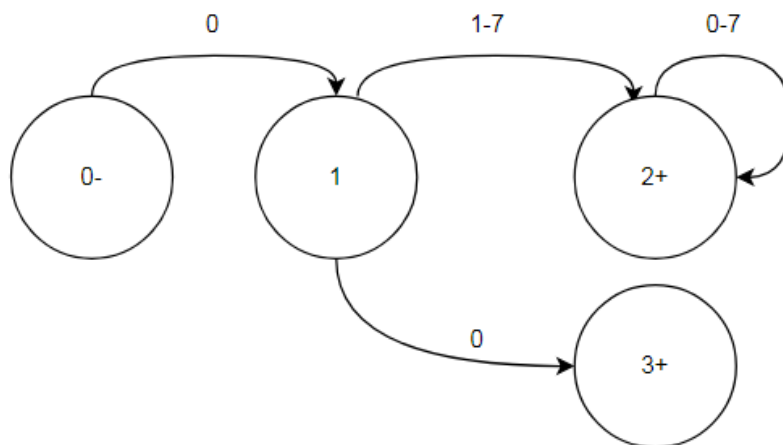
- Los números se reconocen como hexadecimales por empezar con la cadena '0x'
- El cero se reconoce con la cadena 0x0.
- No se reconoce la cadena en caso de repetición de ceros a la izquierda, como por ejemplo: 0x0000.



Por último, un lenguaje de números octales con el prefijo '0'.

Las **convenciones** son:

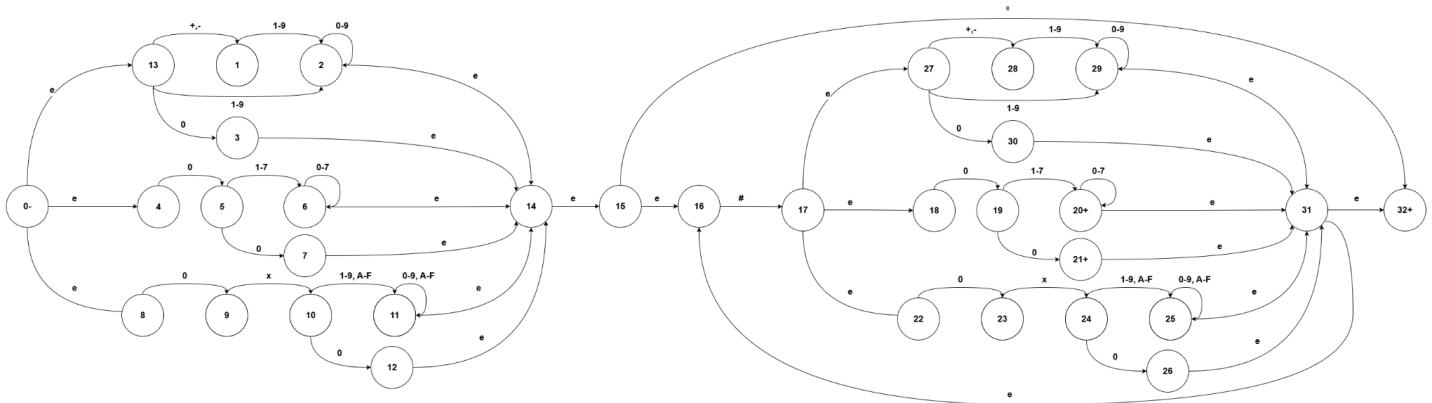
- Los números se reconocen como octales por empezar con el dígito '0'.
- El cero se reconoce con la cadena 00.
- No se reconoce la cadena en caso de repetición de ceros a la izquierda.



Para integrar estos tres lenguajes establecimos una expresión regular para el Lenguaje regular final.

ER : (OCTAL+DECIMAL+HEXADECIMAL).(#.(OCTAL+DECIMAL+HEXADECIMAL))*

Mediante el algoritmo de Thompson llegamos al autómata final con transiciones epsilon:



A continuación adjuntamos el archivo “Automatas - P1.xlsx” donde realizamos los pasos necesarios para llegar al AFD mínimo. El proceso resultó en la siguiente tabla de transiciones utilizada en el código:

Estados	-,+	0	x	#	1-7	8,9	A-F
0-	1	7+	8	8	2+	2+	8
1	8	8	8	8	2+	2+	8
2+	8	2+	8	0-	2+	2+	8
3+	8	8	8	0-	8	8	8
4	8	3+	8	8	6+	6+	6+
5+	8	5+	8	0-	5+	8	8
6+	8	6+	8	0-	6+	6+	6+
7+	8	3+	4	0-	5+	8	8
8	8	8	8	8	8	8	8

Luego, codificamos la función `es_palabra`, la cual recibe la cadena a analizar, junto a la tabla correspondiente a utilizar y la consigna seleccionada. El resultado de la función es 1 (Verdadero) si es palabra del lenguaje o 0 (Falso) si no lo es. Si es palabra del lenguaje, entonces ahí analiza la cantidad de palabras de cada tipo con la función `contar_tipos`.

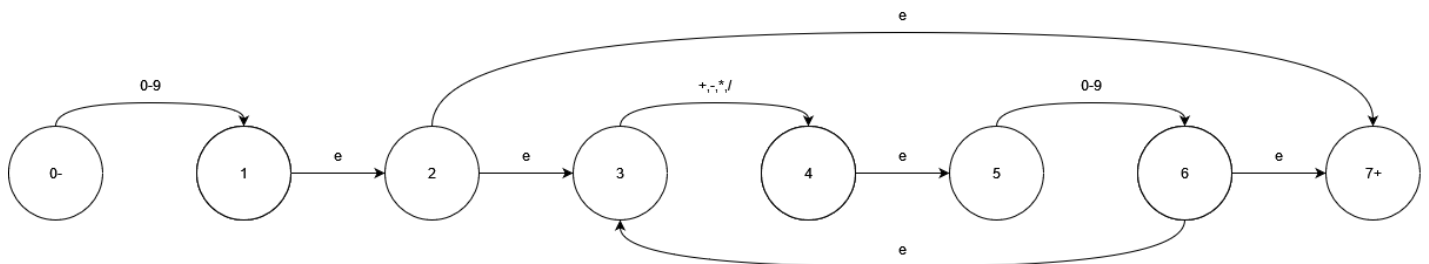
Para realizar la conversión solicitada en el punto 2, creamos una función llamada `caracterANumero`. Esta recibe un carácter numérico y devuelve su valor como un entero. y En un inicio, le pasamos por parámetro una lista de caracteres numéricos. Para ello toma el valor 1ASCII del carácter y le resta el valor ASCII del '0'.



```
1  int caracter_a_numero(char caracter)
2  {
3      return caracter - '0';
4  }
```

En el punto 3, como primer paso declaramos el autómata que reconoce la cadena de la operación. Vale la pena aclarar que la función solo admite operaciones con números de un sólo dígito.

$(0+1+2+3+4+5+6+7+8+9)((+|-|*|/)(0+1+2+3+4+5+6+7+8+9))^*$



Luego, aplicamos el algoritmo de clausura, y terminamos hallando el AFD mínimo. Adjuntamos el archivo “TP Automatas - Punto 3.xlsx” donde se explica el procedimiento a detalle para hallar la tabla de transiciones:

Estado	0-9	+, -, *, /
0-	1+	2
1+	2	0-
2	2	2

Reutilizamos la función `es_palabra` para verificar que la cadena denote una operación correcta. Con la cadena validada, procedemos a resolver la ecuación. Para resolver la ecuación y el problema de la precedencia de operadores, buscamos una función que

transforma una notación infija a una postfija¹. Con la cadena ya reorganizada, procedemos a resolver la ecuación final usando una pila. Para ello codificamos la función `evaluarPostfijo`, que retorna el valor final de la ecuación.

Instructivo y Pantallas de Funcionamiento

Al abrir el ejecutable, nos mostrará las siguientes opciones

```
+-----+
| TP AUTOMATAS - SYSL - GRUPO 6 |
+-----+

+-----CONSIGNAS-----+
| Que consigna del TP desea probar? |
| 1 - Tipos de numeros             |
| 2 - Operaciones                   |
+-----+

Ingrese la opcion correspondiente: |
```

Al seleccionar la consigna 1, nos pedirá la forma de ingreso de la cadena:

```
+-----+
| SELECCIONASTE LA CONSIGNA 1 |
+-----+

+-----CONSIGNA 1-----+
| De que forma ingresara la cadena? |
| 1 - Por linea de comando           |
| 2 - Por archivo                     |
+-----+

Ingrese la opcion correspondiente: |
```

En caso de que sea por archivo, se debe crear un archivo 'cadena.txt' que se encuentre en la misma carpeta que el ejecutable, y que tenga como contenido la cadena a verificar.

A continuación, un ejemplo de ejecución por línea de comando:

¹ <https://www.geeksforgeeks.org/dsa/convert-infix-expression-to-postfix-expression/>

```
Ingrese la cadena a continuacion: +123#0xABC#0777#-123

+-----RESULTADO-----+
| Cadena testada: +123#0xABC#0777#-123 |
| Resultado: Es palabra del lenguaje |
| Cantidad de decimales: 2 |
| Cantidad de octales: 1 |
| Cantidad de hexadecimales: 1 |
+-----+

Probar otra vez? 1 = SI | 0 = NO
```

En caso de elegir la consigna 2:

```
+-----+
| SELECCIONASTE LA CONSIGNA 2 |
+-----+

+-----INGRESO-----+
| De que forma ingresara la cadena? |
| 1 - Por linea de comando |
| 2 - Por archivo |
+-----+

Ingresa la opcion correspondiente: |
```

En caso de que sea por archivo, funciona de la misma manera que con la consigna 1: se debe crear un archivo 'cadena.txt' que se encuentre en la misma carpeta que el ejecutable, y que tenga como contenido la cadena a verificar. A continuación, un ejemplo:

```
Ingrese la opcion correspondiente: 2

+-----RESULTADO-----+
| Cadena testada: 1+2*4-7 |
| Resultado: Es palabra del lenguaje |
| Resultado de la operacion: 2 |
+-----+

Probar otra vez? 1 = SI | 0 = NO
```

A continuación, un ejemplo de ejecución por línea de comando:

```
Ingrese la cadena a continuacion: 1+2*3/4-5

+-----RESULTADO-----+
| Cadena testada: 1+2*3/4-5 |
| Resultado: Es palabra del lenguaje |
| Resultado de la operacion: -3 |
+-----+

Probar otra vez? 1 = SI | 0 = NO
```

A continuación, un ejemplo de ejecución donde la palabra no forma parte del lenguaje:

```
Ingrese la cadena a continuacion: 45#03538#12
```

```
+-----RESULTADO-----+  
| Cadena testeada: 45#03538#12 |  
| Resultado: No es palabra del lenguaje |  
+-----+  
  
Probar otra vez? 1 = SI | 0 = NO
```

Para salir del programa seleccionamos la opción “NO” del menú “Probar otra vez?”