Sintaxis y Semántica de los Lenguajes
Trabajo Práctico 2:
Autómatas

Alumnos: Facundo Bove Hernandez, Matias Valentin Cornalo, Santiago Perez, Santiago Invernizzi

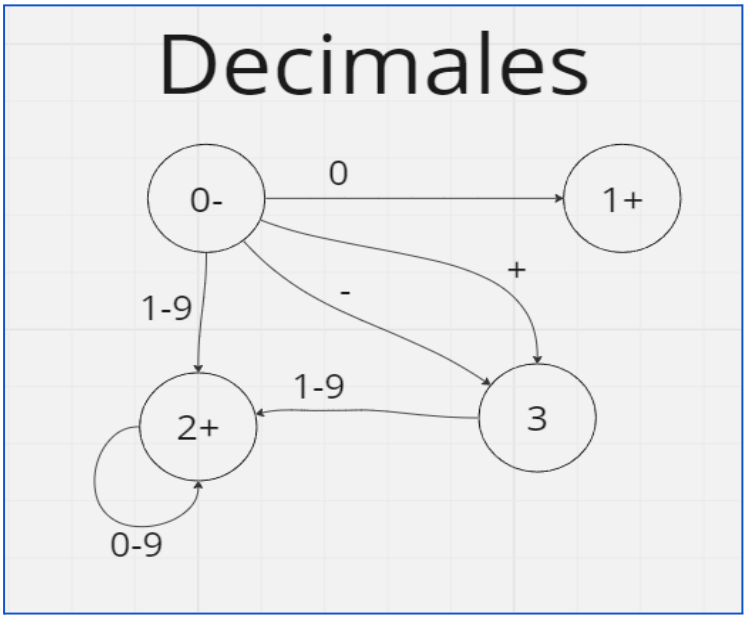
Profesora: Roxana Leituz

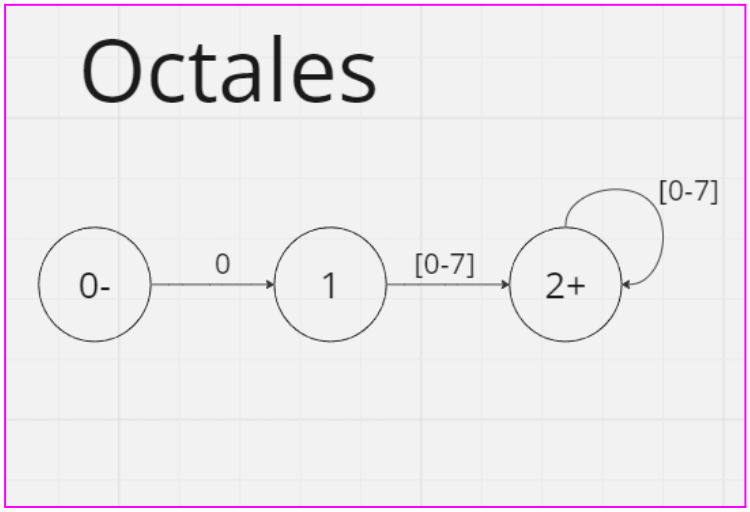
Año: 2023

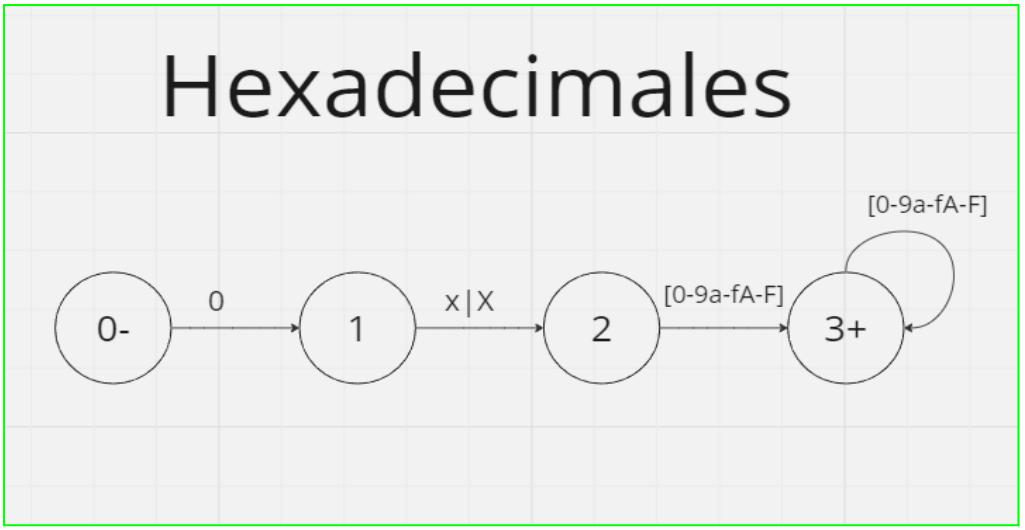
Comisión: K2055

Link del codigo fuente:
https://github.com/fbovehernandez/tp-sintaxis-automatas

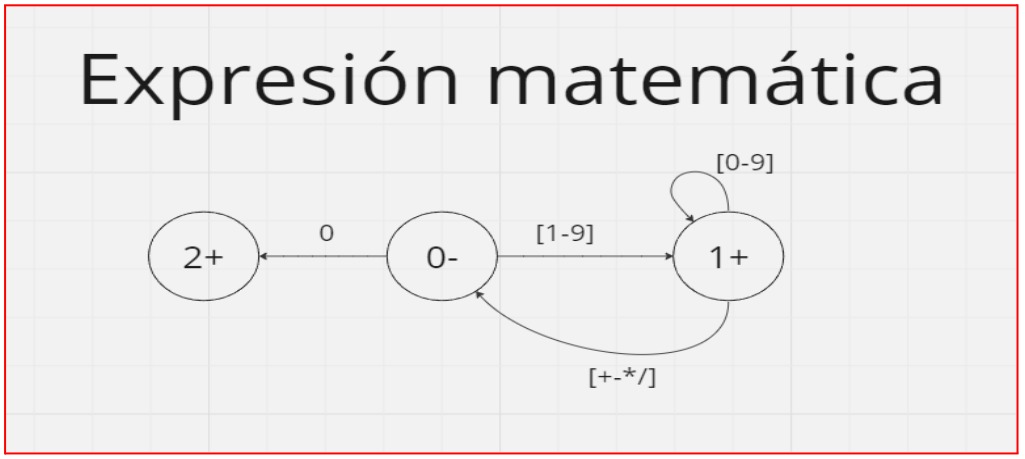
Autómatas







Este es un **autómata** que usamos para verificar las palabras del punto 3, es como un autómata reconocedor de decimales, pero extendido a los operadores que necesitábamos



Funciones Auxiliares Utilizadas

Explicación strtok:
Lo primero es entender que necesito definir 2 vectores de caracteres, uno para la palabra y otro para el carácter separador, además, es necesario un puntero auxiliar donde pueda guardar las diferentes partes de la palabra. Cuando encuentre el primer "carácter espurio/separador" va a quedarse con lo que había detrás de este (en el primer caso, se quedará con la palabra desde su comienzo hasta el carácter). Por ejemplo, si yo tengo la cadena "Quiero$separar$esto" comienza desde el primer elemento y guarda hasta el carácter. ¿Cómo hacemos para seguir guardando el resto de los caracteres? Bueno, llamamos a la misma función pero en vez de pasarle el vector de caracteres como la primera vez (junto con el char vector del separador), voy a pasarle NULL, y asi la funcion sabe que yo quiero que busque desde el último carácter que frenó la palabra. Podríamos asignarlo a diferentes char *vectorAux pero en este caso nos convenía pisarlo y volver a

ejecutar el while con la próxima palabra. Cada vez que ejecutamos el while preguntamos si la palabra es distinta del NULL, por ende va a saber cuando detenerse.

https://www.youtube.com/watch?v=nrO_pXGZc3Y

https://parzibyte.me/blog/2018/11/13separar-cadena-delimitadores-c-strtok/

En este video(primero) se muestra de manera muy buena como funciona esta función. Podríamos haber implementado tambien una pila, ir separando las palabras y después sacar eso y usarlo, pero esta función nos facilita mucho mas la separación de las mismas.

Explicación del funcionamiento del código y toma de decisiones:

Nuestro programa comienza con un menú, dando opciones al usuario para ejecutar las distintas funciones que corresponden a cada punto o ejercicio de este mismo trabajo práctico.
En la **primera función**, se le pide al usuario que ingrese una serie de caracteres (de ser posible numéricos) separados por un caracter en especifico, en este caso sera “$” y a partir de esta cadena la función determina la cantidad de números hexadecimales,octales y decimales ingresados. Para eso utilizamos diferentes autómatas, donde el principal cambio era la tabla de transición, modificando también las columnas y estados finales de ser necesarios. Se evalúa mientras la pila sea distinta de NULL, y chequea primero si la palabra es decimal, de no serlo, chequea que sea octal, y si no hexa, con contadores propios del lenguaje para saber la cantidad de palabras que reconoció de cada uno. En este punto también optamos por el uso de una función auxiliar que encontramos, el detalle de la misma está arriba. También es importante aclarar que no utilizamos una función que verifique el hecho de que se ingrese algo no valido, si no que lo enviamos a un estado de rechazo dentro del autómata, ya que nos pareció una mejor idea.

En la **segunda función,** se le pide al usuario que ingrese un carácter numérico para posteriormente devolver el valor del mismo, es decir, si se ingresa un ‘2’ se obtendrá un 2; si se ingresa un ‘3’, se obtendrá un 3. No es mucho más, pero la principal función que tiene se la damos en el 3er punto.

En la **tercera función** se le pide al usuario que ingrese una operación matemática en forma de cadena, como por ejemplo, “1+5*3+2” y a partir de esta se obtiene el resultado de esa misma expresión aritmética.

Primero verificamos (con otro autómata) si la palabra es válida. Esto implica entre otras cosas, que no empiece con un operador, que no tenga 2 (o más) operadores seguidos y que no termine con uno. Además, verificamos que no se ingrese algún valor que no corresponda a un dígito u operador usados.

Para ello utilizamos 4 pilas, 2 principales, y 2 auxiliares

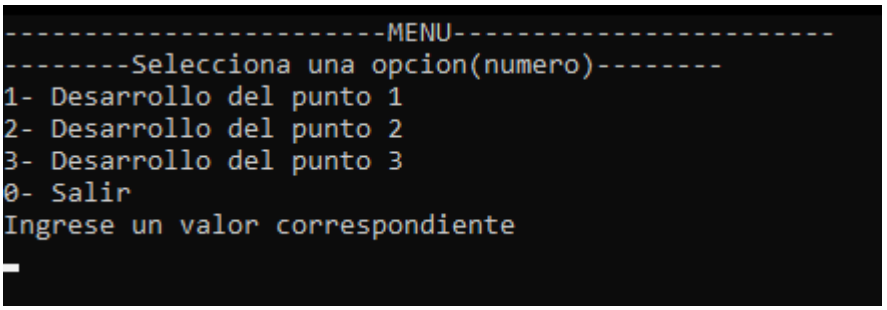
Cuando se ingresa la expresión por consola, utilizando la función del punto 2, convertimos los caracteres a números y los pusheamos en una pila. Acá es donde entra el uso de la primera pila auxiliar, ya que sino, la operación aritmética se realizaba a derecha, y no a

izquierda, problema que nos llevó a pensar en el uso de una pila auxiliar, que opera con todos los valores invertidos. De esta última se van sacando los elementos(pop), siempre primero se sacara un número, para posteriormente sacar un operador, en este momento se verifica que el operador sea de prioridad. Si lo es, se saca un segundo número de la pila (que siempre va a haber ya que por cómo ingresa la cadena sigue el formato num/op/num), se opera y se pushea en la misma pila, para poder mantener el formato mencionado previamente. Si se verificase que el operador no es de prioridad (+ o - en este caso) se hace el push en la otra pila principal, primero el número seguido del operador (no haría falta sacar el 2do numero, ya que no se opera nada). Una vez realizadas todas las operaciones de prioridad (* o /, no importa cual, tienen la misma prioridad) nos queda una segunda pila con los operadores de menor prioridad. Aca de nuevo invertimos la pila usando la segunda auxiliar, para evitar que se resuelva a derecha y no a izquierda. En este punto ya es ir sacando los números y operadores(pop) e ir operando, de la misma forma que antes. Se suman o restan los números y se vuelven a pushear, hasta sacar el último valor que será el output del punto 3.

El uso de float y no int fue para evitar problemas con la división, que de todas formas, siempre trunca el valor, pero si no eran floats, el programa fallaba.

Capturas de pantalla

-> Menú



-> Reconocedor (Punto 1)

```
1- Desarrollo del punto 1
2- Desarrollo del punto 2
3- Desarrollo del punto 3
0- Salir
Ingrese un valor correspondiente
1
Ingrese cadena separada por $
0$0x04$056$+203$0xK
0.
Es palabra decimal
0x04.
Es palabra Hexa
056.
Es palabra Octal
+203.
Es palabra decimal
0xK.
No pertenece al lenguaje
cantidad de caracteres decimales es 2.
cantidad de caracteres octales 1.
cantidad de caracteres hexadecimales es 1.
-----------------------MENU-----------------------
```

-> Transforma un caracter numerico a su valor numérico(Punto 2)

```
Ingrese un valor correspondiente
2
Ingrese un caracter numerico (1-9)
5
El valor numerico del caracter es: 5.
```

-> Resolución de expresión (Punto 3)

```
Ingrese un valor correspondiente
3
Ingrese una cadena de digitos y operaciones
5+3+9/9*5+1
Resultado 14.000000.
```