Documento de Especificación Técnica de Automata_Scala

El programa descrito implementa un Autómata Finito Determinista. Este programa verifica si las palabras de entrada pertenecen o no al lenguaje que representa este Autómata. Para ello, se elaboró un programa en Scala llamado “Automata_Scala” que le permite al usuario crear un autómata AFD a partir de un archivo de texto y una vez creado, el usuario puede insertar las palabras que desee para revisarlas hasta que el usuario decida ponerle fin. Por otro lado, cabe mencionar qué Scala es un lenguaje compilado, es decir, el código escrito se convierte en bytecode de java antes de ejecutarse en la JVM. Se puede escribir y ejecutar código scala en diferentes entornos de desarrollo integrado (IDE), en este caso se utilizó Intellij IDEA como IDE, el código se puede ejecutar con la ayuda de un plugin de Scala. No obstante hay otros IDES como Visual Studio Code con el qué también se puede ejecutar código con un plugin.

A continuación se describen las clases y módulos qué componen el programa.

Automata.scala

Esta clase permite crear instancias de la clase Autómata, las instancias que se creen de esta clase serán nuestros autómatas. Para poder realizar la clase Automata se requirió de varias estructuras de datos para almacenar la información de esta. En este caso, la clase debe tener como parámetros a las transiciones así como el estado inicial y estado finales. Para almacenar las transiciones se usó un Array[Transition] que guarda aquellos objetos de tipo Transition (véase en Transition.Scala); para el estado inicial se terminó usando una variable tipo Char y para los estados finales se empleó un Array[Char] que guarda valores de tipo Char.

Los métodos que acompañan a esta clase son el método pivote(*param..*) el cual recibe el estado actual y el carácter de entrada de la cadena a revisar (ambos de tipo Char) y examina a estos dos valores con la tabla de transiciones para ver hacia dónde se transiciona; al final de esta revisión, regresa un valor de tipo Char que representa el estado destino.

También, la clase tiene el método isFinalState(*param..*) el cual recibe el estado actual de la palabra una vez que se haya revisado toda la cadena, de forma que pueda verificar si este este estado es un estado final, regresando Verdadero si es final o Falso si no lo es.

Transition.scala

Esta clase permite crear instancias que representan las transiciones de nuestro autómata, los cuales más tarde se almacenarán en un Array para formar la tabla de transiciones. Su constructor está conformado por tres parámetros de tipo Char los cuales son el estado inicial, el carácter de entrada y el estado final en ese orden. Igualmente, esta clase tiene un método de impresión de la transición llamado toString().

Constants.scala

Este objeto contiene la ruta de acceso del archivo de texto que contiene toda la información necesaria para crear el autómata. Es un objeto cuya finalidad es almacenar las constantes globales.

Operations.scala

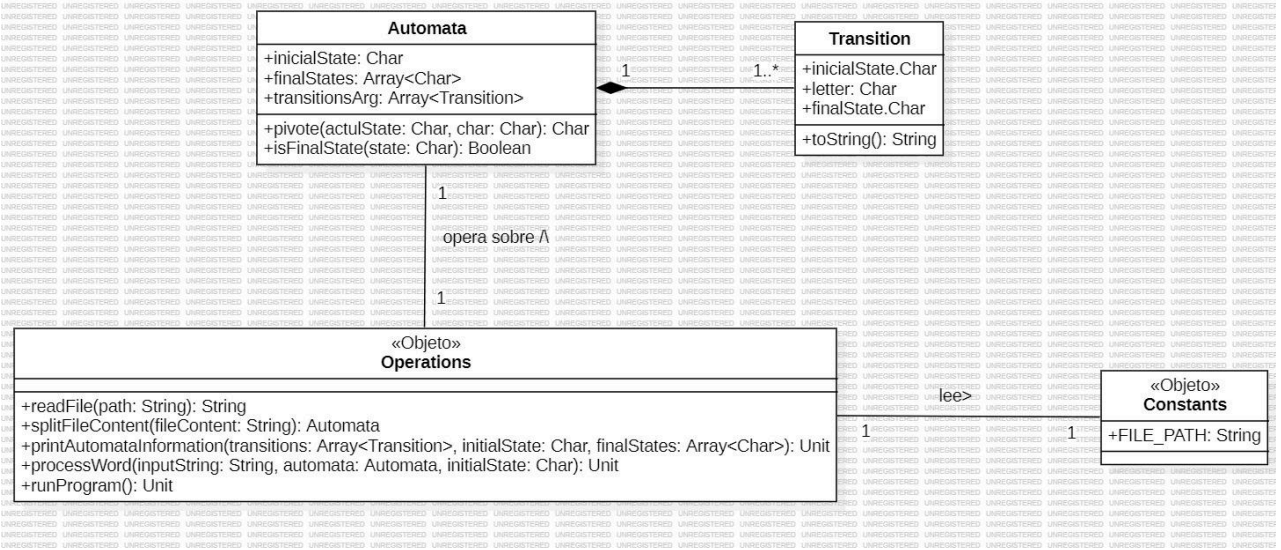
Es un objeto singleton qué encapsula lógica qué no está asociada a ninguna instancia de clase. Contiene métodos qué realizan operaciones necesarias para el programa. Tiene una función readFile qué recibe una dirección de un archivo, lo lee y regresa el contenido de ese archivo en un String, esta función es para leer el archivo con las reglas y transiciones qué definen a un autómata. También tiene una función qué separa el contenido del archivo en transición, estado inicial y estados finales. Estos datos los guarda en estructuras de datos, por ejemplo, las transiciones las guarda en un array de objetos Transition; los estados finales en un array de carácteres y el estado inicial es un tipo de dato carácter. Operations.scala tiene una función qué ejecuta el flujo del programa (runProgram), esta función lee el contenido del archivo, los guarda en estructuras de datos con la función splitFileContent, imprime estas estructuras para el usuario con la función printAutomataInformation y lee el las cadenas qué el usuario ingresa hasta qué el usuario ingrese “fin”, las

cadenas qué ingresa, las envía una por una a una función qué valida si pertenece al lenguaje (processWord)

Principal.scala

Es un objeto con el método main, este tiene el objetivo de ser el punto de entrada al programa, en otras palabras, llama al método “runProgram” de Operations.scala para iniciar el flujo.

Diagrama de clases



Manual de Usuario

Antes de ejecutar el programa primero se debe crear un archivo de texto para crear el autómata. El archivo de texto debe tener la siguiente estructura:

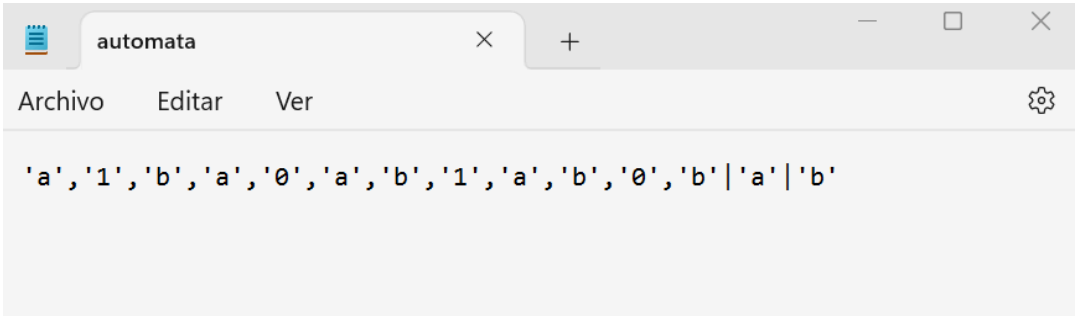
<transiciones>|<estado_inicial>|<estados_finales>

Para la sección de las transiciones se debe seguir un secuencia para cada transición con el estado origen, el carácter de entrada y el estado destino en ese orden, separados por comas, esto se repite las veces que se requiera hasta representar la tabla de transiciones por completo, tomando en cuenta que cada uno de sus elementos debe ir en comillas simples.

Para la sección del estado inicial solo se debe poner un carácter entre comillas simples, el cual representará el estado inicial del autómata.

Para la sección de los estados finales se deben poner todos los estados finales que va a tener el autómata, con comillas simples y separados por coma.

Tome en cuenta este archivo como ejemplo:

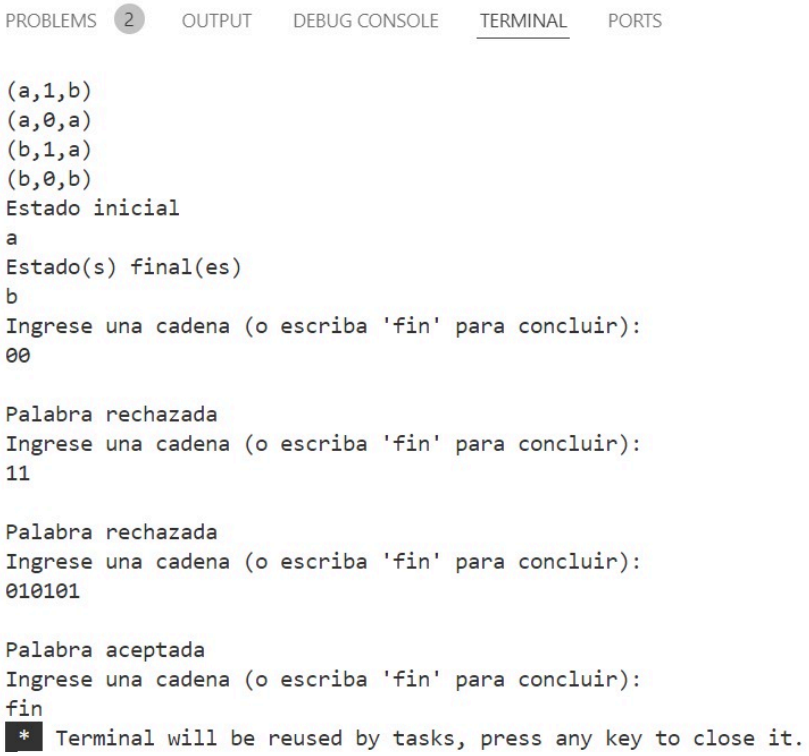


Este documento tiene un autómata que acepta las palabras en {0,1} con número impar de 1's.

Ahora que ya tiene su autómata en un archivo de texto tendrá que copiar la ruta de acceso del archivo y ponerlo en FILE_PATH ubicado en el objeto Constants

```
object Constants {
  val FILE_PATH="src/main/resources/automata.txt"
}
```

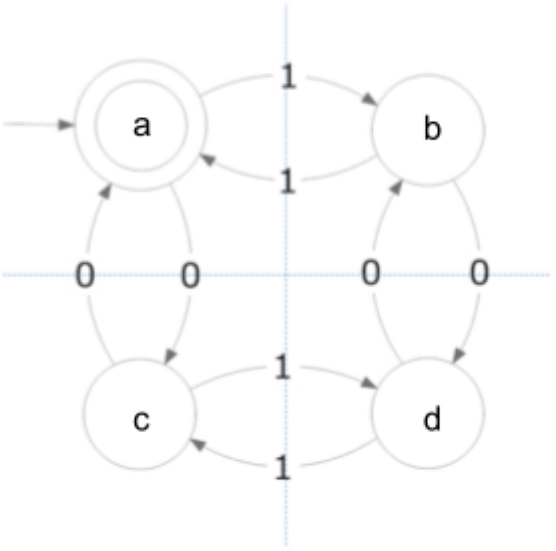
Por último, se debe de ejecutar el programa, debería quedar de la siguiente forma:



Casos de prueba.

Primer caso.

Autómata finito determinista qué reconoce el lenguaje conformado por cadenas con un número par de ceros y un número par de unos.



Descripción
Alfabeto: {1, 0}
Estados: a, b, c, d
Estado inicial: a
Estados finales: a

Estado	Alfabeto	Estado destino
a	1	b
a	0	c
b	1	a
b	0	d
c	1	d
c	0	a
d	1	c
d	0	b

Texto

Archivo Editar Ver

```
'a','1','b','a','0','c','b','1','a','b','0','d','c','1','d','c','0','a','d','1','c','d','0','b'|'a'|'a'
```

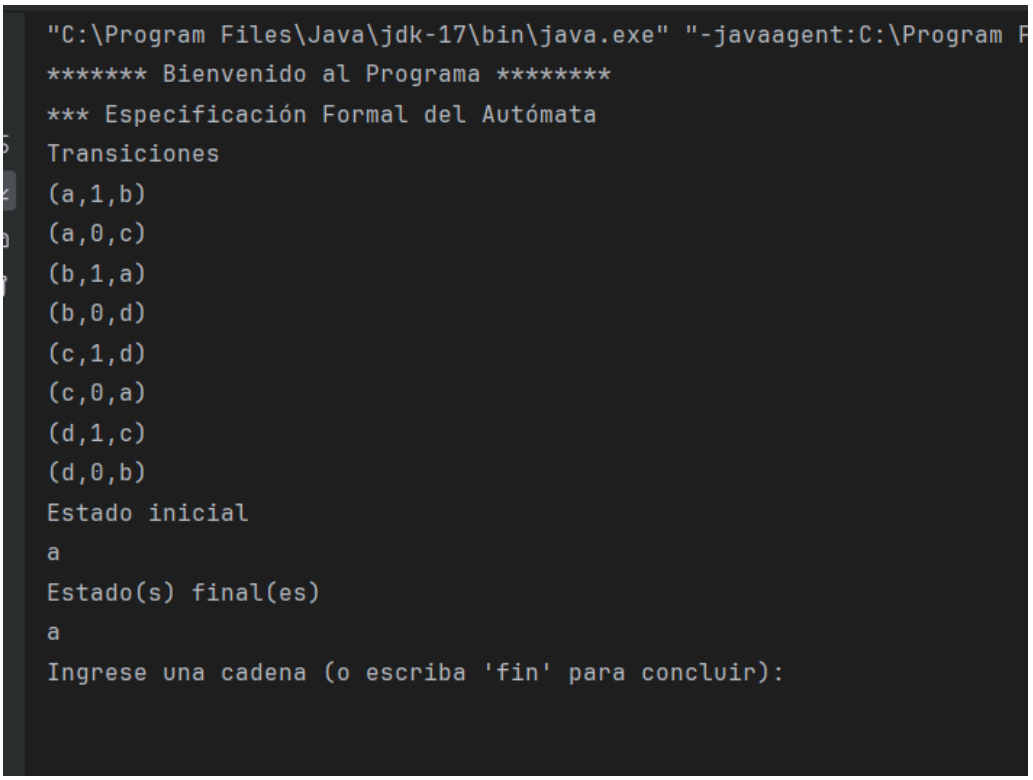
Palabras aceptadas

- "" (cadena vacía)
- "0011"
- "1100"
- 11001100
- 11001010

Palabras rechazadas

- "0"
- "1"
- "001"
- "011"

EJECUCIÓN.



Palabras aceptadas.

```
Ingrese una cadena (o escriba 'fin' para concluir):


Palabra aceptada
Ingrese una cadena (o escriba 'fin' para concluir):
0011

Palabra aceptada
Ingrese una cadena (o escriba 'fin' para concluir):
1100

Palabra aceptada
Ingrese una cadena (o escriba 'fin' para concluir):
11001100

Palabra aceptada
Ingrese una cadena (o escriba 'fin' para concluir):
11001010

Palabra aceptada
```

Palabras rechazadas.

```
Ingrese una cadena (o escriba 'fin' para concluir):
0

Palabra rechazada
Ingrese una cadena (o escriba 'fin' para concluir):
1

Palabra rechazada
Ingrese una cadena (o escriba 'fin' para concluir):
001

Palabra rechazada
Ingrese una cadena (o escriba 'fin' para concluir):
011

Palabra rechazada
Ingrese una cadena (o escriba 'fin' para concluir):
```

Segundo caso de prueba.

Autómata Finito Determinista (AFD) que reconoce cadenas que no contienen a la subcadena “01”

Descripción
Alfabeto: {1, 0}
Estados: {a,b,c}
Estado inicial: a
Estados finales: {a, b}

Estado	Alfabeto	Estado destino
a	1	a
a	0	b
b	1	c
b	0	b
c	1	c
c	0	c

Palabras aceptadas

- 1
- 11111110
- 110
- 00
- 11
- 11110

Palabras rechazadas

- 01
- 101010
- 010101
- 0000001
- 10000001

Archivo TXT

Archivo Editar Ver

```
'a','1','a','a','0','b','b','1','c','b','0','b','c','1','c','c','0','c'|'a'|'a','b'
```

EJECUCIÓN.

```
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program F
******* Bienvenido al Programa ********
*** Especificación Formal del Autómata
Transiciones
(a,1,a)
(a,0,b)
(b,1,c)
(b,0,b)
(c,1,c)
(c,0,c)
Estado inicial
a
Estado(s) final(es)
a
b
Ingrese una cadena (o escriba 'fin' para concluir):
```

Palabras aceptadas.

```
Ingrese una cadena (o escriba 'fin' para concluir):
1

Palabra aceptada
Ingrese una cadena (o escriba 'fin' para concluir):
11111110

Palabra aceptada
Ingrese una cadena (o escriba 'fin' para concluir):
110

Palabra aceptada
Ingrese una cadena (o escriba 'fin' para concluir):
00

Palabra aceptada
Ingrese una cadena (o escriba 'fin' para concluir):
11

Palabra aceptada
Ingrese una cadena (o escriba 'fin' para concluir):
11110

Palabra aceptada
```

Palabras rechazadas.

```
Ingrese una cadena (o escriba 'fin' para concluir):
01

Palabra rechazada
Ingrese una cadena (o escriba 'fin' para concluir):
101010

Palabra rechazada
Ingrese una cadena (o escriba 'fin' para concluir):
010101

Palabra rechazada
Ingrese una cadena (o escriba 'fin' para concluir):
0000001

Palabra rechazada
Ingrese una cadena (o escriba 'fin' para concluir):
10000001

Palabra rechazada
```