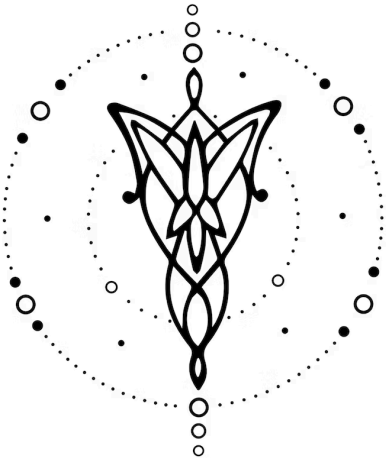

Especificación

Diseño e Implementación de un Lenguaje

v1.0.0

1. Equipo

Nombre	Apellido	Legajo	E-mail
Sol	Rodriguez	63029	solrodriguez@itba.edu.ar
Maria Agustina	Sanguinetti	63115	msanguinetti@itba.edu.ar
Uriel Ángel	Arias	63504	uarias@itba.edu.ar

2. Repositorio

La solución y su documentación serán versionadas en: AutomaTeX.

3. Dominio

Desarrollar un lenguaje de dominio específico (DSL) con tipado estático que permita la creación y simulación de autómatas finitos, de manera tal que una vez compilado un archivo escrito en este lenguaje se devuelva una representación equivalente de los autómatas definidos por medio de un documento en formato LaTeX, el cual contenga su respectivo diagrama de estados y tabla de transiciones.

Los usuarios podrán definir el autómata, especificando los distintos estados y sus tipos, el estado inicial y aquellos que sean finales, determinar las transiciones entre ellos, y establecer el alfabeto que el mismo reconocerá.

Cabe destacar que si bien se reconoce que se podría manejar la definición de autómatas sin especificar su tipo dadas las equivalencias probadas entre los mismos y la existencia de algoritmos para hacer la conversión de uno a otro, se hace uso de tipos de autómatas para reducir la complejidad del proyecto y aprovechar el uso de errores de compilación para marcar errores conceptuales propios de usuarios principiantes con los conceptos teóricos. Se trata entonces de un aspecto sobre el cual el proyecto podría seguir escalando para hacer que su uso sea menos restrictivo y permita la reutilización de los autómatas ya definidos.

Por otro lado, junto con la definición de autómatas se hace necesario el manejo de conjuntos que bien podrían reutilizarse para autómatas con propiedades similares, por ejemplo, al compartir los mismos estados. Es por esto que se provee de tipos de datos que si bien refieren a las propiedades de los autómatas, como estados, transiciones y un alfabeto, programáticamente se distinguen por usar notación y operaciones binarias básicas de conjuntos, entre las cuales se encuentran la unión, la intersección y la diferencia.

La implementación exitosa de este lenguaje proporcionará una herramienta poderosa para desarrolladores de sistemas basados en autómatas finitos, permitiéndoles modelar y verificar la correctitud de los autómatas definidos de una forma clara y fácil de utilizar para que el usuario no pierda tiempo en recordar la sintaxis de LaTeX.

4. Construcciones

El lenguaje desarrollado debería ofrecer las siguientes construcciones, prestaciones y funcionalidades:

(I). Se podrán crear autómatas de tipo DFA, NFA y LNFA (Lambda NFA), y estos podrán ser identificados a través de un nombre o etiqueta.
(II). Se podrán definir estados, transiciones y un alfabeto dentro de cada autómata.
(III). Se podrán crear estados, transiciones y alfabetos independientemente de los autómatas, es decir, como constantes globales.

(IV). Las constantes globales podrán ser de tipo *states* para los estados, *transitions* para las transiciones, *alphabet* para el alfabeto y para todas ellas la letra inicial del nombre u identificador deberá estar en mayúscula.

(V). Para alterar el valor de una variable se deberá indicar su nombre y seguido de dos puntos (":") se deberá indicar el valor que se desea que tenga. Para ello se pueden emplear operadores, los cuales se explicarán más adelante, y, por ende, otras variables o constantes del mismo tipo. También se podrá llamar a sí misma. Por ejemplo, sea A una variable de states a la cual se desea agregar un estado *p*, entonces se la puede redefinir de la siguiente manera: A: A + {*p*}.

(VI). Se podrán definir transiciones utilizando una sintaxis de la forma |*p*|-*a*->|*q*|, |*p*|<-*a*-|*q*| o *p*<-*a*->*q*, siendo *p* y *q* dos estados o conjuntos de estados y *a* un símbolo o conjunto de símbolos del alfabeto definido para un autómata, donde en el primer caso se define una transición del estado *p* a *q* consumiendo *a*, en el segundo caso se define una transición del estado *q* a *p* consumiendo *a*, y en el último caso se definen ambas transiciones mencionadas anteriormente. Si se quiere hacer referencia a más de 2 estados o símbolos, se podrán listar dichos estados o símbolos del alfabeto entre paréntesis, tal como se ve en el siguiente ejemplo: |{*q*0, *q*1}|-{*a*,*b*}->|*q*2|, que indica que los estados *q*0 y *q*1 al consumir *a* o *b* llegan al estado *q*2, es decir, que en una única sentencia se representan 4 transiciones distintas.

(VII). Los conjuntos serán definidos mediante llaves ("{", "}") y sus elementos se deberán separar con comas. No obstante, también se podrá definir un conjunto de un único elemento sin llaves, de manera tal que se asumirá que el conjunto solo tiene un elemento.

(VIII). Se empleará el símbolo @ para representar lambda.

(IX). Se empleará *EMPTY* como palabra reservada para representar el conjunto vacío. Otra manera de representarlo es mediante el uso de {}, pues al no contener elementos se entenderá que se trata del conjunto vacío.

(X). Los autómatas podrán ser de tipo *DFA*, *NFA* y *LNFA*. Los mismos se definen de la forma *tipoAutomata NombrAutomata: []*

(XI). Para especificar los estados, alfabeto y transiciones de un autómata se utilizan *states*, *alphabet* y *transitions* respectivamente. Es de carácter obligatorio definir los mismos dentro de un autómata para hacer posible su creación.

(XII). Dentro del conjunto *states* definido en la creación del autómata se deberán definir los estados "regulares" (no iniciales ni finales), finales y el inicial obligatoriamente.

(XIII). Para ello, se deberán colocar los estados que participarán en dicho conjunto y un símbolo para representar su tipo. En el caso del estado inicial, siendo q0 un estado, el símbolo a usar es >*q*0, mientras que para los estados finales se emplea **q*0.

(XIV). Una vez creado el autómata, se podrán acceder sus estados, transiciones y alfabeto usando el nombre del autómata seguido de un punto y el nombre de dicho conjunto. Por ejemplo, para acceder a los estados de un autómata se deberá escribir lo siguiente: *NombreAutomata.states*, donde se tendrá un conjunto formado por 3 subconjuntos, uno de estados regulares, uno con el estado inicial y uno con los estados finales. Para acceder a sólo uno de estos subconjuntos, se deberá especificar el requerido de la siguiente forma NombreAutomata.states.tipo donde tipo podrá ser reemplazado por las palabras reservadas *initial*, *final* o *regular*.

(XV). Se podrán emplear operadores para realizar operaciones entre conjuntos del mismo tipo. Los operadores provistos son: el "+" para la unión, el "^" para la intersección y el "-" para

la diferencia. De esta manera, se podrá utilizar cualquiera de estos operadores tanto para constantes, como para conjuntos no vinculados a un nombre o etiqueta. Será válida una expresión de la forma A * B, siendo A y B dos constantes que son del mismo tipo y "*" un operador dentro del conjunto {+, -, ^}, y de la forma A * {*a*, *b*, *c*} donde A es de tipo *alphabet* o *states*, dependiendo de si {a, b, c} son elementos del alfabeto o estados. Cabe destacar que estos operadores tienen la misma precedencia. Si se desea indicar cierto orden a la hora de resolver un conjunto de operaciones, se deben usar paréntesis, sino por defecto la lectura se llevará a cabo de izquierda a derecha.

(XVI). Los autómatas se mostrarán en el documento LaTeX.

5. Casos de Prueba

Se proponen los siguientes casos iniciales de prueba de **aceptación**:

(I). Un autómata donde se tenga un estado que es tanto final como inicial y al menos un estado no final ni inicial.

(II). Un autómata del tipo DFA donde no se definan todas las transiciones, de manera tal que en la salida se incluyan dichas transiciones y flechas faltantes que lleguen hacia un nuevo estado trampa.

(III). Un autómata que tenga 2 estados finales.

(IV). Un autómata de tipo LNFA donde se tengan 3 transiciones lambda.

(V). Un autómata de tipo NFA en donde se tenga 2 transiciones que partiendo del mismo estado y consumiendo el mismo símbolo llegue a 3 estados diferentes.

(VI). Un autómata DFA donde se tenga al menos una transición que sale de un estado y llega al mismo estado, es decir, un estado donde su diagrama tenga un lazo.

(VII). Un autómata de tipo DFA donde un par de estados *p* y *q* estén presentes en transiciones de modo tal que al consumir el mismo símbolo del alfabeto, llámese *a*, desde *p* se llegue a *q* y viceversa. Es decir, en los cuales se defina una transición del tipo *|p|<-a->|q|*.

(VIII). Un conjunto de estados como constante donde se tengan 4 estados: uno inicial, 2 finales y uno regular. Luego que se referencie al definir dos autómatas distintos, uno DFA y otro NFA.

(IX). Un autómata LNFA o NFA cuyo conjunto de transiciones contenga únicamente una resta entre un conjunto de transiciones y otro conjunto idéntico definidos previamente.

(X). Un autómata donde el conjunto de estados se forme a partir de la intersección del conjunto de estados de dos autómatas distintos previamente definidos.

(XI). Un autómata donde su alfabeto sea la unión de los alfabetos de dos autómatas distintos previamente definidos.

(XII). Un autómata donde su conjunto de estados sea resultado de aplicar la diferencia simétrica entre otros dos conjuntos de estados usados para dos autómatas definidos previamente.

Además, los siguientes casos de prueba de **rechazo**:

(I). Un autómata definido como un DFA con un estado que consuma un mismo símbolo del alfabeto para transicionar a más de un estado.
(II). Un autómata definido como un DFA o NFA que tenga una transición en donde no se consuma ningún símbolo del alfabeto.
(III). Un autómata que no contenga un estado inicial o contenga más de uno.
(IV). Un autómata que no tenga estados finales.
(V). Un autómata que no tenga estados o alfabeto.
(VI). Un conjunto al que no se le coloca el "}" para terminar su definición.
(VII). Una operación donde los valores con los que se opera no son del mismo tipo.
(VIII). Una operación donde existan más "(" que ")" o viceversa.

6. Ejemplos

Ejemplo 1: Creación de un autómata AFD, al cual se lo nombra *Automata1*, con los estados definidos en *states*, el alfabeto indicado en *alphabet* y las transiciones en *transitions*. El autómata Automata1 aparecerá en el archivo final de LaTeX.

```
// Creamos un autómata AFD
DFA Automata1: [

       // Declaramos los estados
       states: { u, >s, *w, *q },

       // Definimos el alfabeto
       alphabet: {a, b},

       // Declaramos las transiciones
       transitions: {
              |s|-b->|u| ,
              |w|<-b->|u|,            // Equivale a w|-b->u y u|-b->w
              |{s,u}|-a->|w|,         // Equivale a s|-a->w y u|-a->w
              |q|-{a,b}->|w|,
              |w|-a->|q|
       }
]
```

Ejemplo 2: Creación de un autómata AFND, al cual se lo nombra *Automata2*, y de uno AFND-Lambda, *Automata3*, con los estados definidos en *states*, el alfabeto indicado en *alphabet* y las transiciones en *transitions*. Se agregan variables y constantes globales del tipo state, transitions y alphabet para definir estados, transiciones y alfabetos respectivamente que puedan ser utilizados por cualquier autómata. A su vez, se agregan operaciones de conjuntos. Los autómatas Automata2 y Automata3 aparecerán en el archivo final de LaTeX.

```
// Definimos valores constantes y variables que podrán ser empleados para crear
autómatas
// la letra inicial del nombre de dichas constantes debe estar en mayúscula

states Q: { r, o, j };
transitions R:  |s|-a->|{w,q}|;    // = {s-a->w, s-a->q}
alphabet A: { a,b };
states T: t;                    // t = {t}

// Creamos un autómata AFND
NFA Automata2: [
      states: { Q, >s , *w, *q },

      alphabet: A,

      transitions: {
            |q|-a->|r| ,
            |w|<-b->|r|,
            R
      }
];

// Creamos nuevas constantes para armar el nuevo autómata
// Usamos operadores
states H: { h, o };
transitions S1: {|r|-a->|j|, |r|-b->|r|, |j|-{a,b}->|r|};
transitions S2: {|j|-a->|r|};
transitions S3: S1 - S2;       // ⇔ {|r|-a->|j|, |r|-b->|j|, |j|-b->|r|}
states KO: {k, o};

// Creamos un autómata AFND-lambda
LNFA Automata3: [
      states: { (Q^H)+(KO^T),>s , *w, *q, bin },
     // (Q^H)+(KO^T) = ({r,o,j} ∩ {h,o}) ∪ ({k,o} ∩ {t})

      alphabet: A,

      transitions: {
            |w|<-@->|r|,    // = w consume lambda para llegar a r y viceversa
            |s|-cc->|{w,q}|,
            |w|-{cc,dd}->|r|, S3
      }
]
```

Ejemplo 3: Creación de un autómata AFND, al cual se lo nombra *Automata4*, con los estados definidos en *states*, el alfabeto indicado en *alphabet* y las transiciones en *transitions*. Se agrega el uso *states*, *alphabet* y *transitions* para obtener los estados, alfabeto y transiciones de un

autómata, al igual que *initial*, *final* y *regular* para obtener los estados iniciales, finales y regulares de estados definidos globalmente y/o estados de autómatas. El autómata Automata4 aparecerá en el archivo final de LaTeX.

```
states Q: {*r, o};

NFA Automata4: [
      states: { Q.final , w, >q },    // Q.final = {*r}

      alphabet: {a, b},

      transitions: {
            |q|-a->|r| ,
            |w|<-b->|r|,
      }
];

alphabet A: Automata4.alphabet;       // A = {a, b}
states S: Automata4.states;           // S = {*r, w, >q}
states I: Automata4.states.initial;   // I = {>q}
```