Tecnológico de Monterrey

Actividad 2.1 - Diseña tu lenguaje

TC2037 - Implementación de métodos computacionales

José Pablo Zapata Lozano

A00839242

Lucia Paulina Vargas Villalobos

A01199124

Santiago Quintana Moreno

A01571222

Guillermo Lira Sánchez

A00838139

Profesor: Dra. María Valentina Narváez Terán

Grupo:602

Equipo: 2

(1 Mayo 2025)

LÉXICO DSL

Tipo de lexema	Descripción	Expresión regular
Palabra reservada ‘alfabeto’	Inicia la sección donde defines el alfabeto.	\balfabeto\b
Palabra reservada ‘estados’	Inicia la sección dónde listas los estados.	\bestados\b
Palabra reservada ‘inicio’	Inicia la sección que indica el estado inicial.	\binicio\b
Palabra reservada ‘aceptación’	Inicia la sección con los estados de aceptación.	\baceptación\b
Palabra reservada ‘transiciones’	Inicia la sección con las reglas de transición.	\btransiciones\b
Palabra reservada ‘en’	Indica en la transición el símbolo que provoca el salto.	\ben\b
Dos puntos	Separa la palabra clave de su lista o valor.	:
Coma	Separa los elementos dentro de una lista.	,
Flecha	Marca el paso de un estado a otro.	->
Identificador de estado	Nombre de un estado (letra + letra/dígitos/_).	[A-Za-z][A-Za-z0-9_]*
Comentarios de línea	Desde ‘//’ hasta fin de línea.	//.*
Espacios en blanco	Separan tokens, se ignoran.	[\t\r\n]+
Símbolos literales	Carácter o cadena entre comillas simples (‘a’, ‘-’)	'(?:[^'\\]|\\.)'

ORDEN DE RECONOCIMIENTO (Scanner)

1. Ignorar primero los tokens de espacios en blanco [t\r\n]+ y comentarios de línea (//.*), ya que no generan información útil para el parser.
2. Detectar las palabras clave del bloque (alfabeto, estados, inicio, aceptación, transiciones) y la keyword de transición (en), usando expresiones regulares con límites de palabra.
3. Reconocer los operadores y símbolos especiales:
    - Dos puntos (:)
    - Coma (,)
    - Flecha (->)
4. Capturar los identificadores de estados con [A-Za-z][A-Za-z0-9_]*.
5. Capturar los literales de símbolo entre comillas simples con '(?:[^'\\]|\\.)'.

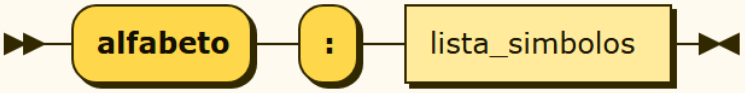
MANEJO DE ESPACIOS Y COMENTARIOS

- **Espacios en blanco:** se agrupan con [\t\r\n]+ y se descartan automáticamente en la fase de tokenización.
- **Comentarios de línea:** definidos por //.*, se eliminan antes de cualquier otro análisis.
- Cualquier carácter o secuencia que no coincida con los patrones del léxico definido debe tratarse como error léxico, notificando la posición en el archivo.

DIAGRAMAS DE SINTAXIS

Diagrama de alfabeto

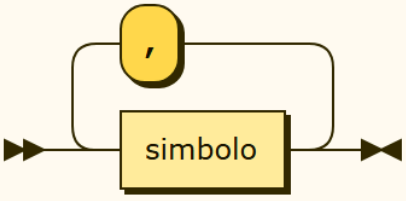
definicion_alfabeto:




```
definicion_alfabeto
        ::= 'alfabeto' ':' lista_simbolos
```

no references

lista_simbolos:

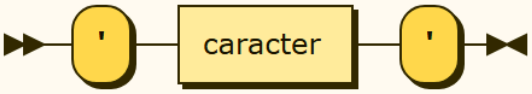



```
lista_simbolos
        ::= simbolo ( ',' simbolo )*
```

referenced by:

- definicion_alfabeto

simbolo:



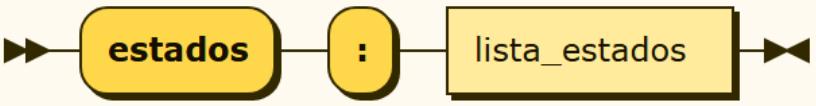

```
simbolo  ::= "'" caracter "'"
```

referenced by:

- lista_simbolos

Diagrama de estados

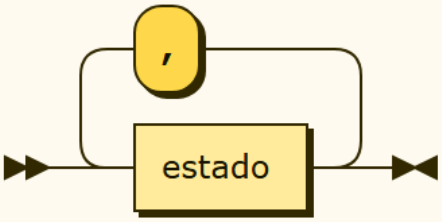
definicion_estados:




```
definicion_estados
         ::= 'estados' ':' lista_estados
```

no references

lista_estados:

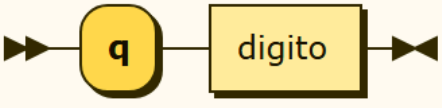



```
lista_estados
         ::= estado ( ',' estado )*
```

referenced by:

- definicion_estados

estado:

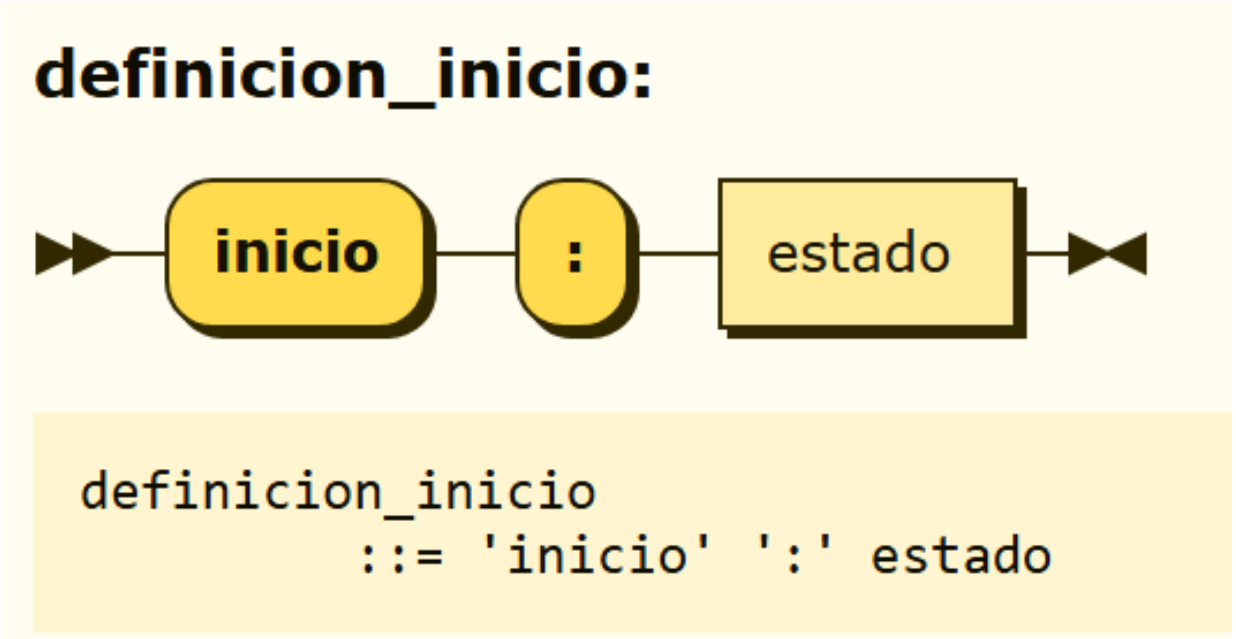



```
estado   ::= 'q' digito
```

referenced by:

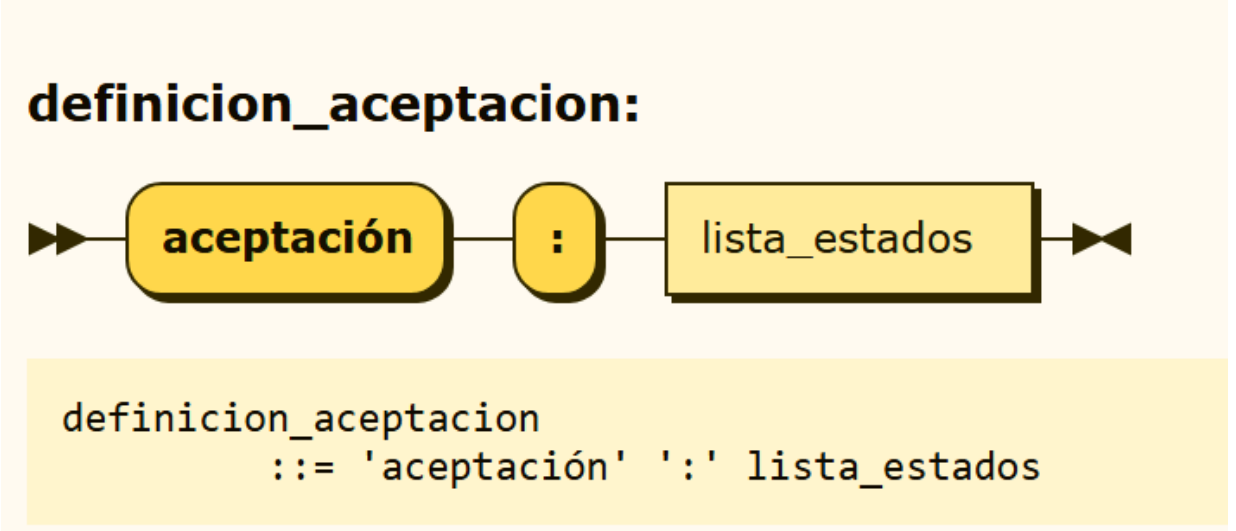
- lista_estados

Estado inicial




```
definicion_inicio
        ::= 'inicio' ':' estado
```

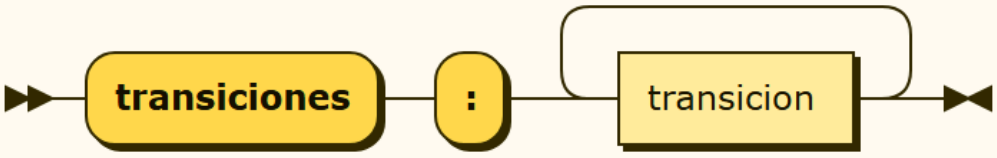
Estado de aceptación




```
definicion_aceptacion
        ::= 'aceptación' ':' lista_estados
```

Transiciones

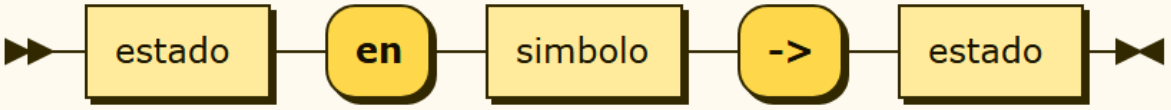
definicion_transiciones:




```
definicion_transiciones
        ::= 'transiciones' ':' transicion+
```

no references

transicion:



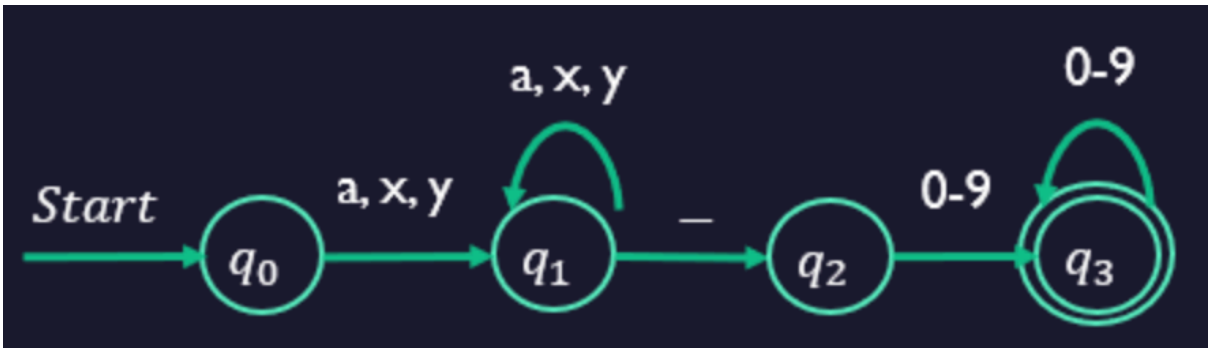

```
transicion
        ::= estado 'en' simbolo '->' estado
```

referenced by:

- definicion_transiciones

AUTÓMATAS DE EJEMPLO EN CÓDIGO

1.-



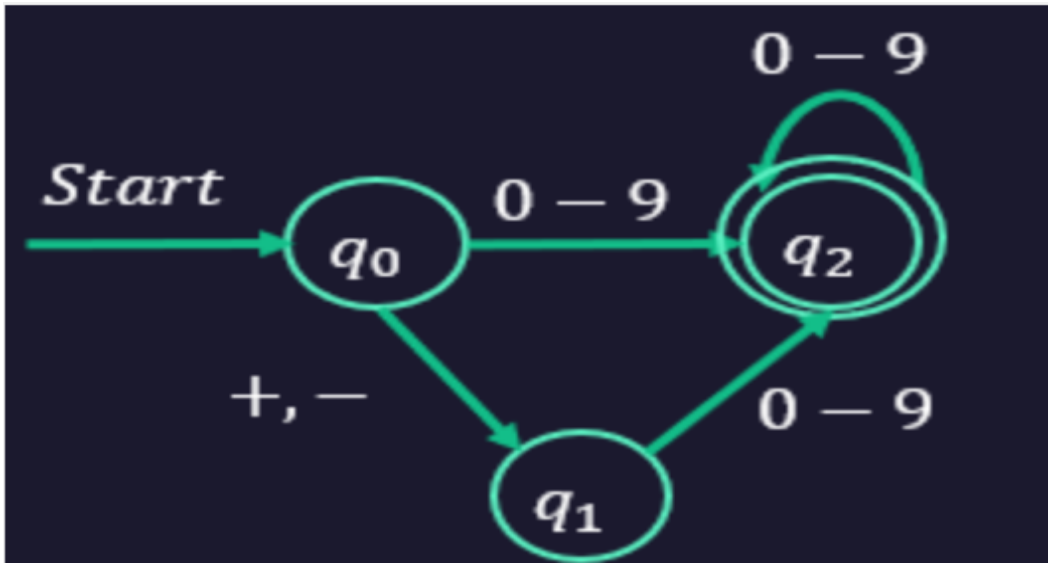
Código:

```
//Autómata 1: cadenas de {a,x,y}+ seguido de '-' y luego uno o más dígitos
alfabeto: 'a', 'x', 'y', '-', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'
estados: q0, q1, q2, q3
inicio: q0
aceptación: q3

transiciones:
  q0 -> q1 en 'a'
  q0 -> q1 en 'x'
  q0 -> q1 en 'y'
  q1 -> q1 en 'a'
  q1 -> q1 en 'x'
  q1 -> q1 en 'y'
  q1 -> q2 en '-'
  q2 -> q3 en '0'
  q2 -> q3 en '1'
  q2 -> q3 en '2'
  q2 -> q3 en '3'
  q2 -> q3 en '4'
  q2 -> q3 en '5'
  q2 -> q3 en '6'
  q2 -> q3 en '7'
  q2 -> q3 en '8'
  q2 -> q3 en '9'
  q3 -> q3 en '0'
```

```
  q3 -> q3 en '1'
  q3 -> q3 en '2'
  q3 -> q3 en '3'
  q3 -> q3 en '4'
  q3 -> q3 en '5'
  q3 -> q3 en '6'
  q3 -> q3 en '7'
  q3 -> q3 en '8'
  q3 -> q3 en '9'
```

2.-



Código:

```
//Autómata 2: enteros con signo opcional (+ o -) seguido de uno o más dígitos
alfabeto: '+', '-', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'
estados: q0, q1, q2
inicio: q0
aceptación: q2

transiciones:
  //dígitos directos desde q0
  q0 -> q2 en '0'
  q0 -> q2 en '1'
  q0 -> q2 en '2'
  q0 -> q2 en '3'
```

```
q0 -> q2 en '4'
q0 -> q2 en '5'
q0 -> q2 en '6'
q0 -> q2 en '7'
q0 -> q2 en '8'
q0 -> q2 en '9'
//signo opcional
q0 -> q1 en '+'
q0 -> q1 en '-'
//tras signo, al menos un dígito
q1 -> q2 en '0'
q1 -> q2 en '1'
q1 -> q2 en '2'
q1 -> q2 en '3'
q1 -> q2 en '4'
q1 -> q2 en '5'
q1 -> q2 en '6'
q1 -> q2 en '7'
q1 -> q2 en '8'
q1 -> q2 en '9'
//bucle de dígitos en q2
q2 -> q2 en '0'
q2 -> q2 en '1'
q2 -> q2 en '2'
q2 -> q2 en '3'
q2 -> q2 en '4'
q2 -> q2 en '5'
q2 -> q2 en '6'
q2 -> q2 en '7'
q2 -> q2 en '8'
q2 -> q2 en '9'
```

Integrante	Tareas realizadas	Decisiones de diseño y justificación
Pablo Zapata	-Definición del léxico y elaboración de la tabla de tokens . -Escritura de las expresiones regulares para cada token.	- Opté por usar nombres en español (alfabeto, estados, inicio, etc.) para mejorar la legibilidad del DSL en nuestro contexto académico. - Decidí incluir límites de palabra (\b...\b) en las regex de palabras clave para evitar coincidencias parciales con identificadores más largos.
Guillermo Lira	- Redacción de la gramática EBNF completa para todas las instrucciones. - Creación de los diagramas de sintaxis (railroad) con BottleCaps.	- Escogí EBNF porque es un estándar ampliamente reconocido y fácil de traducir a parseadores automáticos. - En los diagramas agrupé las listas con cajas repetitivas para enfatizar la repetición de símbolos o estados y hacerlos más claros.
Lucia Vargas	- Traducción de las palabras clave del DSL al español. - Definición de la regla léxica de literales (manejo de escapes) y comentarios. - Redacción de ejemplos de prueba para cada instrucción (alfabeto, estados, inicio, aceptación, transiciones). - Generación de la sección de documentación de sintaxis (casos de uso) en el informe.	- Mantuve la flecha -> sin escaparse en la regex para simplificar el scanner. - Decidí que los comentarios se ignoren completamente, usando la regla //.*, de forma que el parser no procese líneas de documentación. - Opté por estandarizar la forma de escribir literales con '(?:[^'\\]|\\.)'.
Santiago Quintana	- Escritura de los scripts de los dos autómatas de ejemplo. - Validación manual de cadenas de prueba para asegurar el correcto funcionamiento del DSL.	- Defino el autómata de enteros con signo de modo que el signo sea opcional, añadiendo un estado intermedio ‘q1’ tras ‘+/-’. - Incluí bucles en el estado de aceptación para gestionar cadenas de longitud arbitraria.