Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI3725 - Traductores e Interpretadores
Septiembre-Diciembre 2008

Examen Parcial III

(25 puntos)

Carnet:

Nombre:

1. Considere la gramática G

$$
\begin{array}{lcl}
T & \rightarrow & T\,\mathbf{f}\,T \\
T & \rightarrow & T\,\mathbf{e}\,T \\
T & \rightarrow & \mathbf{i}N \\
N & \rightarrow & \mathbf{i}N \\
N & \rightarrow & \lambda
\end{array}
$$

   (a) **(2 puntos)** Calcule el Autómata de Prefijos Viables para la gramática extendida y determine si la gramática es LR(0).
   Aumentamos la Gramática con un nuevo símbolo inicial y denotamos cada producción con un número para futura referencia

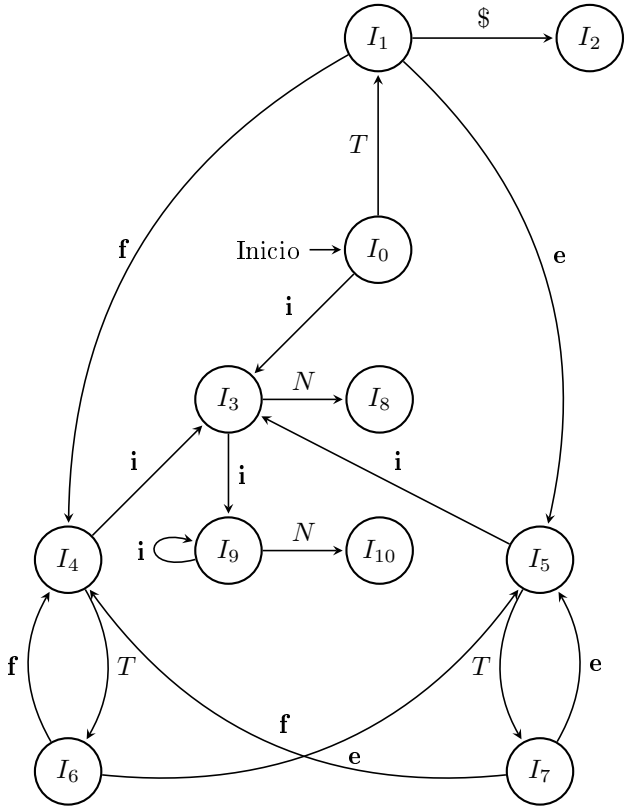
$$
\begin{array}{lcl}
\text{Regla 0:} \quad S & \rightarrow & T\$ \\
\text{Regla 1:} \quad T & \rightarrow & T\,\mathbf{f}\,T \\
\text{Regla 2:} \quad T & \rightarrow & T\,\mathbf{e}\,T \\
\text{Regla 3:} \quad T & \rightarrow & \mathbf{i}\,N \\
\text{Regla 4:} \quad N & \rightarrow & \mathbf{i}\,N \\
\text{Regla 5:} \quad N & \rightarrow & \lambda
\end{array}
$$

   Tomamos el primer item de la producción 0, i.e. $S \rightarrow \cdot T\$$, y calculamos su clausura para construir el Conjunto de Items I_0. Ese conjunto será el estado inicial a partir del cual calculamos el resto de los Conjuntos de Items según las transiciones:

$$
\begin{array}{lcl}
I_0 & : & S \rightarrow \cdot T\$ \\
 & & T \rightarrow \cdot T\mathbf{f}T \\
 & & T \rightarrow \cdot T\mathbf{e}T \\
 & & T \rightarrow \cdot \mathbf{i}N \\
I_1 & : & S \rightarrow T \cdot \$ \\
 & & T \rightarrow T \cdot \mathbf{f}T \\
 & & T \rightarrow T \cdot \mathbf{e}T \\
I_2 & : & S \rightarrow T\$ \cdot \\
I_3 & : & T \rightarrow \mathbf{i} \cdot N \\
 & & N \rightarrow \cdot \mathbf{i}N \\
 & & N \rightarrow \cdot
\end{array}
$$

$$
\begin{array}{rcl}
I_4 & : & T \to T\mathbf{f} \cdot T \\
 & & T \to \cdot T\mathbf{f}T \\
 & & T \to \cdot T\mathbf{e}T \\
 & & T \to \cdot \mathbf{i}N \\
I_5 & : & T \to T\mathbf{e} \cdot T \\
 & & T \to \cdot T\mathbf{f}T \\
 & & T \to \cdot T\mathbf{e}T \\
 & & T \to \cdot \mathbf{i}N \\
I_6 & : & T \to T\mathbf{f}T \cdot \\
 & & T \to T \cdot \mathbf{f}T \\
 & & T \to T \cdot \mathbf{e}T \\
I_7 & : & T \to T\mathbf{e}T \cdot \\
 & & T \to T \cdot \mathbf{f}T \\
 & & T \to T \cdot \mathbf{e}T \\
I_8 & : & T \to \mathbf{i}N \cdot \\
I_9 & : & N \to \mathbf{i} \cdot N \\
 & & N \to \cdot \mathbf{i}N \\
 & & N \to \cdot \\
I_{10} & : & N \to \mathbf{i}N \cdot
\end{array}
$$

El Autómata de Prefijos Viables nos queda como



La Gramática **no** es LR(0) pues presenta conflictos *shift-reduce* en los conjuntos I_6 e I_7.

(b) **(3 puntos)** Construya la Tabla de Parsing SLR usando el algoritmo descrito en clase. Aparecerán conflictos que Ud. debe resolver, justificando cada caso, sabiendo que:

i. **e** asocia hacia la *derecha*.

ii. **f** asocia hacia la *izquierda*.

iii. **e** tiene precedencia sobre **f**.

Para construir la Tabla de Parsing SLR es necesario calcular el $FOLLOW$ de todos los símbolos no terminales. Así tenemos

$$\begin{aligned} FIRST(S) = FIRST(T) &= \{\mathbf{i}\} \\ FIRST(N) &= \{\lambda, \mathbf{i}\} \\ FOLLOW(S) &= \{\$\} \\ FOLLOW(T) = FOLLOW(N) &= \{\$, \mathbf{e}, \mathbf{f}\} \end{aligned}$$

y la Tabla de Parsing nos queda

Estado	**e**	**f**	**i**	$	T	N
0			s3		1	
1	s5	s4		s2		
2				accept		
3	r5	r5	s9	r5		8
4			s3		6	
5			s3		7	
6	**s5**/r1	s4/**r1**		r1		
7	**s5**/r2	s4/**r2**		r2		
8	r3	r3		r3		
9	r5	r5	s9	r5		10
10	r4	r4		r4		

La tabla presenta cuatro conflictos *shift/reduce* los cuales deben ser resueltos usando las reglas de precedencia y asociatividad estipuladas. Los casos a considerar son:

i. Estado 6, $shift5 - reduce1$ con el símbolo **e** en la entrada. El conflicto está entre pasar al estado 5 para esperar $T \rightarrow T\mathbf{e}T$ eventualmente, o reducir $T \rightarrow T\mathbf{f}T$ inmediatamente. Como **e** tiene precedencia sobre **f**, se resuelve el conflicto haciendo $shift5$.

ii. Estado 6, $shift4 - reduce1$ con el símbolo **f** en la entrada. El conflicto está entre pasar al estado 4 para esperar un nuevo $T \rightarrow T\mathbf{f}T$ eventualmente o reducir $T \rightarrow T\mathbf{f}T$ inmediatamente. Como **f** asocia hacia la izquierda, se resuelve el conflicto haciendo $reduce1$.

iii. Estado 7, $shift5 - reduce2$ con el símbolo **e** en la entrada. El conflicto está entre pasar al estado 5 para esperar un nuevo $T \rightarrow T\mathbf{e}T$ eventualmente o reducir $T \rightarrow T\mathbf{e}T$ inmediatamente. Como **e** asocia hacia la derecha, se resuelve el conflicto haciendo $shift5$.

iv. Estado 7, $shift4 - reduce1$ con el símbolo **f** en la entrada. El conflicto está entre pasar al estado 4 para esperar un nuevo $T \rightarrow T\mathbf{f}T$ eventualmente o reducir $T \rightarrow T\mathbf{e}T$ inmediatamente. Como **e** tiene precedencia sobre **f**, se resuelve el conflicto haciendo $reduce2$.

(c) **(2 puntos)** Utilice el parser SLR construido para obtener la derivación más derecha para la palabra **iieieifififi** y muestre el árbol de derivación.

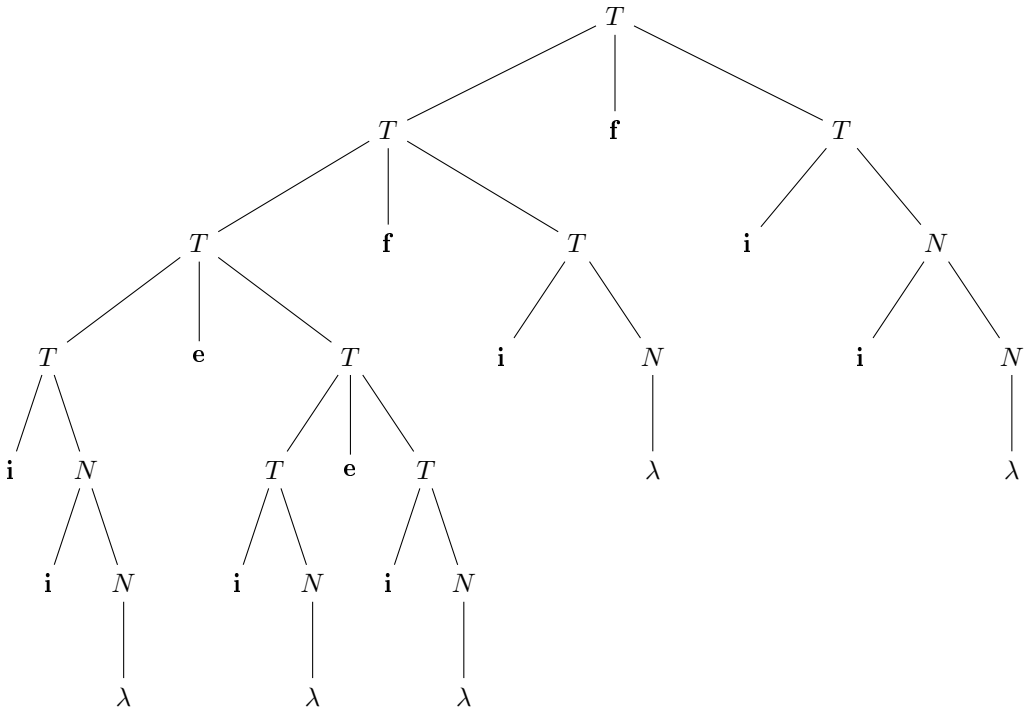
Entrada	Pila	Acción
iieieifififi\$	0 \$	shift 3
ieieifififi\$	3 0 \$	shift 9
eieifififi\$	9 3 0 \$	reduce 5; pop 0; goto(9,N)
eieifififi\$	10 9 3 0 \$	reduce 4; pop 2; goto(3,N)
eieifififi\$	8 3 0 \$	reduce 3; pop 2; goto(0,T)
eieifififi\$	1 0 \$	shift 5
ieififii\$	5 1 0 \$	shift 3
eififii\$	3 5 1 0 \$	reduce 5; pop 0; goto(3,N)
eififii\$	8 3 5 1 0 \$	reduce 3; pop 2; goto(5,T)
eififii\$	7 5 1 0 \$	shift 5
ififii\$	5 7 5 1 0 \$	shift 3
fifii\$	3 5 7 5 1 0 \$	reduce 5; pop 0; goto(3,N)
fifii\$	8 3 5 7 5 1 0 \$	reduce 3; pop 2; goto(5,T)
fifii\$	7 5 7 5 1 0 \$	reduce 2; pop 3; goto(5,T)
fifii\$	7 5 1 0 \$	reduce 2; pop 3; goto(0,T)
fifii\$	1 0 \$	shift 4
ifii\$	4 1 0 \$	shift 3
fii\$	3 4 1 0 \$	reduce 5; pop 0; goto(3,N)
fii\$	8 3 4 1 0 \$	reduce 3; pop 2; goto(4,T)
fii\$	6 4 1 0 \$	reduce 1; pop 3; goto(0,T)
fii\$	1 0 \$	shift 4
ii\$	4 1 0 \$	shift 3
i\$	3 4 1 0 \$	shift 9
\$	9 3 4 1 0 \$	reduce 5; pop 0; goto(9,N)
\$	10 9 3 4 1 0 \$	reduce 4; pop 2; goto(3,N)
\$	8 3 4 1 0 \$	reduce 3; pop 2; goto(4,T)
\$	6 4 1 0 \$	reduce 1; pop 3; goto(0,T)
\$	1 0 \$	shift 2
\$	2 1 0 \$	accept

y la derivación más derecha para la palabra **iieieifififi** se construye usando en orden inverso las reglas indicadas por las reducciones, por tanto

$$
\begin{aligned}
T &\Rightarrow^1 T\mathbf{f}T \\
&\Rightarrow^3 T\mathbf{fi}N \\
&\Rightarrow^4 T\mathbf{fii}N \\
&\Rightarrow^5 T\mathbf{fii} \\
&\Rightarrow^1 T\mathbf{f}T\mathbf{fii} \\
&\Rightarrow^3 T\mathbf{fi}N\mathbf{fii} \\
&\Rightarrow^5 T\mathbf{fifii} \\
&\Rightarrow^2 T\mathbf{e}T\mathbf{fifii} \\
&\Rightarrow^2 T\mathbf{e}T\mathbf{e}T\mathbf{fifii} \\
&\Rightarrow^3 T\mathbf{e}T\mathbf{ei}N\mathbf{fifii} \\
&\Rightarrow^5 T\mathbf{e}T\mathbf{eififii} \\
&\Rightarrow^3 T\mathbf{ei}N\mathbf{eififii} \\
&\Rightarrow^5 T\mathbf{eieififii}
\end{aligned}
$$

$$\Rightarrow^3 \quad \mathbf{i}N\mathbf{eieifififi}$$
$$\Rightarrow^4 \quad \mathbf{ii}N\mathbf{eieififi}$$
$$\Rightarrow^5 \quad \mathbf{iieieififi}$$

cuyo árbol de derivación asociado, que pone en evidencia el respeto a las reglas de precedencia y asociatividad establecidas, es como sigue



2. **(8 puntos)** La numeración de Tarjetas de Crédito consiste en un lenguaje simple aplicado sobre secuencias de dígitos, que deben cumplir con ciertas propiedades aritméticas para considerarse válidas.
Defina formalmente una Gramática de Atributos con la condición que su símbolo inicial se defina como $attr(S) = \{\textbf{tipo} : string, \textbf{valid} : boolean\}$ y que permita identificar y verificar la validez de número de tarjeta, sujeto a las siguientes restricciones:

   (a) Los números de tarjeta que comienzan con el dígito 4 corresponden a VISA y deben tener 13 o 16 dígitos en total.

   (b) Los números de tarjeta que comienzan con los dígitos 51 o 55 corresponden a MasterCard y deben tener 16 dígitos en total.

   (c) Los números de tarjeta que comienzan con los dígitos 34 o 37 corresponden a American Express y deben tener 15 dígitos en total.

   (d) Calcular la verificación según el algoritmo de Luhn. Sea $d_1 d_2 \dots d_k$ el número a verificar y

$$f(d) \quad = \quad \begin{cases} 2d - 9 & si\ 2d \geq 10 \\ 2d & si\ 2d < 10 \end{cases}$$

   entonces:

      i. Si k es par, sumar $d_2 + d_4 + \cdots + d_k + f(d_1) + f(d_3) + \cdots + f(d_{k-1})$ y el resultado debe ser múltiplo de 10 para que la tarjeta sea considerada válida.

      ii. Si k es impar, sumar $d_1 + d_3 + \cdots + d_k + f(d_2) + f(d_4) + \cdots + f(d_{k-1})$ y el resultado debe ser múltiplo de 10 para que la tarjeta sea considerada válida

   (e) Ante cualquier otra numeración inicial, cantidad de dígitos o inconsistencia de la verificación de Luhn, la tarjeta se tomará como inválida pero la gramática debe terminar el reconocimiento indicando tal condición a través de S.**valid**

   En las acciones semánticas puede asumir la existencia de los operadores aritméticos y booleanos habituales, solamente dispone de la comparación por igualdad, pero **no** puede utilizar $if - then - else$ **ni** asumir la existencia de f.

   Defino la Gramática de Atributos

$$G \quad = \quad (\{S, V, M, A, P, I, D\}, \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5}, \mathbf{6}, \mathbf{7}, \mathbf{8}, \mathbf{9}\}, P, S)$$

   con las producciones que se indican más abajo, y defino los siguientes atributos para los símbolos terminales y no-terminales de interés:

   (a) Todos los terminales tienen un atributo v de tipo entero para almacenar su valor decimal y un atributo f de tipo entero para contener el valor de $f(d)$ para el dígito particular.

   (b) El no-terminal P modela las listas de dígitos de longitud par y el no-terminal I modela las listas de dígitos de longitud impar. Ambos tienen un atributo f de tipo entero para acumular el cálculo de la función de Lühn, un atributo n de tipo entero para calcular la longitud de la lista y un atributo p de tipo entero para calcular el valor decimal de sus dos primeros dígitos. De este modo es posible calcular la función de Lühn utilizando los valores de las posiciones adecuadas para cada tipo de secuencia.

   (c) Los no-terminales A, M y V modelan el reconocimiento de las secuencias de dígitos posiblemente coincidentes con una tarjeta American Express, MasterCard o VISA, y realizan la validación de forma. Tienen un atributo *valid* de tipo booleano para indicar la validez de la secuencia de dígitos. Cada uno de estos no-terminales debe analizar dos casos: procesar una secuencia de longitud par y procesar una secuencia de longitud par. Para cada caso, se verifica si la longitud de la secuencia, los dígitos iniciales y el resultado de la función de Lühn son válidos y de la conjunción de dichas verificaciones se deduce la validez de la secuencia.

   (d) El no-terminal S se define exactamente como lo especifica el enunciado y lo único que hace es utilizar lo verificado por A, M o V.

Las producciones de la gramática junto con sus acciones semánticas quedan como

$$
\begin{array}{lcll}
S & \to & A & \left\{\begin{array}{rcl} S.valid & \leftarrow & A.valid \\ S.tipo & \leftarrow & \text{'American Express'} \end{array}\right. \\
S & \to & M & \left\{\begin{array}{rcl} S.valid & \leftarrow & M.valid \\ S.tipo & \leftarrow & \text{'MasterCard'} \end{array}\right. \\
S & \to & V & \left\{\begin{array}{rcl} S.valid & \leftarrow & V.valid \\ S.tipo & \leftarrow & \text{'VISA'} \end{array}\right. \\
A & \to & P & \left\{\begin{array}{rcl} A.valid & \leftarrow & false \end{array}\right. \\
A & \to & I & \left\{\begin{array}{rcl} A.valid & \leftarrow & (I.n = 15) \wedge (I.p = 34 \vee I.p = 37) \wedge (I.f \, mod \, 10 = 0) \end{array}\right. \\
M & \to & P & \left\{\begin{array}{rcl} M.valid & \leftarrow & (P.n = 16) \wedge (P.p = 51 \vee P.p = 55) \wedge (P.f \, mod \, 10 = 0) \end{array}\right. \\
M & \to & I & \left\{\begin{array}{rcl} M.valid & \leftarrow & false \end{array}\right. \\
V & \to & P & \left\{\begin{array}{rcl} V.valid & \leftarrow & (P.n = 16) \wedge (P.p \, div \, 10 = 4) \wedge (P.f \, mod \, 10 = 0) \end{array}\right. \\
V & \to & I & \left\{\begin{array}{rcl} V.valid & \leftarrow & (I.n = 13) \wedge (I.p \, div \, 10 = 4) \wedge (I.f \, mod \, 10 = 0) \end{array}\right. \\
P & \to & D_1 D_2 P_1 & \left\{\begin{array}{rcl} P.n & \leftarrow & P_1.n + 2 \\ P.f & \leftarrow & D_1.f + D_2.v + P_1.f \\ P.p & \leftarrow & D_1.v * 10 + D_2.v \end{array}\right. \\
P & \to & D_1 D_2 & \left\{\begin{array}{rcl} P.n & \leftarrow & 2 \\ P.f & \leftarrow & D_1.f + D_2.v \\ P.p & \leftarrow & D_1.v * 10 + D_2.v \end{array}\right. \\
I & \to & D_1 D_2 I_1 & \left\{\begin{array}{rcl} I.n & \leftarrow & I_1.n + 2 \\ I.f & \leftarrow & D_1.v + D_2.f + I_1.f \\ I.p & \leftarrow & D_1.v * 10 + D_2.v \end{array}\right. \\
I & \to & D & \left\{\begin{array}{rcl} I.n & \leftarrow & 1 \\ I.f & \leftarrow & D.v \\ I.p & \leftarrow & D.v \end{array}\right. \\
D & \to & \mathbf{0} & \left\{\begin{array}{rcl} D.v & \leftarrow & 0 \\ D.f & \leftarrow & 0 \end{array}\right. \\
D & \to & \mathbf{1} & \left\{\begin{array}{rcl} D.v & \leftarrow & 1 \\ D.f & \leftarrow & 2 \end{array}\right. \\
D & \to & \mathbf{2} & \left\{\begin{array}{rcl} D.v & \leftarrow & 2 \\ D.f & \leftarrow & 4 \end{array}\right. \\
D & \to & \mathbf{3} & \left\{\begin{array}{rcl} D.v & \leftarrow & 3 \\ D.f & \leftarrow & 6 \end{array}\right. \\
D & \to & \mathbf{4} & \left\{\begin{array}{rcl} D.v & \leftarrow & 4 \\ D.f & \leftarrow & 8 \end{array}\right. \\
D & \to & \mathbf{5} & \left\{\begin{array}{rcl} D.v & \leftarrow & 5 \\ D.f & \leftarrow & 1 \end{array}\right. \\
D & \to & \mathbf{6} & \left\{\begin{array}{rcl} D.v & \leftarrow & 6 \\ D.f & \leftarrow & 3 \end{array}\right. \\
D & \to & \mathbf{7} & \left\{\begin{array}{rcl} D.v & \leftarrow & 7 \\ D.f & \leftarrow & 5 \end{array}\right. \\
D & \to & \mathbf{8} & \left\{\begin{array}{rcl} D.v & \leftarrow & 8 \\ D.f & \leftarrow & 7 \end{array}\right. \\
D & \to & \mathbf{9} & \left\{\begin{array}{rcl} D.v & \leftarrow & 9 \\ D.f & \leftarrow & 9 \end{array}\right.
\end{array}
$$

3. **(5 puntos)** Construya una Máquina de Turing determinística estándar de **dos** cintas que acepte palabras del lenguaje de los palíndromos sobre $\Sigma = \{a, b\}$. La máquina recibirá como entrada BwB con $w \in \{a, b\}^*$ y siempre debe detenerse de manera normal, aceptando por estado final solamente si w es palíndromo independientemente de su longitud, i.e. λ, *bab* y *abba* son todos palíndromos. En su respuesta debe enunciar la máquina de estados formalmente, presentar el diagrama de estados y describir el funcionamiento de la misma.

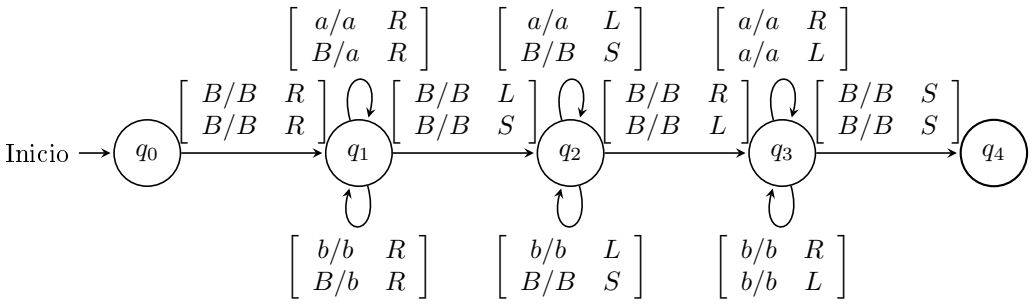
   La Máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ puede ser como sigue

$$\begin{aligned} Q &= \{q_0, q_1, q_2, q_3, q_4\} \\ \Sigma &= \{\mathbf{a}, \mathbf{b}\} \\ \Gamma &= \{\mathbf{a}, \mathbf{b}, \mathbf{B}\} \\ F &= \{q_4\} \end{aligned}$$

   cuya función de transición δ se enumera

$$\begin{aligned} \delta(q_0, B, B) &= [q_1; B, R; B, R] \\ \delta(q_1, a, B) &= [q_1; a, R; a, R] \\ \delta(q_1, b, B) &= [q_1; b, R; b, R] \\ \delta(q_1, B, B) &= [q_2; B, L; B, S] \\ \delta(q_2, a, B) &= [q_2; a, L; B, S] \\ \delta(q_2 b, B) &= [q_2; b, L; B, S] \\ \delta(q_2, B, B) &= [q_3; B, R; B, L] \\ \delta(q_3, a, a) &= [q_3; a, R; a, L] \\ \delta(q_3, b, b) &= [q_3; b, R; b, L] \\ \delta(q_3 B, B) &= [q_4; B, S; B, S] \end{aligned}$$

   El diagrama de estados correspondiente a la máquina es



   La operación de la máquina comienza por ubicarse al principio de la entrada y copiarla a la cinta dos, utilizando el ciclo del estado q_1. Una vez terminada la copia, se retrocede el cabezal de la cinta 1 hasta el inicio manteniendo el cabezal de la cinta 2 al final de la copia, utilizando el ciclo del estado q_2. El ciclo del estado q_3 mueve el cabezal de la cinta 1 de izquierda a derecha y el cabezal de la cinta 2 de derecha a izquierda siempre y cuando los símbolos sean coincidentes, garantizando que se cumple la invariante de ser palíndromos. Si el ciclo alcanza un blanco en ambos extremos, entonces la palabra es palíndromo. Nótese que si la entrada es vacía, la máquina puede transitar desde q_0 hasta q_4 sin utilizar los ciclos, aceptándola como palíndromo también.

4. **(5 puntos)** Demuestre que no existe un **algoritmo** tal que reciba la codificación de una Máquina de Turing M y que pueda **decidir** si M se detiene con la entrada 101.

Demostración utilizando Reducción.

Asumamos que existe la Máquina de Turing L_{101} definiendo un **algoritmo** que recibe como entrada la representación de una Máquina de Turing M y que acepta dicha entrada si $101 \in L(M)$ y no acepta en caso contrario, deteniéndose siempre.

Consideremos la Máquina de Turing T que solamente acepta la palabra 101 sobre el alfabeto $\{1, 0\}$ con construcción trivial de una cinta para la entrada. La máquina acepta exactamente la palabra 101 y ante cualquier diferencia se desplaza hasta el final de la cinta y se cuelga leyendo blancos.

Consideremos ahora la Máquina de Turing Pre-procesadora P definida de la siguiente forma:

(a) La entrada P es la Representación de una Máquina de Turing M y una cadena $w \in \{0,1\}^*$, i.e. $R(M)w$.

(b) La máquina P verifica que la entrada corresponda a una representación válida para una máquina. En caso contrario, se detiene rechazando la entrada.

(c) La máquina P construye la Representación de una nueva Máquina de Turing M' tal que corresponde a la Máquina de Turing M cuya operación es como sigue:

   i. Coloca w en la cinta de entrada de M de manera que M' ya tiene una entrada fija ("cableada").

   ii. Para todo estado q en M y todo símbolo $a \in \Sigma$ que **no** tenga transición definida, se define una nueva transición hacia el estado inicial de la máquina T. De este modo, en cualquier estado q en donde M se detendría, ahora pasa a ejecutar la máquina que acepta 101.

Ahora, podemos construir la Máquina de Turing H que recibe como entrada la Representación de una Máquina de Turing M y una cadena $w \in \Sigma^*$. Esta máquina:

(a) Recibe la entrada $R(M)w$ y la pasa al pre-procesador P que construye la máquina M'.

(b) La máquina P emite $R(M')$ por su salida, la cual es pasada a la máquina L_{101}.

Es claro que H aceptará $R(M)w$ si y sólo si $L(M')$ acepta 101. La construcción de M' nos asegura que esta acepta 101, siempre y cuando M se detenga con w. Esto quiere decir que H es equivalente al Problema de la Parada para Máquinas de Turing, y lo hemos reducido a L_{101} utilizando P como pre-procesador. Sabemos que H no es decidible, por tanto L_{101} no puede ser decidible como habíamos supuesto; en consecuencia **no existe** un **algoritmo** que decida si $101 \in L(M)$.

Nota: para que una reducción tenga sentido, la respuesta del problema a reducir tiene que ser la misma que daría el problema original, esto es, el problema que sale del reductor **debe** tener la misma respuesta que tendría el original. En este caso, basta cablear la entrada de w y asegurarse que la nueva máquina acepte 101**solamente** si la primera termino, y por eso se agregan las transiciones donde originalmente se habría detenido M y se obliga a T a colgarse en cualquier otro caso.

Demostración utilizando el Teorema de Rice

Se nos pide determinar si existe un **algoritmo** que permita decidir si $L(M)$ contiene a 101 para *cualquier* máquina M. Esto es, se quiere saber si la propiedad $101 \in L(M)$ es decidible. Consideremos los lenguajes:

(a) $L_\lambda = \{\lambda\}$. Este lenguaje es Recursivamente Enumerable, pues basta construir una Máquina de Turing que se detenga y acepte solamente si la entrada es BB, y se cuelgue para cualquier otra entrada.

(b) $L_{101} = \{101\}$. Este lenguaje es Recursivamente Enumerable, pues basta construir una Máquina de Turing que se detenga y acepte solamente si la entrada es $B101B$, y se cuelgue para cualquier otra entrada.

Ahora bien, $101 \in L_{101}$ pero $101 \notin L_\lambda$ por tanto la propiedad $101 \in L$ donde L es Recursivamente Enumerable es no-trivial, y de acuerdo con el Teorema de Rice, dicha propiedad no sería decidible. Si la propiedad no es decidible quiere decir que no existe un **algoritmo** que determine si 101 pertenece a $L(M)$ para una M arbitraria.