Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI3725 - Traductores e Interpretadores
Septiembre-Diciembre 2014

Carnet:_____

Nombre:_____

Examen III

(25 puntos)

Pregunta 1	Pregunta 2	Pregunta 3	Pregunta 4	Pregunta 5	**Total**
6 puntos	7 puntos	6 puntos	4 puntos	2 puntos	**25 puntos**

Pregunta 1 - 6 puntos

Esta pregunta consta de tres (3) subpreguntas de selección, numeradas de 1.1 a 1.3. Cada una de las subpreguntas viene acompañada de cuatro posibles respuestas (a, b, c, d), entre las cuales sólo **una** es correcta. Ud. deberá marcar completamente y sin posibilidad de confusión la opción que considere correcta.

Cada una de las subpreguntas respondida correctamente tiene un valor de dos (2) puntos. Las subpreguntas con más de una respuesta marcada, serán consideradas incorrectas. Dos preguntas contestadas incorrectamente, eliminan una pregunta correcta.

1.1 Si D enumera el lenguaje L en orden lexicográfico, ***siempre*** es cierto que:

(a) $\bar{L}$ no es recursivamente enumerable.

(b) Existe D' que decide L.

(c) L es infinito.

(d) Todas las anteriores.

1.2 Sea G una gramática que **no** es $SLR(1)$, entonces ***siempre*** es cierto que:

(a) No existe un reconocedor $LL(k)$ para G.

(b) Existe un reconocedor $LR(0)$ para G, que no tiene conflictos.

(c) Existe una gramática G' que genera el mismo lenguaje que G, pero que si es $SLR(1)$.

(d) El autómata de prefijos viables para G tiene por lo menos un estado que manifiesta un conflicto *shift/reduce* no resoluble usando el criterio $SLR(1)$.

1.3 Sean P_1 y P_2 problemas de decisión. Si existe una reducción polinomial de P_1 a P_2, entonces:

(a) P_1 y P_2 están en NP

(b) P_2 es $NP - Completo$

(c) P_1 y P_2 tienen la misma complejidad.

(d) Todas de las anteriores.

Pregunta 2 - 7 puntos

Sea la gramática $G = \{\{S, A, B\}, \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}, P, S\}$ con producciones P según

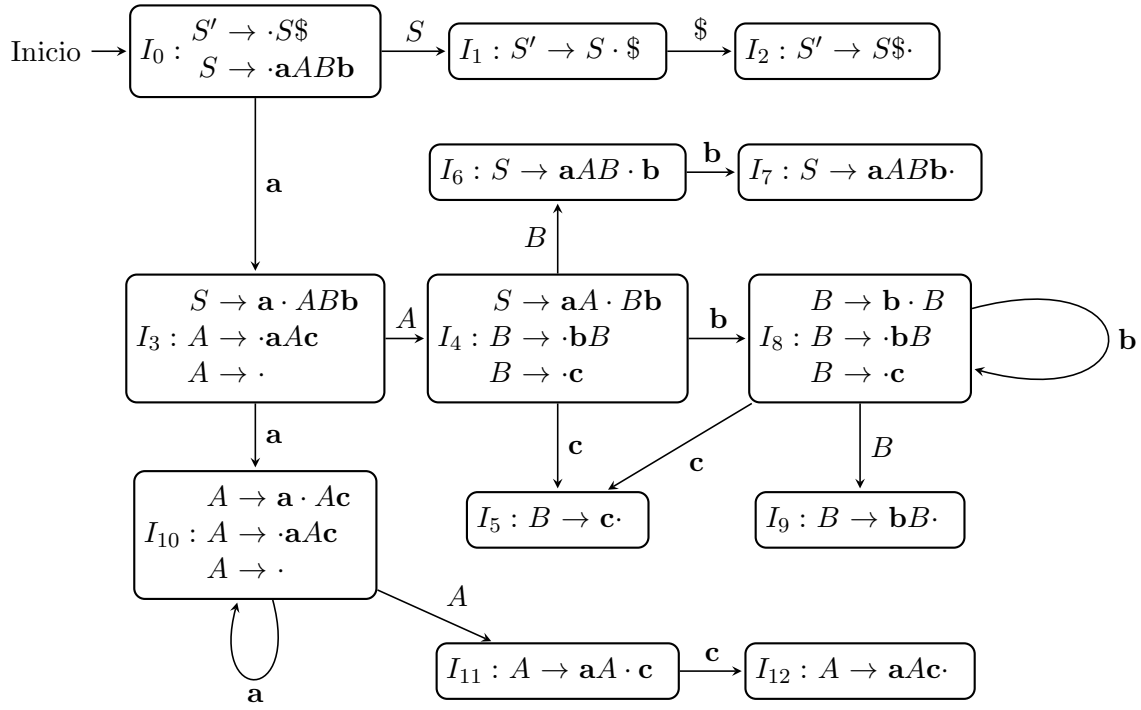
$$
\begin{aligned}
S &\to \mathbf{a}AB\mathbf{b} \\
A &\to \mathbf{a}A\mathbf{c} \\
A &\to \lambda \\
B &\to \mathbf{b}B \\
B &\to \mathbf{c}
\end{aligned}
$$

a) **(3 puntos)** Calcule el Autómata Determinístico de Prefijos Viables para la gramática extendida, y determine si la gramática es $LR(0)$.

Aumentamos la Gramática con un nuevo símbolo inicial y enumeramos cada producción:

$$
\begin{aligned}
&\textit{Regla } 0: \ S' \to S\$ \\
&\textit{Regla } 1: \ S \to \mathbf{a}AB\mathbf{b} \\
&\textit{Regla } 2: \ A \to \mathbf{a}A\mathbf{c} \\
&\textit{Regla } 3: \ A \to \lambda \\
&\textit{Regla } 4: \ B \to \mathbf{b}B \\
&\textit{Regla } 5: \ B \to \mathbf{c}
\end{aligned}
$$

El Autómata de Prefijos Viables nos queda como



La Gramática **no** es $LR(0)$ pues presenta conflictos *shift-reduce* en los conjuntos I_3 e I_{10}.

b) **(2 puntos)** Construya la Tabla de Análisis para un reconocedor ascendente utilizando el algoritmo presentado en clase, con el criterio $SLR(1)$.

Para construir la Tabla de Análisis $SLR(1)$ es necesario calcular el $FOLLOW$ de todos los símbolos no terminales. Así tenemos

$$\begin{aligned} FIRST(S') = FIRST(S) &= \{\mathbf{a}\} \\ FIRST(A) &= \{\lambda, \mathbf{a}\} \\ FIRST(B) &= \{\mathbf{b}, \mathbf{c}\} \\ FOLLOW(S') = FOLLOW(S) &= \{\$\} \\ FOLLOW(A) &= \{\mathbf{b}, \mathbf{c}\} \\ FOLLOW(B) &= \{\mathbf{b}\} \end{aligned}$$

y la Tabla de Análisis nos queda

Estado	**a**	**b**	**c**	**$**	S	A	B
0	s3				1		
1				s2			
2				Accept			
3	s10	r3	r3			4	
4		s8	s5				6
5		r5					
6		s7					
7				r1			
8		s8	s5				9
9		r4					
10	s10	r3	r3			11	
11			s12				
12		r2	r2				

c) **(2 puntos)** Use el reconocedor para encontrar la derivación más derecha de la palabra **aaaccbbcb**.

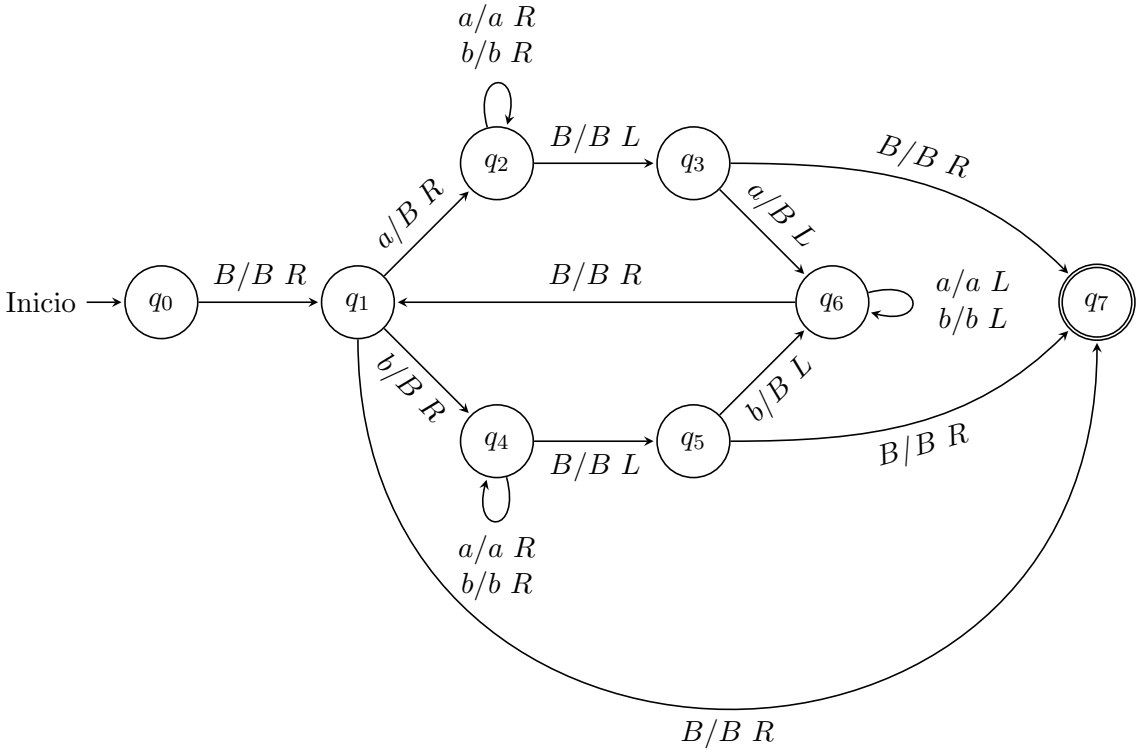
Entrada	Pila	Acción
aaaccbbcb$	0 $	shift 3
aaccbbcb$	3 0 $	shift 10
accbbcb$	10 3 0 $	shift 10
ccbbcb$	10 10 3 0 $	reduce 3; pop 0; goto(10,A)
ccbbcb$	11 10 10 3 0 $	shift 12
cbbcb$	12 11 10 10 3 0 $	reduce 2; pop 3; goto(10,A)
cbbcb$	11 10 3 0 $	shift 12
bbcb$	12 11 10 3 0 $	reduce 2; pop 3; goto(3,A)
bbcb$	4 3 0 $	shift 8
bcb$	8 4 3 0 $	shift 8
cb$	8 8 4 3 0 $	shift 5
b$	5 8 8 4 3 0 $	reduce 5; pop 1; goto(8,B)
b$	9 8 8 4 3 0 $	reduce 4; pop 2; goto(8,B)
b$	9 8 4 3 0 $	reduce 4; pop 2; goto(4,B)
b$	6 4 3 0 $	shift 7
$	7 6 4 3 0 $	reduce 1; pop 4; goto(0,S)
$	1 0 $	shift 2
$	2 1 0 $	Accept

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

$$
\begin{aligned}
S &\Rightarrow^1 \mathbf{a}A\underline{B}\mathbf{b} \\
&\Rightarrow^4 \mathbf{a}A\mathbf{b}\underline{B}\mathbf{b} \\
&\Rightarrow^4 \mathbf{a}A\mathbf{bb}\underline{B}\mathbf{b} \\
&\Rightarrow^5 \mathbf{a}\underline{A}\mathbf{bbcb} \\
&\Rightarrow^2 \mathbf{aa}\underline{A}\mathbf{cbbcb} \\
&\Rightarrow^2 \mathbf{aaa}\underline{A}\mathbf{ccbbcb} \\
&\Rightarrow^3 \mathbf{aaaccbbcb}
\end{aligned}
$$

Pregunta 3 - 6 puntos

Construya una Máquina de Turing *determinística* estándar de **una** cinta que reciba como entrada una palabra de la forma BwB, donde $w \in \{a, b\}^*$ y decida si w es palíndromo para cualquier longitud de w. Provea la representación gráfica o tabular de su máquina.



Pregunta 4 - 4 puntos

Demuestre que no existe un **algoritmo** tal que reciba la codificación de dos Máquinas de Turing M_1 y M_2 y pueda **decidir** si M_1 acepta más palabras que M_2.

Demostración usando Reducción

Supongamos que existe la Máquina de Turing $L_{|L_1|>|L2|}$ definiendo un **algoritmo** que recibe como entrada la representación de dos Máquinas de Turing M_1 y M_2, aceptando si $|L(M_1)| > |L(M_2)|$. Consideremos ahora la Máquina de Turing Pre-Procesadora P definida de la siguiente forma:

1. Su entrada es la Representación de una Máquina M y una cadena $w \in \{0, 1\}^*$, i.e. $R(M)w$.
2. La máquina P verifica que la entrada corresponda a una representación válida para una máquina. En caso contrario se cuelga.
3. La máquina P construye la representación de una nueva Máquina M_1 tal que:
   - Agrega estados y transiciones previas al estado inicial de M de manera que copien w en la cinta de entrada. Esto, en efecto, "cablea" la entrada w para M.

- Para todo estado q en M y todo símbolo $a \in \Sigma$ que **no** tenga transición definida, se define una nueva transición hacia una nueva submáquina N_1. De este modo, en cualquier estado q en el cual M se detendría, ahora pasa a una submáquina N_1.
- La submáquina N_1 tiene estados y transiciones suficientes para dejar en blanco la cinta de entrada, posicionándose en el extremo izquierdo.
- El resto de las transiciones de la submáquina N_1 son tales que acepten solamente las palabras 0 o 1, y para el resto se cuelga.

4. La máquina P construye la representación de una nueva Máquina M_2 tal que:
   - Agrega estados y transiciones previas al estado inicial de M de manera que copien w en la cinta de entrada. Esto, en efecto, "cablea" la entrada w para M.
   - Para todo estado q en M y todo símbolo $a \in \Sigma$ que **no** tenga transición definida, se define una nueva transición hacia una nueva submáquina N_2. De este modo, en cualquier estado q en el cual M se detendría, ahora pasa a una submáquina N_2.
   - La submáquina N_2 tiene estados y transiciones suficientes para dejar en blanco la cinta de entrada, posicionándose en el extremo izquierdo.
   - El resto de las transiciones de la submáquina N_2 son tales que aceptan solamente la palabra 0, y para el resto se cuelga.
5. La máquina P emite por su salida $R(M_1)R(M_2)$

Ahora, podemos construir la Máquina de Turing H que recibe como entrada la Representación de una Máquina de Turing M y una cadena $w \in \{0,1\}^*$. Esta máquina:

- Recibe la entrada $R(M)w$ y la pasa al pre-procesador P que construye las máquina M_1 y M_2.
- La máquina P emite $R(M_1)R(M_2)$ por su salida, la cual es pasada a la máquina $L_{|L_1|>|L2|}$.

Es claro que H aceptará $R(M)w$ si y sólo si $L_{|L_1|>|L2|}$ acepta $R(M_1)R(M_2)$. La construcción de M_1 nos asegura que esta aceptará solamente las palabras 0 o 1 siempre y cuando M se detenga con w, y que M_2 solamente aceptará la palabra 0 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 $L_{|L_1|>|L2|}$ utilizando a P como pre-procesador. Sabemos que H no es decidible, por tanto $L_{|L_1|>|L2|}$ no puede ser decidible como habíamos supuesto; en consecuencia, no existe un algoritmo que decida si una máquina M_1 acepta más palabras que otra máquina M_2.

Demostración usando el Teorema de Rice
(Que no es nada obvia...)

Se nos pide determinar si existe un **algoritmo** que permita decidir si $L(M_1)$ contiene más palabras que $L(M_2)$ para *cualquier* par de Máquinas M_1 y M_2.
Sabemos que *comparar* dos valores es **decidible**, por lo tanto debemos considerar si es decidible determinar $|L(M)| = n$ con $n \in \mathbb{N}$ para cualquier M. De ese modo, podríamos aplicar ese algoritmo **dos veces** (una para M_1 y otra para M_2) y luego comparar los valores obtenidos.
Consideremos los lenguajes:

- $L_{101010} = \{101010\}$ es Recursivamente Enumerable, pues basta construir una Máquina que se detenga y acepte sólo si la entrada es $B101010B$, y se cuelgue para cualquier otra entrada.
- $L_{Sigma^*} = \Sigma^*$ es Recursivamente Enumerable, pues basta construir una Máquina que acepte cualquier entrada.

Ahora bien, L_{101010} contiene exactamente una palabra, mientras que L_{Sigma^*} contiene más de una, así que la propiedad "L contiene una palabra", con L Recursivamente Enumerable, no es trivial, y de acuerdo con el Teorema de Rice, no existe **algoritmo** que la decida.
Si consideramos la propiedad "L contiene n palabras" para cualquir $n \in \mathbb{N}$, siempre podremos definir el lenguaje $L_{n\ words}$ que tiene n palabras, Recursivamente Enumerable por ser **finito**, pero L_{Sigma^*} siempre tendrá más palabras, así que la propiedad "L contiene exactamente n palabras", con L Recursivamente Enumerable, no es trivial, y de acuerdo con el Teorema de Rice, no existe **algoritmo** que la decida.
Si no puede decidirse la cantidad precisa de palabras de L Recursivamente Enumerable, tampoco es decidible comparar la cardinalidad de dos lenguajes Recursivamente Enumerables.

Pregunta 5 - 2 puntos
El Teorema de Rice establece que cualquier propiedad no trivial sobre Lenguajes Recursivamente Enumerables no es decidible. "Ser aceptado/enumerado por una Máquina de Turing" es una propiedad trivial, pues todo Lenguaje Recursivamente Enumerable es aceptado por una Máquina de Turing, y sabemos que aceptar y enumerar son equivalentes. Entonces, indique dos (2) propiedades triviales adicionales para Lenguajes Recursivamente Enumerables.
Una propiedad trivial es aquella que es cumplida por **todos** o **ninguno** de los Lenguajes Recursivamente Enumerables. Así, algunas posibles respuestas incluyen:

- No ser aceptado/enumerado por ninguna Máquina de Turing.
- Si P es una propiedad trivial, entonces $\bar{P}$ es trivial.
- Ser subconjunto de Σ^*.
- Contener a $L_\emptyset$.
- Ser aceptado por una Máquina de Turing con un número par (o impar) de estados.