Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información.
CI–3725 – Traductores e Interpretadores
Abril–Julio 2012

Examen III

Solución
(25 puntos)

0. (12 puntos) Considere cuidadosamente cada uno de los siguientes enunciados y seleccione exactamente una respuesta de entre las opciones disponibles, que haga cierto el enunciado en cuestión. Cada respuesta correcta **suma punto y medio (1.5)**, sin embargo **tres (3) respuestas incorrectas cancelan una correcta**. Tiene la opción de contestar la opción *No sabe / No contesta*, lo cual anula la puntuación de dicha respuesta, mas tampoco es penalizada. Cualquier enunciado que no sea respondido, o que sea respondido con dos o más respuestas, se considerará incorrecto.

   (a) ¿Cuál de las siguientes condiciones es suficiente para asegurar que una gramática no es LL(k), para cualquier k?

      i. Existen conflictos de la forma *shift–reduce*.

         No, estos conflictos solo aplican en LR.

      ii. La gramática no está en Forma Normal de Chomsky.

         No, esta forma normal asegura tiempo de ejecución $O(n^3)$ para el algoritmo CKY. No tiene relevancia para LL ni LR.

      iii. [[**Existe recusión izquierda, directa o indirecta.**]]

         Si, la recursión izquierda haría que la Maquina entrara en un ciclo infinito.

      iv. Ninguna de las anteriores.

         No, la opción iii. es correcta.

      v. *No Sabe / No Contesta.*

   (b) Diga cuál de los siguientes escenarios para un item LR(1), I_k, presenta un conflicto de forma *shift–reduce*. (Nota: Las letras en mayúscula son no–terminales y las letras en minúscula son terminales.)

      i. $\lceil\lceil T \to rUe., \{o, k\} \in I_k \wedge \quad \rceil\rceil$
      $\lfloor\lfloor S \to t.oRy, \{y, e, a, h\} \in I_k \rfloor\rfloor$

         Si, pues la o que está a la derecha del punto en el segundo item, aparece en los look–aheads del primero.

      ii. $C \to arLoS., \{p, e, r, z\} \in I_k \wedge$
      $R \to iCArdO., \{m, o, n, a, s, c, l\} \in I_k$

         No, aquí no existe conflicto alguno.

      iii. $T \to UriNG., \{r, o, c, k, s\} \in I_k \wedge$
      $C \to Ho.mSkY, \{r, u, l, e, s\} \in I_k$

         No, aquí no existe conflicto alguno.

      iv. $R \to IcK., \{a, s, t, l, e, y\} \in I_k \wedge$
      $R \to oLL., \{f, t, w\} \in I_k$

         No, aquí hay un conflicto, pero es reduce–reduce.

      v. *No Sabe / No Contesta.*

(c) Continuando con el enunciado anterior, diga ahora cuál de los siguientes escenarios para un item LR(1), I_k, presenta un conflicto de forma *reduce–reduce*.

i. $T \to rUe., \{o, k\} \in I_k \wedge$
$S \to t.oRy, \{y, e, a, h\} \in I_k$

No, aquí hay un conflicto, pero es shift–reduce.

ii. $C \to arLoS., \{p, e, r, z\} \in I_k \wedge$
$R \to iCArdO., \{m, o, n, a, s, c, l\} \in I_k$

No, aquí no existe conflicto alguno.

iii. $T \to UriNG., \{r, o, c, k, s\} \in I_k \wedge$
$C \to Ho.mSkY, \{r, u, l, e, s\} \in I_k$

No, aquí no existe conflicto alguno.

iv. $\lceil\lceil R \to IcK., \{a, s, t, l, e, y\} \in I_k \wedge \rceil\rceil$
$\lfloor\lfloor R \to oLL., \{f, t, w\} \in I_k \quad \rfloor\rfloor$

Si, pues la t aparece en los look–aheads tanto del primer, como del segundo item.

v. *No Sabe / No Contesta.*

(d) ¿Cuál de las siguientes afirmaciones es cierta?

i. Un *Reconocedor* siempre termina, para cualquier entrada.

No, cuando un Reconocedor *siempre termina se le conoce como* Decididor.

ii. Los *Computadores* con más de dos cintas, computan más funciones que aquellos de dos cintas.

No, cualquier Maquina de Turing multicintas tiene el mismo poder que las Máquinas de Turing de una sola cinta.

iii. Un *Enumerador* nunca termina.

No, si se está enumerando un Lenguaje finito, el Enumerador *termina al enumerarlos todos.*

iv. [[**Ninguna de las anteriores.**]]

Si, todas las anteriores son falsas.

v. *No Sabe / No Contesta.*

(e) Podemos asegurar que un Lenguaje es Recursivo si:

i. Existe un enumerador que lo enumera.

No, esto solo aseguraría que es Recursivamente Enumerable.

ii. [[**El Lenguaje y su complemento, son Recursivamente Enumerables.**]]

Si, saber si $x \in L^C$ es equivalente a saber si $x \notin L$. Y como sabemos también si $x \in L$, entonces L es Recursivo.

iii. Existe una biyección de las frases del Lenguaje, con algún subconjunto de los números naturales.

No, esto solo aseguraría que es Recursivamente Enumerable.

iv. Ninguna de las anteriores.

No, la opción ii. es correcta.

v. *No Sabe / No Contesta.*

(f) ¿Cual de las siguientes relaciones es cierta?:

i. $NPSPACE \subseteq NL$

No, no toda Máquina que toma espacio polinomial puede estar acotada también por un espacio logarítmico.

ii. $DSPACE(f(n)) = NSPACE(f(n))$, para todo f y n.

No, el Teorema de Savitch enuncia $DSPACE(f^2(n)) = NSPACE(f(n))$. El factor cuadrático es muy relevante.

iii. [[$NPSPACE \subseteq PSPACE$]]

Si, pues $NPSPACE = PSPACE$.

iv. $DTIME(f(n)) = NTIME(f^2(n))$, para todo f y n.

No, volver determinista un algoritmo no–determinista debería consumir mas tiempo. De hecho, la relación correcta es: $NTIME(f(n)) \subseteq DTIME(2^{f(n)})$.

v. *No Sabe / No Contesta.*

(g) Sea f una función y $T_f(n)$ el tiempo que toma (al mejor Computador determinista) computar el valor de f, dada una entrada de tamaño n. $T_f(n)$ está definido como:

$$T_f(n) = n^2 + 2^{\log_2 n^3} + \log_2(n * 2^n) + 3$$

¿Cuál es la clase de complejidad más pequeña a la cual pertenece f?

i. $DTIME(\log_2 n)$
ii. $DTIME(n^2)$
iii. [[$DTIME(n^3)$]]
iv. $DTIME(2^n)$
v. *No Sabe / No Contesta.*

Simplficando $T_f(n)$, nos queda la siguiente fórmula: $n^2 + n^3 + log_2(n) + n + 3$, la cual es $O(n^3)$

(h) Se puede asegurar que, dada una clase de complejidad C, un Lenguaje L es C–completo si:

i. Para cualquier otra clase de complejedidad Z, tal que $Z \subseteq C$, L pertence a la clase Z.

No, Z podría ser $\emptyset$, ya que $\emptyset \subseteq C$ y nada partenece a $\emptyset$.

ii. Existe otra clase de complejedidad Z, tal que $Z \subset C$ y L es Z–duro.

No, Z podría ser $\emptyset$, ya que $\emptyset \subseteq C$ y todos los problemas en $\emptyset$ son $\emptyset$–duros, por rango vacío. Asegurar esto sería decir que todo lenguaje L es C–completo.

iii. L pertenece a la clase C y para cualquier otro lenguaje T que pertenezca a C también, T debe ser igual a L.

No, esto implicaría que solo para las clases unitarias, su Lenguaje es C–completo.

iv. [[L **pertenece a la clase** C **y es** C**–duro.**]]

Si, esta es la definición de completitud.

v. *No Sabe / No Contesta.*

1. (3 pts) Suponga usted que cuenta con los siguientes Computadores, que no modifican sus cintas de entrada y reciben como parámetros que cintas a usar como entradas y salidas:

   - Un computador $MINUS(e_1, e_2, s)$, de tres cintas (dos de entrada y una de salida).
     - e_1: Primera cinta de entrada. Tendrá una frase de la forma 1^a, con $a \geq 0$.
     - e_2: Segunda cinta de entrada. Tendrá una frase de la forma 1^b, con $a \geq b \geq 0$.
     - s: Cinta de salida. Al ejecutar la máquina, tendrá la frase 1^{a-b}.
   - Un computador $MUL(e_1, e_2, s)$, de tres cintas (dos de entrada y una de salida).
     - e_1: Primera cinta de entrada. Tendrá una frase de la forma 1^a, con $a \geq 0$.
     - e_2: Segunda cinta de entrada. Tendrá una frase de la forma 1^b, con $b \geq 0$.
     - s: Cinta de salida. Al ejecutar la máquina, tendrá la frase 1^{a*b}.
   - Un computador $DIV(e_1, e_2, s)$, de tres cintas (dos de entrada y una de salida).
     - e_1: Primera cinta de entrada. Tendrá una frase de la forma 1^a, con $a \geq 0$.
     - e_2: Segunda cinta de entrada. Tendrá una frase de la forma 1^b, con $b > 0$.
     - s: Cinta de salida. Al ejecutar la máquina, tendrá la frase $1^{a \ div \ b}$.
   - Un computador $FACT(e, s)$, de dos cintas (una de entrada y una de salida).
     - e: Cinta de entrada. Tendrá una frase de la forma 1^a, con $a \geq 0$.
     - s: Cinta de salida. Al ejecutar la máquina, tendrá la frase $1^{a!}$.

   Se desea que implemente un Computador $COMB$, de cuatro cintas (dos de entrada, una auxiliar y una de salida) para la siguiente función:

   - La primera cinta de entrada tendrá una frase de la forma 1^a, con $a \geq 0$.
   - La segunda cinta de entrada tendrá una frase de la forma 1^b, con $a \geq b \geq 0$.
   - Al ejecutar la máquina, la cinta de salida tendrá la frase $1^{\binom{a}{b}}$, el coeficiente binomial de a en b.

   Nota: Todas las cintas de su Máquina son modificables, pero al finalizar la ejecución de la misma el resultado debe quedar en la cinta de salida. No puede utilizar ningún símbolo adicional. Puede suponer que las cintas de su Computador están enumeradas del 1 al 4, donde la cinta 1 es la primera cinta de entrada, la cinta 2 es la segunda cinta de entrada, la cinta 3 es la cinta auxiliar y la cinta 4 es la cinta de salida. Puede utilizar estos números para especificar con que cintas trabajar a los Computadores disponibles.

   Pista: Recuerde que la fórmla cerrada para calcular el coeficiente binomial es: $\binom{n}{k} = \frac{n!}{k!(n-k)!}$

Solución:

Ejecutar $MINUS(1, 2, 3)$
Ejecutar $FACT(3, 4)$
Ejecutar $FACT(2, 3)$
Ejecutar $FACT(1, 2)$
Ejecutar $MUL(3, 4, 1)$
Ejecutar $DIV(2, 1, 4)$

Nótese que, a pesar de que las Máquinas suministradas no pueden modificar sus cintas de entrada, nada impide que la Máquina $COMB$ lo haga.

2. (5 pts) Considere la siguiente Gramática:

$$\begin{array}{lcl} P & \rightarrow & P\ ,\ id\ :=\ E \\ & | & id\ :=\ E \\ E & \rightarrow & E\ <\ E\ :\ E\ > \\ & | & id \end{array}$$

a) (1 pt) Elimine la Recursión Izquierda presente en la Gramática.

Solución:

$$\begin{array}{lcl} P & \rightarrow & id\ :=\ E\ P' \\ P' & \rightarrow & ,\ id\ :=\ E\ P' \\ & | & \lambda \\ E & \rightarrow & id\ E' \\ E' & \rightarrow & <\ E\ :\ E\ >\ E' \\ & | & \lambda \end{array}$$

b) (2 pts) De la Gramática resultante, calcule los conjuntos de $FIRST_1$ y $FOLLOW_1$ para cada símbolo no–terminal de la misma.

Solución:

	$FIRST_1$	$FOLLOW_1$
P	$\{id\}$	$\{\$\}$
P'	$\{cima, \lambda\}$	$\{\$\}$
E	$\{id\}$	$\{:, >, coma, \$\}$
E'	$\{<, \lambda\}$	$\{:, >, coma, \$\}$

Se utiliza el símbolo terminal coma para representar a la coma $(,)$ *que hay en la gramàtica y que no se confunda con los separadores del conjunto.*

c) (2 pts) Utilizando el resultado anterior, construya la tabla de parsing $LL(1)$.

Solución:

Primero aumentaremos la gramática con el símbolo inicial nuevo y enumeraremos las reglas.

$$\begin{array}{llcl} (0) & S & \rightarrow & P\ \$ \\ (1) & P & \rightarrow & id\ :=\ E\ P' \\ (2) & P' & \rightarrow & ,\ id\ :=\ E\ P' \\ (3) & & | & \lambda \\ (4) & E & \rightarrow & id\ E' \\ (5) & E' & \rightarrow & <\ E\ :\ E\ >\ E' \\ (6) & & | & \lambda \end{array}$$

Ahora, procedemos a construir la table de parsing $LL(1)$.

	id	$<$	$>$	$:=$	$:$	$,$	$\$$
S							0
P	1						
P'						2	3
E	4						
E'		5	6		6	6	6

3. (5 pts) Considere la siguiente Gramática:

$$\begin{array}{lcl} S & \rightarrow & p\ T\ C \\ T & \rightarrow & o\ T \\ & | & w \\ C & \rightarrow & e\ C \\ & | & r \end{array}$$

a) (3 pts) Construya el autómata de prefijos viables LR(1) para dicha Gramática.

Solución:

Primero aumentaremos la gramática con el símbolo inicial nuevo y enumeraremos las reglas.

$$\begin{array}{llcl} (0) & S' & \rightarrow & S\ \$ \\ (1) & S & \rightarrow & p\ T\ C \\ (2) & T & \rightarrow & o\ T \\ (3) & & | & w \\ (4) & C & \rightarrow & e\ C \\ (5) & & | & r \end{array}$$

Definiremos ahora los estados del autómata.

$$\begin{array}{lllll} I_0: & S' & \rightarrow & .\ S\ \$\ , & \{\$\} \\ & S & \rightarrow & .\ p\ T\ C\ , & \{\$\} \\ \\ I_1: & S & \rightarrow & p\ .\ T\ C\ , & \{\$\} \\ & T & \rightarrow & .\ o\ T\ , & \{e, r\} \\ & T & \rightarrow & .\ w\ , & \{e, r\} \\ \\ I_2: & S & \rightarrow & p\ T\ .\ C\ , & \{\$\} \\ & C & \rightarrow & .\ e\ C\ , & \{\$\} \\ & C & \rightarrow & .\ r\ , & \{\$\} \\ \\ I_3: & S & \rightarrow & p\ T\ C\ .\ , & \{\$\} \\ \\ I_4: & C & \rightarrow & e\ .\ C\ , & \{\$\} \\ & C & \rightarrow & .\ e\ C\ , & \{\$\} \\ & C & \rightarrow & .\ r\ , & \{\$\} \\ \\ I_5: & C & \rightarrow & e\ C\ .\ , & \{\$\} \\ \\ I_6: & C & \rightarrow & r\ .\ , & \{\$\} \\ \\ I_7: & T & \rightarrow & o\ .\ T\ , & \{e, r\} \\ & T & \rightarrow & .\ o\ T\ , & \{e, r\} \\ & T & \rightarrow & .\ w\ , & \{e, r\} \\ \\ I_8: & T & \rightarrow & o\ T\ .\ , & \{e, r\} \\ \\ I_9: & T & \rightarrow & w\ .\ , & \{e, r\} \\ \\ I_{10}: & S' & \rightarrow & S\ .\ \$\ , & \{\$\} \end{array}$$

Con ese conjunto de estados plantearemos ahora las transiciones en la tabla siguiente:

	p	o	w	e	r	S	T	C
I_0	I_1					I_{10}		
I_1		I_7	I_9				I_2	
I_2				I_4	I_6			I_3
I_3								
I_4				I_4	I_6			I_5
I_5								
I_6								
I_7		I_7	I_9				I_8	
I_8								
I_9								
I_{10}								

b) (1 pt) Construya las tabla de acciones y de *go–to* a partir del autómata obtenido.

Solución:

	p	o	w	e	r	\$	S	T	C
I_0	*avanzar* 1						10		
I_1		*avanzar* 7	*avanzar* 9					2	
I_2				*avanzar* 4	*avanzar* 6				3
I_3						*reducir* 1			
I_4				*avanzar* 4	*avanzar* 6				5
I_5						*reducir* 4			
I_6						*reducir* 5			
I_7		*avanzar* 7	*avanzar* 9					8	
I_8				*reducir* 2	*reducir* 2				
I_9				*reducir* 3	*reducir* 3				
I_{10}						*aceptar*			

Todas las casillas que están vacías corresponden al valor rechazar.

c) (1 pt) ¿Admite la Gramática un parser LALR(1)? De ser así, diga cuantos estados tendría el autómata de prefijos viables LALR(1).

Solución:

Dado que los cuerpos de los items presentes en cada conjunto de estados es diferente, el autómata construido ya está en $LALR(1)$. En consecuencia, la gramática si admite un parser $LALR(1)$ y tiene 11 estados (los mismos que $LR(1)$ en este caso).

Nota: Era posible realizar un estado explícito que avanzara el \$ *a partir del item* 10. *Hacerlo es completamente válido y la respuesta sería idéntica, salvo que se tendrían 12 estados en vez de 11.*

‘‘PAGINA EN BLANCO PARA REALIZAR CÁLCULOS.’’