Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI3725 - Traductores e Interpretadores
Septiembre-Diciembre 2007

Examen Parcial III

(25 puntos)

Carnet:

Nombre:

1. Considere la gramática G

$$\begin{aligned} T &\rightarrow T\,\mathbf{f}\,T \\ T &\rightarrow T\,\mathbf{e}\,T \\ T &\rightarrow \mathbf{i} \end{aligned}$$

(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

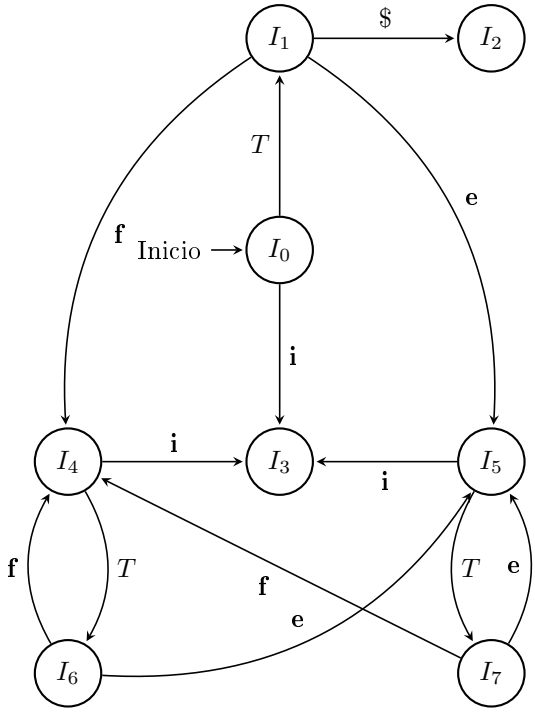
$$\begin{array}{lrcl} \text{Regla 0:} & S & \rightarrow & T\$ \\ \text{Regla 1:} & T & \rightarrow & T\mathbf{f}T \\ \text{Regla 2:} & T & \rightarrow & T\mathbf{e}T \\ \text{Regla 3:} & T & \rightarrow & \mathbf{i} \end{array}$$

Calculamos los Conjuntos de Items y su Clausura partiendo de I_0 como sigue:

$$\begin{array}{lcl} I_0 & : & S \rightarrow \cdot T\$ \\ & & T \rightarrow \cdot T\mathbf{f}T \\ & & T \rightarrow \cdot T\mathbf{e}T \\ & & T \rightarrow \cdot \mathbf{i} \\ I_1 & : & S \rightarrow T \cdot \$ \\ & & T \rightarrow T \cdot \mathbf{f}T \\ & & T \rightarrow T \cdot \mathbf{e}T \\ I_2 & : & S \rightarrow T\$ \cdot \\ I_3 & : & T \rightarrow \mathbf{i} \cdot \\ I_4 & : & T \rightarrow T\mathbf{f} \cdot T \\ & & T \rightarrow \cdot T\mathbf{f}T \\ & & T \rightarrow \cdot T\mathbf{e}T \\ & & T \rightarrow \cdot \mathbf{i} \\ I_5 & : & T \rightarrow T\mathbf{e} \cdot T \\ & & T \rightarrow \cdot T\mathbf{f}T \\ & & T \rightarrow \cdot T\mathbf{e}T \\ & & T \rightarrow \cdot \mathbf{i} \end{array}$$

$$
\begin{array}{rcl}
I_6 & : & T \to T\mathbf{f}T\cdot \\
 & & T \to T\cdot \mathbf{f}T \\
 & & T \to T\cdot \mathbf{e}T \\
I_7 & : & T \to T\mathbf{e}T\cdot \\
 & & T \to T\cdot \mathbf{f}T \\
 & & T \to T\cdot \mathbf{e}T
\end{array}
$$

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

$$
\begin{array}{rcl}
FIRST(S) = FIRST(T) & = & \{\mathbf{i}\} \\
FOLLOW(S) & = & \{\$\} \\
FOLLOW(T) & = & \{\mathbf{e}, \mathbf{f}, \$\}
\end{array}
$$

y la Tabla de Parsing nos queda

Estado	**e**	**f**	**i**	$	T
0			s3		1
1	s5	s4		s2	
2				accept	
3	r3	r3		r3	
4			s3		6
5			s3		7
6	**s5**/r1	**s4**/r1		r1	
7	s5/**r2**	s4/**r2**		r2	

La tabla presenta cuatro conflictos *shift-reduce* los cuales deben ser resueltos usando las regla 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 $T \to T\mathbf{e}T$ eventualmente, o reducir $T \to T\mathbf{f}T$ 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 6 para esperar un nuevo $T \to T\mathbf{f}T$ eventualmente o reducir $T \to T\mathbf{f}T$ inmediatamente. Como **f** asocia hacia la derecha, se resuelve el conflicto haciendo $shift4$.

iii. Estado 7, $shift5 - reduce2$ con el símbolo **e** en la entrada. El conflicto está entre pasar al estado 4 para esperar un nuevo $T \to T\mathbf{e}T$ eventualmente o reducir $T \to T\mathbf{e}T$ inmediatamente. Como **e** asocia hacia la izquierda, se resuelve el conflicto haciendo $reduce2$.

iv. Estado 7, $shift4 - reduce2$ con el símbolo **f** en la entrada. El conflicto está entre pasar al estado 5 para esperar $T \to T\mathbf{f}T$ eventualmente o reducir $T \to T\mathbf{e}T$ inmediatamente. Como **e** tiene precedencia sobre **f**, se resuelve el conflicto haciendo $reduce2$.

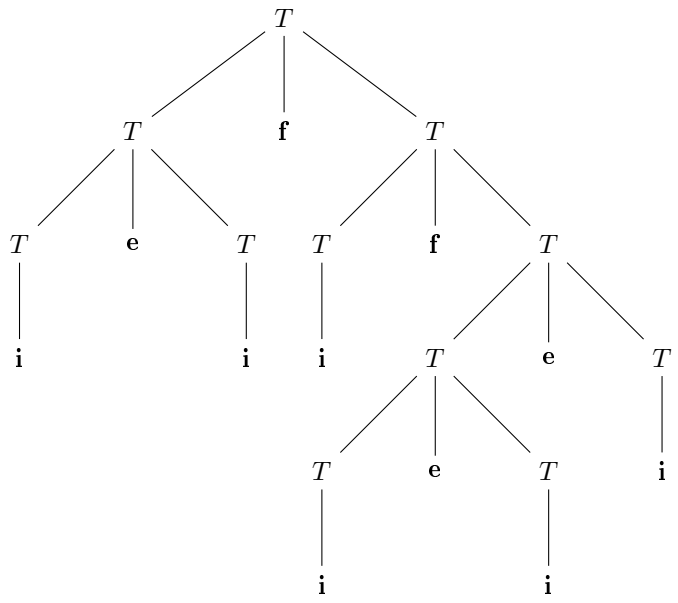
(c) **(2 puntos)** Utilice el parser SLR construido para obtener la derivación más derecha para la palabra **ieififieiei** y muestre el árbol de derivación.

Entrada	Pila	Acción
ieififieiei$	0$	shift 3
eififieiei$	$\underline{3}$0$	reduce 3; pop 1; goto(0,T)
eififieiei$	10$	shift 5
ififieiei$	510$	shift 3
fifieiei$	$\underline{3}$510$	reduce 3; pop 1; goto(5,T)
fifieiei$	$\underline{751}$0$	reduce 2; pop 3; goto(0,T)
fifieiei$	10$	shift 4
ifieiei$	410$	shift 3
fieiei$	$\underline{3}$410$	reduce 3; pop 1; goto(4,T)
fieiei$	6410$	shift 4
ieiei$	46410$	shift 3
eiei$	$\underline{3}$46410$	reduce 3; pop 1; goto(4,T)
eiei$	646410$	shift 5
iei$	5646410$	shift 3
ei$	$\underline{3}$5646410$	reduce 3; pop 1; goto(5,T)
ei$	$\underline{756}$46410$	reduce 2; pop 3; goto(4,T)
ei$	646410$	shift 5
i$	5646410$	shift 3
$	$\underline{3}$5646410$	reduce 3; pop 1; goto(5,T)
$	$\underline{756}$46410$	reduce 2; pop 3; goto(4,T)
$	$\underline{646}$410$	reduce 1; pop 3; goto(4,T)
$	$\underline{641}$0$	reduce 1; pop 3; goto(0,T)
$	10$	shift 2
$	210$	accept

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

$$
\begin{aligned}
T &\Rightarrow^1 T\mathbf{f}\underline{T} \\
&\Rightarrow^1 T\mathbf{f}T\mathbf{f}\underline{T} \\
&\Rightarrow^2 T\mathbf{f}T\mathbf{f}T\mathbf{e}\underline{T} \\
&\Rightarrow^3 T\mathbf{f}T\mathbf{f}\underline{T}\mathbf{ei} \\
&\Rightarrow^2 T\mathbf{f}T\mathbf{f}T\mathbf{e}\underline{T}\mathbf{ei} \\
&\Rightarrow^3 T\mathbf{f}T\mathbf{f}\underline{T}\mathbf{eiei} \\
&\Rightarrow^3 T\mathbf{f}\underline{T}\mathbf{fieiei} \\
&\Rightarrow^3 \underline{T}\mathbf{fifieiei} \\
&\Rightarrow^2 T\mathbf{e}\underline{T}\mathbf{fifieiei} \\
&\Rightarrow^3 \underline{T}\mathbf{eififieiei} \\
&\Rightarrow^3 \mathbf{ieififieiei}
\end{aligned}
$$

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



2. Sea la Gramática G que describe el lenguaje de los números romanos escritos en minúscula menores a 1000

$$\begin{array}{rcl} R & \rightarrow & CDU \\ C & \rightarrow & E \mid \mathbf{cd} \mid \mathbf{d}E \mid \mathbf{cm} \\ E & \rightarrow & \lambda \mid E\mathbf{c} \\ D & \rightarrow & F \mid \mathbf{xl} \mid \mathbf{l}F \mid \mathbf{xc} \\ F & \rightarrow & \lambda \mid F\mathbf{x} \\ U & \rightarrow & G \mid \mathbf{iv} \mid \mathbf{v}G \mid \mathbf{ix} \\ G & \rightarrow & \lambda \mid G\mathbf{i} \end{array}$$

(a) **(3 puntos)** Defina formalmente una Gramática de Atributos que utilice exactamente **un** atributo por cada símbolo no terminal, tal que:

i. Asegure que la cantidad de **c**, **x** e **i** en secuencia siempre sea menor o igual a tres.

ii. Permita calcular el valor numérico decimal a partir del número romano en particular.

En las acciones semánticas puede asumir la existencia de una función *error* para reportar errores semánticos y utilizar selectores $if-then-else$, además de los operadores aritméticos habituales, pero solamente dispone de la comparación por igualdad.

Defino la Gramática de Atributos

$$G = (\{R, C, E, D, F, U, G\}, \{\mathbf{c}, \mathbf{d}, \mathbf{i}, \mathbf{m}, \mathbf{v}, \mathbf{x}\}, P, R\}$$

con las mismas producciones, y defino los siguientes atributos para los símbolos no-terminales de interés:

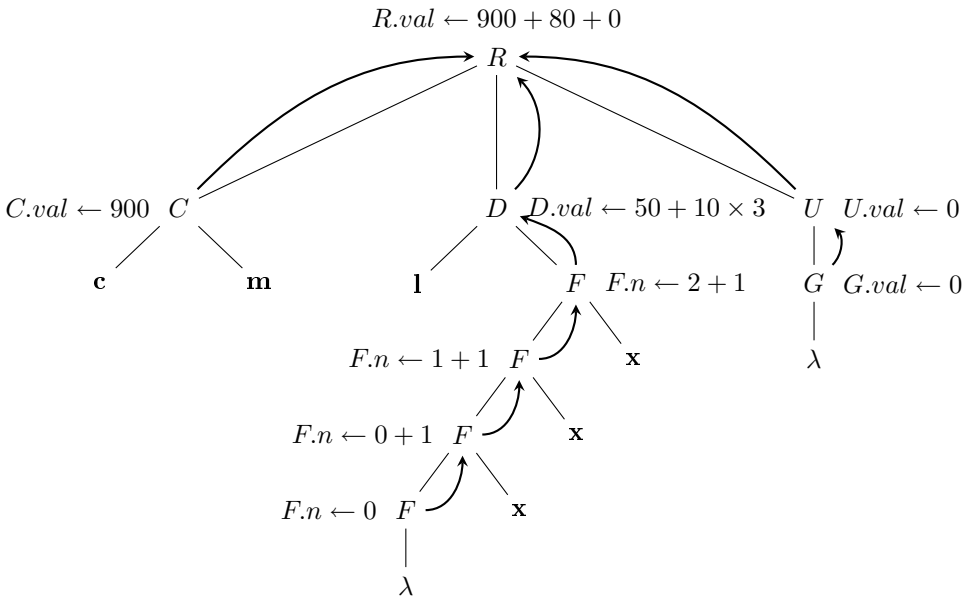
i. Los no-terminales R, C, D y U tendrán asociado el atributo v de tipo entero. Este atributo servirá para calcular el valor decimal asociado.

ii. Los no-terminales E, F y G tendrán asociado el atributo n de tipo entero. Este atributo servirá para contar el número de ocurrencias de los terminales **c**, **x** e **i** respectivamente, y participarán en el cálculo del valor decimal asociado.

Finalmente, las acciones semánticas quedan como

$$\begin{array}{rcll} R & \rightarrow & CDU & \{R.v := C.v + D.v + U.v\} \\ C & \rightarrow & E & \{C.v := 100 * E.n\} \\ C & \rightarrow & \mathbf{cd} & \{C.v := 400\} \\ C & \rightarrow & \mathbf{d}E & \{C.v := 500 + 100 * E.n\} \\ C & \rightarrow & \mathbf{cm} & \{C.v := 900\} \\ E & \rightarrow & \lambda & \{E.n := 0\} \\ E & \rightarrow & E_1\mathbf{c} & \{\textbf{if } E_1.n = 3 \textbf{ then } error \textbf{ else } E.n := E_1.n + 1\} \\ D & \rightarrow & F & \{D.v := 10 * F.n\} \\ D & \rightarrow & \mathbf{xl} & \{D.v := 40\} \\ D & \rightarrow & \mathbf{l}F & \{D.v := 50 + 10 * F.n\} \\ D & \rightarrow & \mathbf{xc} & \{D.v := 90\} \\ F & \rightarrow & \lambda & \{F.n := 0\} \\ F & \rightarrow & F_1\mathbf{x} & \{\textbf{if } F_1.n = 3 \textbf{ then } error \textbf{ else } F.n := F_1.n + 1\} \\ U & \rightarrow & G & \{U.v := G.n\} \\ U & \rightarrow & \mathbf{iv} & \{U.v := 4\} \\ U & \rightarrow & \mathbf{v}G & \{U.v := 5 + G.n\} \\ U & \rightarrow & \mathbf{ix} & \{U.v := 9\} \\ G & \rightarrow & \lambda & \{G.n := 0\} \\ G & \rightarrow & G_1\mathbf{i} & \{\textbf{if } G_1.n = 3 \textbf{ then } error \textbf{ else } G.n := G_1.n + 1\} \end{array}$$

(b) **(3 puntos)** Presente el árbol de derivación decorado con el cálculo de atributos para la palabra **cmlxxx**.



3. **(6 puntos)** Construya una Máquina de Turing determinística estándar de **una** sola cinta semi-infinita que acepte palabras del lenguaje $L = \{0^k | k = 2^n\}$. La máquina recibirá como entrada BwB con $w \in 0^*$ y siempre debe detenerse de manera normal, aceptando por estado final solamente si $|w| = 2^n$ para algún $n \geq 0$. En su respuesta debe enunciar la máquina de estados formalmente, presentar el diagrama de estados y describir el funcionamiento de la máquina.

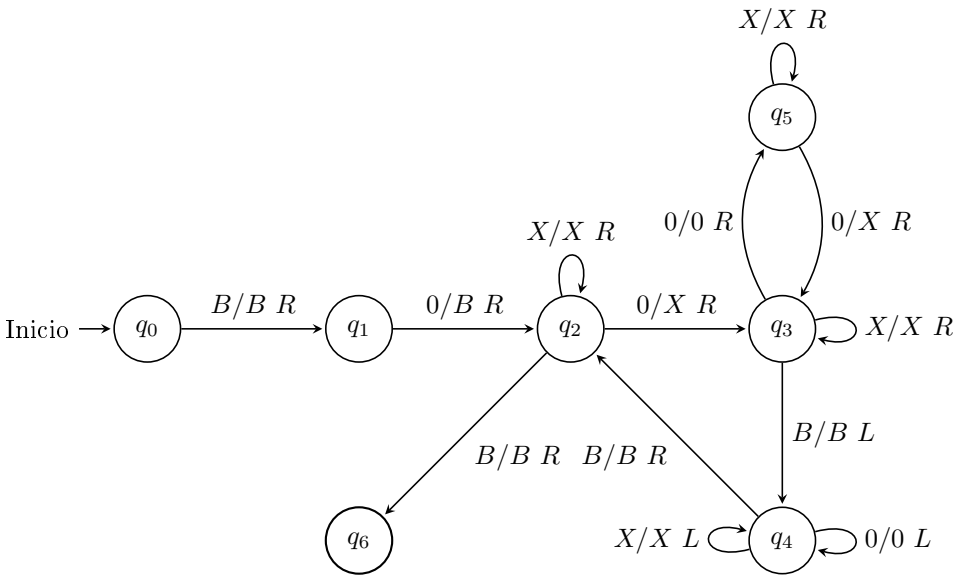
   La Máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ puede ser como sigue

$$\begin{aligned} Q &= \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\} \\ \Sigma &= \{0\} \\ \Gamma &= \{0, X, B\} \\ F &= \{q_6\} \end{aligned}$$

   cuya función de transición δ se enumera

$$\begin{aligned} \delta(q_0, B) &= (q_1, B, R) \\ \delta(q_1, 0) &= (q_2, B, R) \\ \delta(q_2, X) &= (q_2, X, R) \\ \delta(q_2, B) &= (q_6, B, R) \\ \delta(q_2, 0) &= (q_3, X, R) \\ \delta(q_3, X) &= (q_3, X, R) \\ \delta(q_3, 0) &= (q_5, 0, R) \\ \delta(q_3, B) &= (q_4, B, L) \\ \delta(q_4, 0) &= (q_4, 0, L) \\ \delta(q_4, X) &= (q_4, X, L) \\ \delta(q_4, B) &= (q_2, B, R) \\ \delta(q_5, X) &= (q_5, X, R) \\ \delta(q_5, 0) &= (q_3, X, R) \end{aligned}$$

El diagrama de estados correspondiente a la máquina es



La construcción de la máquina se desprende de un razonamiento simple: si $k = 2^n$ quiere decir $\frac{k}{2^n} = 1$, es decir, que podemos dividir k entre 2 tantas veces como sea necesario hasta llegar a uno; para dividir entre 2 simplemente vamos tachando (marcando con X) uno de cada dos 0 de la palabra original. Si durante el proceso nos queda una cantidad impar de 0 quiere decir que el número no era divisible entre 2 y por lo tanto no puede ser potencia de 2. Tenemos el caso borde en el cual $k = 1$ que es el único caso en el cual la cantidad de 0 es impar. Entonces la máquina opera así:

(a) Al comienzo de la entrada *tiene* que haber al menos un 0 pues la mínima potencia de 2 sería $2^0 = 1$. Por lo tanto desde el estado q_1 procederemos hacia q_2 solamente si hay un 0, el cual será cambiado por un B para seguir siendo usado como "centinela" del extremo izquierdo de la cinta. Si hubiera solamente un 0 en la cinta, inmediatamente se pasaría del estado q_2 al q_6 aceptando la entrada.

(b) Se establece un ciclo entre los estados q_2, q_3 y q_5 que comienza por tachar el siguiente 0 y luego tachar un 0 no, un 0 si, así hasta llegar al extremo derecho de la palabra. Como puede haber tachaduras previas, estas son ignoradas tanto en q_3 como q_5.

   i. El extremo derecho de la palabra puede alcanzarse con un conteo impar de 0 bien sea en q_2 que no encontró un 0 para tachar, o bien en q_5 que no encontró un 0 para tachar. Como ninguno de esos dos estados es final, si el conteo es impar la máquina se detiene pero no acepta.

   ii. El extremo derecho de la palabra puede alcanzarse con un conteo par de 0 en q_3. En ese caso, se utiliza q_4 para moverse hacia la izquierda, saltando encima de 0 y X, hasta llegar al B que actúa como centinela.

(c) Una vez de regreso al B que actúa como centinela:

   i. Si en la cinta solamente quedan X hasta el extremo derecho quiere decir que pudieron tacharse todos los 0 en sucesivas pasadas indicando que la longitud es potencia de 2, entonces q_2 salta por encima de todas las X y pasará al estado final q_6 al encontrar el B al extremo.

   ii. Si en la cinta queda algún 0, la transición de q_2 a q_3 se ejercitará nuevamente, repitiendo el ciclo de marcado, i.e. dividiendo entre dos, como se describe en el paso (b).

4. **(6 puntos)** Demuestre que no existe un **algoritmo** tal que reciba la codificación de una Máquina de Turing M y que pueda **decidir** si $L(M)$ tiene exactamente tres palabras.

   Asumamos que existe la Máquina de Turing L_3 definiendo un **algoritmo** que recibe como entrada la representación de una Máquina de Turing M y que acepta la entrada si $L(M)$ tiene exactamente tres palabras y no acepta en caso contrario, deteniéndose siempre.

   Consideremos la Máquina de Turing T que solamente acepta las palabras '**foo**', '**bar**' y '**baz**' sobre el alfabeto $\{\mathbf{a}, \mathbf{b}, \mathbf{f}, \mathbf{o}, \mathbf{r}, \mathbf{z}\}$, con construcción trivial en dos cintas: la primera cinta es la cinta de entrada; la segunda cinta contiene $B\mathbf{foo}B\mathbf{bar}B\mathbf{baz}B$; la máquina compara la cinta de entrada con cada uno de los componentes de la segunda cinta, aceptando si la comparación es exitosa y rechazando si no lo es. La Máquina de Turing T puede ser codificada sobre el alfabeto $\{0, 1\}$ siguiendo el método estándar descrito en la literatura y en clase.

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

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

   (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 una nueva máquina M' tal que:

      i. Simula la ejecución de M sobre la entrada w.

      ii. Si la simulación de M rechaza la entrada, entonces la máquina M' también rechaza la entrada.

      iii. Si la simulación de M acepta la entrada w, entonces se utiliza la máquina T sobre la entrada w. Si T acepta, entonces M'acepta; si T rechaza, entonces M' rechaza.

   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 produce la máquina intermedia M'.

   (b) La máquina intermedia M' es pasada como entrada a la máquina L_3.

   Es claro que H aceptará $R(M)w$ si y sólo si $L(M')$ tiene exactamente tres palabras. La construcción de M' nos asegura que solamente aceptará las palabras '**foo**', '**bar**' y '**baz**', siempre y cuando el cómputo de M acepte w. Esto quiere decir que H es equivalente al Problema de la Parada para Máquinas de Turing, y lo hemos reducido a L_3 utilizando P como preprocesador. Sabemos que H no es decidible, por tanto L_3 no puede ser decidible como habíamos supuesto; en consecuencia **no existe** un **algoritmo** que decida si $L(M)$ tiene exactamente tres palabras.

   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. Bastaba construir cualquier máquina intermedia que aceptara exactamente tres palabras.

5. **(3 puntos extra)** Sean $L_1, L_2, \ldots, L_k$ una colección de lenguajes sobre un alfabeto Σ cualquiera, tales que:

   (a) $\forall i \neq j,\ L_i \cap L_j = \emptyset$

   (b) $L_1 \cup L_2 \cup \cdots \cup L_k = \Sigma^*$

   (c) L_i es *recursivamente enumerable* $\forall i = 1, 2, \ldots, k$

   Demuestre que todos los L_i son *recursivos*.

   Nota: si la sumatoria de los puntos obtenidos en las preguntas 1 a 4 **no** son suficientes para aprobar el examen, esta pregunta **no** será tomada en cuenta.

   La condición (a) nos dice que cualquier palabra $w \in \Sigma^*$ está contenida exactamente en uno de los L_i. La condición (b) nos dice que la unión de todos los L_i es equivalente a Σ^*. Consideremos la palabra $x \in \Sigma^*$ y digamos que $x \in L_j$; entonces x no está en ninguno de los otros L_i cuando $i \neq j$. Sabemos que L_j es recursivamente enumerable por la condición (c), y si consideramos $L_1 \cup L_2 \cup \cdots \cup L_{j-1} \cup L_{j+1} \cup \cdots \cup L_k$ notaremos que se trata de la unión de lenguajes Recursivamente Enumerables, que por un Teorema estudiado en clase, también es Recursivamente Enumerable. Pero $L_1 \cup L_2 \cup \cdots \cup L_{j-1} \cup L_{j+1} \cup \cdots \cup L_k = \Sigma^* - L_j = \overline{L_j}$. Así tenemos que L_j es Recursivamente Enumerable y $\overline{L_j}$ también es Recursivamente Enumerable; nuevamente, por un Teorema estudiado en clase sabemos que cuando un lenguaje L es Recursivamente Enumerable y su complemento también lo es, entonces el lenguaje es Recursivo.