Computabilidad y Complejidad

Boletín de Ejercicios 1 -- Soluciones (Máquina de Turing)

---

01. Sea `P` la operación sobre lenguajes definida como sigue: para cada palabra `x` del lenguaje, si `x` contiene un número par de símbolos `b`, entonces cada símbolo `a` de x pasa a ser `aa`; si la palabra `x` tiene un número impar de símbolos `b`, entonces queda como está. Por ejemplo, si `x = babaa`, entonces `P(x) = baabaaaa`; si `x = baa`, entonces `P(x) = baa`. ¿Es la familia de los lenguajes recursivamente enumerables cerrada respecto de la operación `P`?

La familia de los lenguajes recursivamente enumerables es cerrada bajo `P`. Sea L_{pb} `= (a*ba*b)*a*` [el lenguaje formado por todas aquellas palabras `x` tales que $|x|_b$ es un número par] y sea L_{ib} su complementario. Ambos son regulares y, por tanto, recursivamente enumerables. Sea `h` el homomorfismo tal que `h(a) = aa` y `h(b) = b`. Entonces $L = (L \cap L_{pb}) \cup (L \cap L_{ib})$ y $P(L) = h(L \cap L_{pb}) \cup (L \cap L_{ib})$. Así, puesto que la clase de los lenguajes recursivamente enumerables es cerrada bajo homomorfismos, unión e intersección, si `L` es recursivamente enumerable también lo será `P(L)`; por lo que la clase es cerrada bajo `P`.

02. Sean L_1 y L_2 lenguajes. Se define la operación & como sigue:

$$L_1 \& L_2 = \{ x \in L_1 / (\exists y \notin L_2) (|x| = |y|)\}.$$

¿Es la clase de los lenguajes recursivos cerrada bajo la operación &?

La clase de los lenguajes recursivos sí es cerrada bajo &. Supóngase que el complementario de L_2 (en adelante L_2^C) es finito, puesto que

$$L_1 \& L_2 = \{ x \in L_1 / (\exists y \in L_2^C) (|x| = |y|)\},$$

también lo es $L_1 \& L_2$ y por tanto es recursivo.

Supóngase en lo que sigue que L_2^C es infinito. Sean L_1 y L_2 dos lenguajes recursivos. El lenguaje L_2^C será también recursivo y por tanto existirán máquinas de Turing M_1 y G_2^C tales que M_1 acepta a L_1 y se detiene para cada entrada y G_2^C genera a L_2^C en orden canónico. A partir de M_1 y G_2^C construimos una máquina M que funciona de la siguiente manera: Cuando M recibe una entrada x simula en primer lugar a M_1 sobre x. Si M_1 rechaza a x, M se detiene rechazando. En caso contrario, M activa al generador G_2^C e irá, por tanto, generando las palabras de L_2^C en orden canónico deteniéndose momentáneamente (en una pausa) cada vez que genere una palabra. Así cada vez que G_2^C genera una palabra z, M comparará |x| con |z|. Si |z| < |x|, M activa de nuevo a G_2^C que genera la siguiente palabra repitiéndose este proceso. Si |z| = |x|, M se detiene aceptando. Si |z| > |x|, M se detiene rechazando (puesto que habrá verificado que no hay ninguna palabra en L_2^C de la misma longitud que x). Por tanto la máquina M así construida acepta el lenguaje $L_1 \& L_2$ y se detiene para cada entrada. En consecuencia $L_1 \& L_2$ es recursivo.

03. Sean $L \subseteq \Sigma^*$ y $a \in \Sigma$. Se define la operación $Insertar(a,L) = \{xay \ / \ xy \in L\}$.

a) Si L es recursivo ¿lo es también $Insertar(a,L)$?

Si L es recursivo también lo es $Insertar(a,L)$. Sea M una MT que reconoce a L y que se detiene para cada entrada. Definimos una MT M' que actúa de acuerdo al siguiente algoritmo de cómputo:

> Operando a partir del comienzo de x:
> **SI** existe una nueva ocurrencia de a, todavía no considerada, **ENTONCES** eliminarla y aplicar la palabra resultante x' a M, si ésta se detiene aceptando hacer lo mismo, en otro caso restaurar x y volver a proceder;
> **EN OTRO CASO** detenerse rechazando.

Para realizar este cómputo se utiliza una cinta multibanda con:

- Un sector para la palabra de entrada x.

- Un sector para marcar la primera posición de la cinta para, al operar, no salirse de la misma.

- Un sector en la cinta para realizar el pertinente marcaje que controla el procesamiento de las a's. Marca en cada etapa el símbolo a eliminado para dar lugar a x'.

- Dos sectores, con los que se opera en consonancia, para procesar la correspondiente entrada x' obtenida a partir de x con el correspondiente símbolo a (en cada caso) eliminado. En uno de ellos se realiza el procesamiento de la entrada x', en el otro se lleva un marcaje ($\hat{}$) que señala la posición más a la derecha alcanzada en el sector de procesamiento de x'. De este modo, cuando (si ha lugar) se ha de procesar una nueva entrada x' (con excepción de la primera) antes de hacerlo, como paso previo, lo primero que se realiza es limpiar, poniendo a blancos, el sector de procesamiento de x' utilizando el marcaje del otro sector (como delimitador del tramo de cinta a considerar), modificando su marcaje de modo que se ajuste inicialmente a la entrada x' (lo que inicialmente se hace cuando se procesa la primera x').

Claramente M' reconoce a $Insertar(a,L)$ y se detiene para cada entrada, por tanto $Insertar(a,L)$ es recursivo.

b) Si L es recursivamente enumerable ¿lo es también $Inserta(a,L)$?

Si L es recursivamente enumerable también lo es $Inserta(a,L)$. Sea G un generador de L, seguidamente definimos un generado G' para $Inserta(a,L)$. El generador G' simula a G de modo que cuando G genera una palabra x, G' genera las $|x|+1$ palabras obtenidas a partir de x insertando un símbolo a en cada una de las posiciones admisibles.

04. Sean L y L' dos lenguajes, se define la operación F de modo que

$$F(L,L') = \{x \in L \ / x^r \notin L'\}.$$

Si L y L' son lenguajes recursivos ¿lo es también $F(L,L')$?

Si L y L' son recursivos, entonces también lo es $F(L,L')$. Para demostrarlo definimos una MT F que lo reconoce y que se detiene para cada entrada. Sea M una MT que reconoce a L y que se detiene para cada entrada y M' una MT que reconoce a L' y que también se detiene para cada entrada. F opera como sigue: dada una entrada x la aplica a M (copiándola previamente en otra

cinta asociada a M), si M rechaza F se detiene rechazando; en otro caso obtiene el reverso de x y lo aplica a M', si M' rechaza F se detiene aceptando, en otro caso se detiene rechazando. Para obtener el reverso de la entrada x puede recorrerla de derecha a izquierda e ir escribiendo los símbolos, en la secuencia leída, en otra cinta, asociada a M', avanzando sobre ella de izquierda a derecha..

05. ¿Son los lenguajes recursivamente enumerables cerrados para los homomorfismos inversos? ¿Y los lenguajes recursivos?

Sea un homomorfismo $h: \Sigma^* \longrightarrow \Gamma^*$ y sea $L \subseteq \Gamma^*$ un lenguaje recursivamente enumerable. Demostramos que también lo es $h^{-1}(L)$. Recuérdese que $h^{-1}(L) = \{x \in \Sigma^* / h(x) \in L\}$. Puesto que h admite una descripción finita puede codificarse en la función de transición (en el control finito) de una MT. Sea M una MT tal que L(M) = L. Seguidamente definimos una MT M' con $L(M') = h^{-1}(L)$. M' tiene h embebido en su función de transición y opera como sigue: dada una entrada x, M' obtiene h(x) y se la aplica a M dejándole a ésta el resto de la computación. Por tanto M' acepta a x si y sólo si M acepta a h(x), esto es $L(M') = \{x \in \Sigma^* / h(x) \in L\} = h^{-1}(L)$.

Para el caso de los lenguajes recursivos se procede de modo similar partiendo de una máquina de Turing M que reconozca a L y se detenga para cada entrada.

06. ¿Son los lenguajes recursivamente enumerables cerrados para los homomorfismos?

Sea un homomorfismo $h: \Sigma^* \longrightarrow \Gamma^*$ y sea $L \subseteq \Sigma^*$ un lenguaje recursivamente enumerable. Demostramos que también lo es h(L). Recuérdese que $h(L) = \{h(x) / x \in L\}$. Puesto que h admite una descripción finita puede codificarse en la función de transición de un generador de Turing. Sea M un generador de Turing tal que G(M) = L. Seguidamente definimos un generador de Turing M' con G(M') = h(L). M' tiene h embebido en su función de transición y opera como sigue: simula a M de modo que cada vez que M genera una palabra x, M' genera la palabra h(x). Por tanto M' genera el lenguaje $GL(M') = \{h(x) / x \in L\} = h(L)$, por lo que h(L) es un lenguaje recursivamente enumerable.

07. Sean L_1, L_2 y L lenguajes no vacíos. Sea $\mathscr{L}_{REN}$ la clase de los lenguajes recursivamente enumerables. Pruebe o refute la siguiente implicación:

$(\forall L_1, L_2, L)$

$$[(L_1 \in \mathscr{L}_{REN} \wedge L_2 \in \mathscr{L}_{REN} \wedge L_1 \cap L_2 = \varnothing \wedge L_1 \cap L = \varnothing \wedge L \cap L_2 = \varnothing) \Rightarrow L \in \mathscr{L}_{REN}]$$

El enunciado es falso. Sea L un lenguaje no recursivamente enumerable. L^c es un lenguaje infinito y por tanto existen dos palabras diferentes que pertenecen al mismo; sean x_1 y x_2 dos palabras en estas condiciones. Sean ahora $L_1 = \{x_1\}$ y $L_2 = \{x_2\}$. Se tiene que $L_1 \in \mathscr{L}_{REN} \wedge L_2 \in \mathscr{L}_{REN} \wedge L_1 \cap L_2 = \varnothing \wedge L_1 \cap L = \varnothing \wedge L \cap L_2 = \varnothing$ y sin embargo $L \notin \mathscr{L}_{REN}$.

08. Se define la siguiente operación sobre palabras: $P(x) = 0^n x 0^n$ donde $n = |x|_0$. Esta operación se extiende del modo usual a lenguajes, esto es: $P(L) = \{P(x) / x \in L\}$. Pruebe o refute las siguientes implicaciones:

1) L es recursivo $\Rightarrow$ P(L) es recursivo.

Veamos que P(L) es recursivo cuando L lo es. Puesto que L es recursivo existe una MT M que se detiene para cada entrada y que reconoce a L. Sea M' la MT constituida, y que opera, como sigue: M' tiene un módulo M" que cuando recibe la entrada z (de M') examina si su número de ceros es (o no) múltiplo de 3, ya que si pertenece a P(L) necesariamente ha de ser así. [Esto se puede llevar a cabo utilizando tres estados: p_0, p_1 y p_2; se inicia el proceso en p_0 y cada vez que se lee un cero se transita al siguiente estado (operando consecuentemente en módulo 3), esto es, se transita de p_0 a p_1, de p_1 a p_2 y de p_2 a p_0. Si al terminar se está en el estado p_0, entonces el número de ceros es múltiplo de 3 , en cualquier otro caso no lo es.] Si no es múltiplo de 3, M' se detiene sin aceptar. Si el número de ceros es múliplo de 3, entonces M" lo divide por 3 [este cociente puede obtenerse incrementando un contador, originalmente a cero, cada vez que se realiza la transición de p_2 a p_0] y examina si existe en la entrada z un prefijo y un sufijo de ceros con esta longitud. Si no los hay M' se detiene sin aceptar.
Si los hay los elimina y emite la palabra z' resultante, continuándose como sigue. Seguidamente arranca la máquina M y le suministra la entrada z'. Procediendo a partir de este punto tal y como lo hace M. Así M' es una MT de que se detiene para cada entrada y que reconoce a P(L).

2) P(L) es recursivo $\Rightarrow$ L es recursivo.

Del apartado anterior se deriva que la operación P es inyectiva, en consecuencia podemos proceder como sigue. Puesto que P(L) es recursivo existe una MT M_1 que se detiene para cada entrada y que lo reconoce. Sea M_1' la MT definida como sigue: Cuando recibe una entrada x calcula su número de ceros n y le añade el prefijo y el sufijo 0^n; la palabra x', así formada, $x' = 0^n x 0^n$, se aplica a M_1 procediendo, a partir de aquí, igual que M_1. Así M_1' es una MT que se detiene para cada entrada y que reconoce a L.

09. Sea Σ un alfabeto y $R \subseteq \Sigma^*$ un lenguaje recursivo dado. Para L, $L' \subseteq \Sigma^*$ se define

$$L \blacklozenge L' = \{x \in L \,/\, (\exists y \in L')\ (\text{prefijos}(x) \cap \text{sufijos}(y) \cap R \neq \varnothing)\}.$$

Si L y L' son lenguajes recursivamente enumerables ¿lo es también $L \blacklozenge L'$?

Veremos que en las condiciones dadas el lenguaje $L \blacklozenge L'$ es recursivamente enumerable. Puesto que R es un lenguaje recursivo existe una máquina de Turing que se detiene para cada entrada y que reconoce a R, sea M_R un máquina en estas condiciones. Puesto que L es un lenguaje recursivamente enumerable existe una máquina de Turing M_L que lo reconoce. Puesto que L' es un lenguaje recursivamente enumerable existe un generador de Turing $G_{L'}$ que lo genera. En lo que sigue vamos a definir una máquina de Turing M que reconozca a $L \blacklozenge L'$. M está constituida y opera como sigue. Cuando se le aplica una entrada x a M ésta le aplica x a M_L, si M_L reconoce a x, y sólo en este caso, se arranca el generador $G_{L'}$ de modo que inicie la generación de cada palabra cada vez que se le aplique una señal de continuación. Cada vez que $G_{L'}$ genera una palabra y se le aplica, junto con x, a una máquina PS que obtendrá uno por uno todos los prefijos (de x) y sufijos (de y) comunes, mientras sea posible, cada vez que se le aplique una señal de operación. (Para llevar a cabo esta tarea se puede disponer de dos cintas, una para x y otra para y, cada una con dos sectores para mantener una marca que especifique el prefijo y sufijo considerado cada vez; esta marca se va desplazando a mediada que se realiza le cálculo.) De este modo el módulo PS obtiene el primer prefijo-sufijo común (λ) y se le aplica a M_R si ésta acepta la palabra x es aceptada por M, en otro caso se envía a PS la señal de operación para genere el siguiente prefijo-sufijo común y se vuelve a proceder, cuando ya no haya más prefijo-sufijos comunes se envía la señal a $G_{L'}$ para que continúe con el proceso de generación. De este modo el lenguaje que acepta la máquina M es $L \blacklozenge L'$.

10. Dados un alfabeto Σ, $L \subseteq \Sigma^*$ y $x \in \Sigma^*$, se define la operación cociente

$$x^{-1}L = \{ y \in \Sigma^* / xy \in L \}.$$

a) Si L es un lenguaje recursivo ¿lo es también $x^{-1}L$ para cada $x \in \Sigma^*$?

Sí. Si L es un lenguaje recursivo existe una máquina de Turing M que lo reconoce y que se detiene para cada entrada. Seguidamente definiremos una máquina de Turing M' que se detendrá para cada entrada y que reconocerá a $x^{-1}L$. La máquina M' está constituida, de modo que tiene a la palabra x integrada en su función de transición, y opera como sigue. Cuando se le aplica una palabra y se la concatena a x (que se encuentra almacenada en el control finito) obteniendo la palabra xy que seguidamente se le aplica a M comportándose a partir de aquí como ésta. Así se tiene que $L(M') = x^{-1}L$.

b) Si para $x \in \Sigma^*$, $x^{-1}L$ es un lenguaje recursivo ¿lo es también L?

No necesariamente. Sea el alfabeto $\Sigma = \{0,1,\$\}$, y sea $L \subseteq \{0,1\}^*$ un lenguaje recursivemente enumerable pero no recursivo. En estas condiciones: $\$^{-1}L = \varnothing$. Así $\$^{-1}L$ es recursivo sin serlo L.

c) Si L es un lenguaje recursivamente enumerable ¿lo es también $x^{-1}L$ para cada $x \in \Sigma^*$?

Sí. Como L es un lenguaje recursivamente enumerable existe un generador M que lo genera, esto es, G(M) = L. Seguidamente definimos una máquina de Turing M' que genera a $x^{-1}L$. La máquina M' está constituida, de modo que tiene a la palabra x integrada en su función de transición, y opera como sigue. Simula al generador M de modo que cuando éste genera una palabra z (momentáneamente se detiene esperando una señal de continuación para continuar con la generación de L), M' comprueba si z tiene como prefijo a x, en cuyo caso elimina x de z obteniendo el correspondiente sufijo z' que es generado, en otro caso no se realiza ninguna generación relativa a la palabra z. Seguidamente se envía a M la señal para que continúe con la generación de las palabras de L.

11. Sea L un lenguaje, se define $P(L) = \{x / (\exists u,v)(x = uv \wedge vu \in L)\}$.

I. Si L es un lenguaje recursivamente enumerable ¿lo es también P(L)?
II. Si L es un lenguaje recursivo ¿lo es también P(L)?

I. Si L es un lenguaje recursivamente enumerable, entonces también lo es P(L). Para establecerlo demostraremos que existe un generador de Turing que genera P(L). Puesto que L es recursivamente enumerable existe un generador de Turing M tal que G(M) = L. A partir de M definiremos un generador M' tal que G(M') = P(L). M' está definido para operar como sigue: utiliza el generador M para generar palabras de L; inicialmente opera hasta generar la primera, luego seguirá operando cuando reciba una señal de generación. Cada vez que M genera una palabra z ésta se aplica a un módulo F que factoriza secuencialmente la misma en todos los pares (v,u) tales que z = vu, generando seguidamente, a su vez, en cada caso, la palabra uv. Cuando F termina este proceso envía a M una señal de generación para que continúe con la tarea de generación de L. Así se tiene que G(M') = P(L). Para llevar a cabo estas factorizaciones en el módulo F puede utilizarse una cinta con dos sectores: el inferior para z y el superior para llevar mediante una marca, ✓, el control de donde termina v y donde comienza u; por ejemplo, puede tomarse que u comience en la posición marcada. Esta marca se avanza una posición, si es posible, cada vez que se recibe una señal de factorización después de haber generado la anterior palabra uv correspondiente a la z en curso; cuando ya no se puede avanzar la marca se envía al generador la señal de generación anteriormente mencionada. Se comienza colocándola sobre el primer símbolo de z ($v = \lambda$ y $u = z$) y se termina en la celdilla, en blanco, siguiente al último símbolo de z ($v = z$ y $u = \lambda$). Para el caso $z = \lambda$ se toma directamente como única factorización $u = v = \lambda$.

II. Si L es un lenguaje recursivo, entonces también lo es P(L). Para establecerlo demostraremos que existe una máquina de Turing M• que se detiene para cada entrada y que reconoce a P(L). Puesto

que L es recursivo existe una máquina de Turing M° que se detiene para cada entrada con L(M°) = L. La máquina M• está definida para operar como sigue: cuando se le aplica una entrada z ésta, a su vez, se aplica internamente a un módulo de factorización F' que, en condiciones similares al F, obtiene un par (u,v) con z = uv cada vez que, con excepción de la primera, se le envía una señal de factorización. Cuando se le suministra z obtiene automáticamente el primer par. Si recibe una señal de factorización y ya ha realizado todas las posible factorizaciones detiene todo el proceso y rechaza la entrada. Cuando factoriza un par (u,v) proporciona como salida la palabra t = vu que se le aplica seguidamente a la máquina M°, si ésta acepta t, entonces M• acepta z, en otro caso, si rechaza t, se envía al módulo F' una señal de factorización para si es posible obtenga la siguiente factorización. Claramente, en estas condiciones, M• se detiene para cada entrada y L(M•) = P(L).

12. Sea Σ un alfabeto, para $x,y \in \Sigma^*$ se define la operación

$$\diamondsuit(x,y) = \{z / (\exists u,v,w)(x = uv \wedge y = vw \wedge z = uvw)\}.$$

Esta operación se extiende a lenguajes $L,L' \subseteq \Sigma^*$ de la manera habitual, esto es,

$$\diamondsuit(L,L') = \bigcup_{x\in L,\, y\in L'} \diamondsuit(x,y).$$

Si L y L' son lenguajes recursivos ¿lo es también $\diamondsuit(L,L')$?

Si L y L' son lenguajes recursivos, entonces también lo es $\diamondsuit(L,L')$. Para establecerlo demostraremos que existe una máquina de Turing M■ que se detiene para cada entrada y que reconoce a P(L). Puesto que L y L' son recursivos existen dos máquinas de Turing M y M' que se detienen para cada entrada y L(M) = L y L(M') = L'. La máquina M■ está definida para operar como sigue: cuando se le aplica una entrada z ésta se pasa a un módulo de factorización T que la factoriza en ternas (u,v,w) tales que z = uvw cada vez que recibe una señal de factorización, excepto en la primera ocasión que lo hace automáticamente, y proporciona como salida las palabras x = uv e y = vw. Si al recibir la señal de factorización no hay ninguna nueva terna que obtener detiene todo el proceso y rechaza la entrada z. La salida x se le aplica ahora a la máquina M si ésta rechaza se envía una señal de factorización a T, si acepta se le aplica la palabra y a M' si ésta rechaza se envía una señal de factorización a T, en otro caso, si acepta, se acepta la entrada z. Para factorizar, el módulo T (sin entrar en todos los detalles precisos) puede utilizar una cinta con tres sectores: el inferior para z y los otros dos para albergar, en cada uno una marca, $\checkmark_1$ y $\checkmark_2$, que permitan controlar donde se ubican los segmentos u, v y w. Así, por ejemplo, la marca del sector intermedio ($\checkmark_1$) puede señalar la celdilla donde termina u y la del sector superior ($\checkmark_2$) donde comienza w. Cada vez que recibe una señal de factorización mueve una posición, hacia la derecha, la marca del sector superior, salvo si ya ha sobrepasado el final de z en cuyo caso avanza, si es posible, una posición la marca del sector intermedio y sobre la celdilla contigua se coloca la marca del sector superior ($v = \lambda$). Inicialmente la marca intermedia se encuentra en la celdilla en blanco inmediatamente anterior al primer símbolo de z y la marca superior sobre el primer símbolo de z ($u = v = \lambda$ y $w = z$). En la última factorización la marca intermedia se encuentra sobre el último símbolo de z y la superior sobre la celdilla inmediatamente a la derecha ($u = z$ y $v = w = \lambda$). En el caso en que $z = \lambda$ sólo produce la factorización $u = v = w = \lambda$. Claramente, en estas condiciones, la máquina M■ se detiene para cada entrada y L(M■) = $\diamondsuit$(L,L').

13. Sea Σ un alfabeto y $L \subseteq \Sigma^*$. Sea $R \subseteq \Sigma^*$ un lenguaje recursivo. Se define la operación

$$P_R(L) = \{x \in L / (x \notin R) \wedge (x = x^r)\}.$$

I. Si L es un lenguaje recursivamente enumerable ¿lo es también $P_R(L)$?

II. Si L es un lenguaje recursivamente enumerable ¿lo es también L – $P_R(L)$?

Sea el lenguaje $\mathbb{I} = \{x \in \Sigma^* / x = x^r\}$. En primer lugar estableceremos que este lenguaje es recursivo; para este fin definiremos una máquina de Turing $M_{\mathbb{I}}$ que se detenga para cada entrada y que lo reconozca. La máquina $M_{\mathbb{I}}$ tiene dos cintas y opera como sigue: cuando recibe una entrada x la recorre copiándola en la otra cinta, luego ubica el cabezal de una de las cintas al principio de la palabra (y lo desplaza de izquierda a derecha) y en la otra al final (y lo desplaza de derecha a izquierda) comenzando a comparar los símbolos correspondientes que encuentra cuando recorre las dos cintas. Si en este proceso encuentran los cabezales un blanco acepta, si en algún momento los símbolos leídos por los dos cabezales difieren rechaza.
En lo sigue, y como ya es habitual, el complementario de un lenguaje L lo representaremos como L^c.

I. Si L es un lenguaje recursivamente enumerable, entonces $P_R(L)$ también lo es. Sea $W = R^c \cap \mathbb{I}$, donde R^c, (por lo dicho) el complementario de R, es un lenguaje recursivo por ser el complemento una operación de cierre en la familia lenguajes recursivos; así W es también un lenguaje recursivo por ser la intersección otra operación de cierre. Basta ahora ver, inmediatamente a partir del enunciado, que $P_R(L) = L \cap W$. Puesto que W por ser recursivo es también recursivamente enumerable y ser la intersección una operación de cierre en los lenguajes recursivamente enumerables se sigue que $P_R(L)$ es un lenguaje recursivamente enumerable.

II. Si L es un lenguaje recursivamente enumerable, entonces también lo es L – $P_R(L)$. Basta notar que, a partir de lo anterior, $L - P_R(L) = L - (L \cap W) = L \cap (L \cap W)^c = L \cap (L^c \cup W^c) = (L \cap L^c) \cup (L \cap W^c) = L \cap W^c$. (Nótese que W^c es un lenguaje recursivo.)

14. Sean L_1, L_2, L_3 lenguajes. Se define la operación $\nabla(L_1, L_2, L_3)$ que define el lenguaje formado por las palabras que pertenecen al menos a dos de sus argumentos.

I. ¿Es la operación ∇ una operación de cierre en la familia de los lenguajes recursivos?
II. ¿Es la operación ∇ una operación de cierre en la familia de los lenguajes recursivamente enumerables?

A partir de la definición pude verse que $\nabla(L_1, L_2, L_3) = (L_1 \cap L_2) \cup (L_1 \cap L_3) \cup (L_2 \cap L_3)$. Puesto que la intersección y la unión son operaciones de cierre tanto en los lenguajes recursivos como en los lenguajes recursivamente enumerables se tiene que la operación ∇ es de cierre en ambas familias de lenguajes.

15. Sea Σ un alfabeto y sea < una relación de orden canónico definida sobre Σ^*. Sean dos lenguajes $L, L' \subseteq \Sigma^*$, se define la operación $\mu(L,L')$ del modo que sigue:

$$\mu(L,L') = \{x \in L / (\exists y \in L')(y \leq x)\}.$$

1) Si L y L' son lenguajes recursivamente enumerables, entonces ¿lo es también $\mu(L,L')$?
2) Si L y L' son lenguajes recursivos, entonces ¿lo es también $\mu(L,L')$?

Veamos que la operación binaria μ es de cierre en ambos casos. Lo estableceremos mediante dos caminos diferentes: uno, utilizando propiedades de cierre y el otro, asociando las correspondientes máquinas de Turing.

Utilizando las propiedades de cierre procedemos como sigue. Si L' = ∅, entonces $\mu(L,L') = \emptyset$, que es tanto un lenguaje recursivo como recursivamente enumerable. Si L' ≠ ∅, entonces sea $\min_{\leq}(L')$ el mínimo elemento de L' de acuerdo al orden canónico dado <. Sea $\Theta = \{x \in \Sigma^* / x < \min_{\leq}(L')\}$ que claramente es un lenguaje finito, en consecuencia es también un lenguaje recursivo. Así su complementario $\Theta^c = \Sigma^* - \Theta = \{x \in \Sigma^* / \min_{\leq}(L') \leq x\}$ también es recursivo y, por tanto, también recursivamente enumerable. Ahora puede verse que $\mu(L,L') = L \cap \Theta^c$, puesto que

$$x \in \mu(L,L') \Leftrightarrow \neg(x \in \neg L \wedge \neg \min_{\leq}(L') \leq x) \Leftrightarrow x \in L \cap \Theta^c ,$$

en consecuencia puesto que los lenguajes recursivos, y también los recursivamente enumerables, son cerrados para la intersección se tiene que la operación $\mu(L,L')$ es de cierre tanto en la clase de los lenguajes reursivos como en la clase de los lenguajes recursivamente enumerables.

Otra manera de resolver la cuestión es definir las máquinas de Turing correspondientes en cada caso.

- Para el caso #.1 sea M una máquina de Turing tal que L(M) = L, y sea M' un generador de Turing tal que G(M') = L'. Definimos seguidamente una máquina de Turing M_μ tal que $L(M_\mu) = \mu(L,L')$. Esta máquina opera como sigue. Cuando se le suministra una palabra de entrada x ésta se aplica a la máquina M y sólo en el caso de que ésta acepte se continúa arrancando el generador M' que estará controlado mediante una señal de continuación. Este generador cuando se arranca comienza el proceso generador y cuando genera la primera palabra se suspende hasta que recibe la señal de continuación en cuyo momento continuará el proceso generativo y seguirá así operando, mientras sea necesario, de acuerdo a este esquema. Cada vez que se genera una palabra y se comprueba si o no y ≤ x; si es que **sí** se termina el proceso aceptando x, si es que **no** se envía al generador la señal de continuación. El test y ≤ x se lleva a cabo en un módulo que siempre responde **si** o **no** y opera como sigue: se examina en primer lugar si y = x si son iguales, entonces se retorna **sí**, en otro caso se arranca un generador canónico de Σ* asociado al orden < y se ve qué palabra aparece primero, si lo hace y, entonces y < x y se retorna **sí**; en otro caso, si primero aparece x, entonces y > x y se retorna **no**. En consecuencia la máquina M_μ así definida cumple que $L(M_\mu) = \mu(L,L')$ y, por tanto, $\mu(L,L')$ es un lenguaje recursivamente enumerable.

- Para el caso #.2 sean A y A' dos máquinas de Turing que se detienen para cada entrada y que además cumplen que L(A) = L y L(A') = L'. Definimos seguidamente una máquina de Turing A_μ que se detiene para cada entrada y tal que $L(A_\mu) = \mu(L,L')$. Esta máquina opera como sigue. Cuando se le suministra una palabra de entrada x ésta se aplica a la máquina A, si ésta rechaza la palabra x se rechaza, y si ésta acepta se continúa arrancando un generador canónico de Σ* con las mismas características que en el caso anterior. Cada vez que genera una palabra y se aplica a A', si esta máquina acepta y entonces se acepta x, si rechaza e y ≠ x se envía al generador la señal de continuación; si y = x entonces se termina rechazando x.

16. Se define la operación $P:\{a,b\}^* \longrightarrow \wp(\{a,b\}^*)$ del modo que sigue:

I. $P(\lambda) = \{\lambda\}$
II. $P(x) = a^*x$, si $x = by$
III. $P(x) = b^*x$, si $x = ay$

¿Es la operación P de cierre dentro de la clase de los lenguajes recursivos?

Veamos que la operación P es de cierre en la clase de los lenguajes recursivos. Sea Σ = {a,b}. Un modo inmediato de hacerlo es observar que $P(L) = (L \cap \{\lambda\}) \cup \{a\}^*(L \cap \{b\}\Sigma^*) \cup \{b\}^*(L \cap \{a\}\Sigma^*)$ y que al ser $\{\lambda\}$, $\{b\}\Sigma^*$, $\{a\}\Sigma^*$, $\{a\}^*$ y $\{b\}^*$ lenguajes recursivos (de hecho son regulares y todos los lenguajes regulares son lenguajes recursivos, ya que cada

autómata finito determinista puede simularse, de modo directo, mediante una máquina de Turing cuyo cabezal nunca retrocede y que en consecuencia se detiene para cada entrada), y ser éstos cerrados para la unión, la intersección y la concatenación resulta que $P(L)$ también es recursivo.

17. Dado un lenguaje L se define el lenguaje

$$\mu(L) = \{x \,/ (x = x^r) \wedge (x \in \text{prefijos}(L))\}.$$

Si L es recursivamente enumerable, entonces ¿lo es también $\mu(L)$?

Si el lenguaje L es recursivamente enumerable, entonces también lo es $\mu(L)$. Para demostrarlo describimos una máquina de Turing con varias cintas M_L que lo reconoce. Puesto que L es recursivamente enumerable existe un generador de Turing G_L que lo genera y que se encuentra controlado durante la generación, en el sentido habitual, mediante una señal de control, de continuación, que le permite generar las palabras de una en una, con las correspondientes pausas, de acuerdo a esta señal. La máquina M_L opera como sigue. Cuando se le aplica una entrada x comprueba primeramente que $x = x^r$, para ello copia la palabra x en una cinta auxiliar y vuelve al principio de x en la cinta de entrada recorriendo seguidamente x y x^r síncronamente en sentidos contrarios, comprobando que los respectivos símbolos examinados son iguales; si en ambas cintas encuentra simultáneamente un blanco, y sólo en estas circunstancias, da por satisfecha la condición, esto es: $x = x^r$. Seguidamente, si esta condición se cumple, arranca el generador G_L que comienza la generación; siempre que genera una palabra se suspende hasta que recibe la señal de continuación; con la última palabra generada z comprueba que x es un prefijo de la misma, para ello escribe z en una cinta auxiliar y recorre desde el principio síncronamente x y z comprobando que los símbolos examinados coindicen; mientras esta coincidencia se dé continúa la comprobación que finaliza con la aceptación si se encuentra un blanco al examinar x (tanto si en z aparece un blanco como si no), en otro caso, si esta coincidencia de símbolos no se produce, se envía la señal de control al generador volviéndose a repetir esta fase.