Repaso Practica 2 Ejercicios

Ejercicio 1.

Probar que el lenguaje LU = {(<M>, w) | M acepta w} pertenece a la clase RE.

Idea General

La idea general es construir una MT MLU que acepte el lenguaje LU. Esta MT recibirá como entrada el par (<M>, w), siendo <M> una codificación de Maquina de Turing y w una entrada. MT MLU verificara que sea un par valido, si no lo es entonces rechaza. Si es un par valido entonces simulara la ejecución de <M> sobre w. Si <M> acepta, entonces MLU aceptara, si <M> rechaza entonces MLU rechazara y si <M> loopea entonces MLU loopeara.

Construcción

- MLU tiene 2 cintas.
- Dada la entrada (<M>,w), MLU hace lo siguiente:
  1) Verifica que (<M>,w) sea un par valido. Si no es un par valido MLU para en qR.
  2) Copia <M> en la cinta 2 y a medida que se copia se va borrando el contenido de la cinta 1 (se escribe B). Luego de esto se deja el cabezal posicionado en la cinta 1 sobre el primer símbolo de w.
  3) Ejecuta <M> sobre el w en la cinta 1. Si <M> para en qA entonces MLU parara en qA, si <M> para en qR entonces MLU parara en qR y si <M> loopea entonces MLU loopeara.

Verificación de correctitud

- LU ∈ RE

Dado que L(MLU) = LU, existe una maquina que acepta LU (la que se acaba de construir). Por lo tanto LU ∈ RE, ya que existe una máquina que lo acepta (y no necesariamente para).

Ejercicio 2.

Justificar o responder según el caso:

a. Se puede decidir si una MT M, a partir de la cadena vacía λ, escribe alguna vez un símbolo no blanco. Ayuda: ¿Cuántos pasos puede hacer M antes de entrar en loop?

Si se puede.

Siendo Q el conjunto de estados de M. Cada estado va a tener como máximo una transición desde el símbolo blanco (como la cadena es vacía (λ) no va a haber nada escrito en la cinta, entonces le vamos a dar solo importancia a las transiciones que parten desde blanco).

Se puede construir una MT que simula la ejecución de M y cuenta la cantidad de pasos que esta realiza. Si la cantidad de pasos realizados es mayor a |Q| y no se escribió nada en

la cinta, entonces M no escribe alguna vez un símbolo no blanco partiendo de la cadena vacia. Puede que exista un estado en donde partiendo desde el blanco se escribe algo, pero ocurre que no se puede llegar a ese estado si se parte de la cadena vacía.

La maquina antes de entrar en loop puede hacer |Q| pasos.

b. Se puede decidir si a partir de una cadena w, una MT M que sólo se mueve a la derecha se detiene. Ayuda: ¿Cuántos pasos puede hacer M antes de entrar en loop?

Si se puede decidir. M puede hacer |w| + |Q| pasos antes de entrar en loop. Esto se debe a que una vez que se termina de leer w, se comienza a leer solo blanco. Como siempre se va a leer el mismo símbolo y se va a mover a la derecha y cada estado va a tener a lo sumo una transición desde el blanco, si ya se hicieron |Q| transiciones luego de que se comenzó a leer blanco y se continúa ejecutando, entonces se están repitiendo configuraciones, por lo que se comenzó a loopear.

c. Se puede decidir, dada una MT M, si existe una cadena w a partir de la cual M se detiene en a lo sumo 10 pasos. Ayuda: ¿Hasta qué tamaño de cadenas hay que chequear?

Si, se puede decidir. Para eso se ejecutan a lo sumo 10 pasos de la MT M sobre cadenas con longitud de a lo sumo 10. Si son de longitud mayor entonces tan solo recorrer la cadena implica más de 10 pasos. Si M no se detiene en esos 10 pasos, entonces no existe cadena.

d. Intuitivamente, ¿se puede decidir, dada una MT M, si existe una cadena w de a lo sumo 10 símbolos a partir de la cual M para?

No, no se puede decidir, no hay forma de saber cuándo la maquina comienza a loopear.

Ejercicio 3.

Considerando la reducción de HP a LU descripta en clase, responder:

a. Explicar por qué la función identidad, es decir la función que a toda cadena le asigna la misma cadena, no es una reducción de HP a LU.

HP = { (<M>, w) | M para sobre w}

LU = { (<M>, w) | M acepta w}

La función de identidad no es un reducción de HP a LU debido a que esta mapea cada cadena a si misma, eso hace que en el caso en que una MT M para sobre w rechazándolo el par (<M>, w) si pertenece a HP y se va a mapear a si mismo pero este (<M>, w) no pertenece a LU (pero si a HP), por lo que no se cumple la definición de reducción:

- a partir de todo w ∈ HP, entonces f(w) ∈ LU
- a partir de todo w ∉ HP, entonces f(w) ∉ LU.

b. Explicar por qué las MT M ´ generadas en los pares (<M>, w), o bien se detienen aceptando, o bien loopean.

Esto se hace para que todos los pares (<M>, w) que pertenecen a HP, pertenezcan a LU y todos los pares (<M>, w) que no pertenecen a HP, no pertenezcan a LU. Esto se ve bien en el caso de rechazo, en donde el par (<M>, w) pertenece a HP puesto a que M se detiene sobre w, si esto se dejara igual cuando se aplica la función de reducción entonces ese par no pertenecería a LU (porque M no acepta w), por lo que para pertenecer M tiene que aceptar w. En el caso en que M loopea sobre w el par (<M>, w) no pertenece a ninguno de los dos lenguajes, por lo que se deja igual. Por lo tanto las MT M ´ generadas en los pares (<M>, w), o bien se detienen aceptando, o bien loopean.

c. Explicar por qué la función utilizada para reducir HP a LU también sirve para reducir HP^C a LU^C.

La función utilizada para reducir HP a LU también sirve para reducir HP^C a LU^C puesto a que cumple que lo que no pertenece a HP (por lo tanto pertenece a HP^C) no pertenece a LU (por lo tanto pertenece a LU^C) y lo que si pertenece a HP (por lo tanto no pertenece a HP^C) pertenece a LU (por lo tanto no pertenece a LU^C).

d. Explicar por qué la función utilizada para reducir HP a LU no sirve para reducir LU a HP.

La función utilizada para reducir HP a LU no sirve para reducir LU a HP, puesto que en los casos en donde se tiene un par (<M>, w) en el cual M rechaza w (M para en qR) dicho par no va a pertenecer a LU y como la función de reducción cambia los estados qR de M por qA, el par f((<M>, w)) si va a pertenecer a HP (se cumple que M para sobre w). No se cumple la definición de reducción mencionada en el punto a.

e. Explicar por qué la siguiente MT Mf no computa una reducción de HP a LU: dada una cadena válida (<M>, w), Mf ejecuta M sobre w, si M acepta entonces genera el par de salida (<M>, w), y si M rechaza entonces genera la cadena 1.

No computa una reducción de HP a LU puesto que al Mf estar ejecutando M sobre w, este M puede loopear, haciendo que la Mf en algunos casos no se detenga (no realiza una cantidad finita de pasos), y por lo tanto la función no es computable, por lo tanto no es una función de reducción valida. Para que una función sea computable tiene que existir una Mf que la computa y siempre se detiene.

A su vez, en el caso en que M rechaza, se genera la cadena 1, que no pertenece a LU (es una par invalido), pero el par (<M>, w) del que se partió si pertenecía a HP, no se cumple la definición de reducción.

Ejercicio 4.

Sea el lenguaje DHP = {wi | Mi para desde wi, según el orden canónico}. Encontrar una reducción de DHP a HP.

HP = { (<M>, w) | M para sobre w}

DHP = {wi | Mi para desde wi, según el orden canónico}.

Se quiere probar DHP <= HP.

Definición de la reducción

f(wi) = (<Mi>, wi)

Computabilidad

Existe una Mf que computa f: calcula cual es el indice i generando todas las cadenas en orden canónico e incrementando un contador hasta que la cadena generada coincida con la recibida. Una vez que se sabe el índice i, comienza a generar todas las MT en orden canónico hasta encontrar la MT Mi. De esta manera se obtiene el par (<Mi>, wi).

Correctitud

wi ∈ DHP → Mi para desde wi según el orden canónico → Mf genera (<Mi>, wi) donde Mi para sobre wi → (<Mi>, wi) ∈ HP

wi ∉ DHP → Mi no para desde wi según el orden canónico → Mf genera (<Mi>, wi) donde Mi no para sobre wi → (<Mi>, wi) ∉ HP

Ejercicio 5.

Sean TAUT y NOSAT los lenguajes de las fórmulas booleanas sin cuantificadores, respectivamente, tautológicas (satisfactibles por todas las asignaciones de valores de verdad), e insatisfactibles (ninguna asignación de valores de verdad las satisface). Encontrar una reducción de TAUT a NOSAT.

Definicion de la reduccion

f(ϕ) = ~(ϕ) siendo esta la negación de ϕ

Computabilidad

Existe una Mf que computa f: al comienzo de la cadena agrega un símbolo de negación y un paréntesis de abertura y al final de la cadena agrega un paréntesis de cierre, negando así la cadena.

Correctitud

ϕ ∈ TAUT → ϕ es una tautológica → ~(ϕ) al ser la negación de una tautología es una contradicción → ~(ϕ) ∈ NOSAT

ϕ ∉ TAUT → ϕ es una formula booleana valida: ϕ no es una tautología → ~(ϕ) al no ser la negación de una tautología no es una contradicción → ~(ϕ) ∉ NOSAT

ϕ es formula booleana invalida: ~(ϕ) es una formula booleana invalida→ ~(ϕ) ∉ NOSAT

Ejercicio 6.

Un autómata linealmente acotado (ALA) es una MT con una sola cinta, con la restricción de que su cabezal sólo puede moverse a lo largo de las celdas ocupadas por la cadena de entrada. Probar que el lenguaje aceptado por un ALA es recursivo. Ayuda: ¿en cuántos pasos se puede detectar que el ALA entra en loop?

Un lenguaje aceptado por un ALA es recursivo puesto que se puede construir una MT equivalente que detecta cuando el ALA entra en loop. Dado a que solo se puede mover a lo largo de las celdas ocupadas por la cadena de entrada, antes de entrar a un loop realizara:

$|w|. |Q|. |\Sigma|^{|w|}$ pasos.

Por lo que fácilmente se puede construir una MT ALA' que cumpla con todas las restricciones de ALA, solo que además lleva la cuenta de los pasos que se van realizando, permitiendo que pare en qR una vez que detecta que se entró en un loop.

Ejercicio 7.

Construir una MT que genere todos los índices i tales que (<Mi>, wi) ∈ HP, según el orden canónico.

Solución

1. MT GHP hace k := 0.
2. Ejecuta a lo sumo k pasos de todos los Mi generados en orden canónico hasta el momento sobre los correspondientes wi hasta llegar al Mk y wk.
3. Si Mi para sobre wi entonces MT GHP imprime el i.
4. Se borra la salida (para respetar el orden canónico), se hace k := k + 1 y se vuelve al paso 2.