MA1101-1 Introducción al Álgebra
Profesor: Leonardo Sánchez C.
Auxiliar: Marcelo Navarro

Pauta P5 - Auxiliar 7: Relaciones

12 de Mayo de 2018

P5. [**Aplicación de Relaciones: Teoria de la Computación**] El objetivo de esta pregunta es ver una de las aplicaciones de las clases de equivalencia en el campo de la teoría de la computación, particularmente en autómatas finitos y maquinas de Turing.

Definición 1.1 (Símbolo). *Se define un símbolo como un carácter cualquiera. Por ejemplo:* $a, 1, \heartsuit$, *etc.*

Definición 1.2 (String). *Se define un string o palabra como una secuencia de símbolos o bien, un símbolo. Por ejemplo:* $a, b, hola, aabba, 00010, 101, \heartsuit\heartsuit 0a$, *etc. También se define el string vació 'ε', como aquel string que simplemente no hace nada a dentro de una secuencia. Ejemplo:* $'hola\varepsilon' = 'hola'$.

Definición 1.3 (Concatenación). *Dados dos strings* $'a'$ *y* $'b'$, *se define la concatenación entre* $'a'$ *y* $'b'$ *como* $'a \cdot b'$ *o bien* $'ab'$, *donde simplemente se ha puesto al final de* $'a'$, *el string* $'b'$. *Por ejemplo: la concatenación entre* $'hol'$ *y* $'a'$ *es* $'hola'$.

Definición 1.4 (Alfabeto). *Se define un alfabeto como un conjunto de símbolos, lo denotaremos por* Σ.

Definición 1.5. *Se define* Σ^* *como el conjunto de todas las palabras o strings que se pueda formar a partir de los símbolos en* Σ.
Por ejemplo si $\Sigma = \{a, b\}$ *entonces* $\Sigma^* = \{\varepsilon, a, b, aa, bb, ab, ba, aaa, \dots, aababbabb, \dots\}$.
Este conjunto se define recursivamente por.

- $\varepsilon \in \Sigma^*$
- $w \in \Sigma^*$ *y* $a \in \Sigma \Rightarrow wa \in \Sigma^*$

Definición 1.6 (Lenguaje). *Definimos un Lenguaje* L, *como un subconjunto de* Σ^*, *es decir,* $\emptyset \subseteq L \subseteq \Sigma^*$.

Sea el alfabeto $\Sigma = \{0, 1\}$ y sea $L \subseteq \Sigma^*$ un lenguaje cualquiera. se define la relación $\equiv_L$ sobre Σ^* tal que

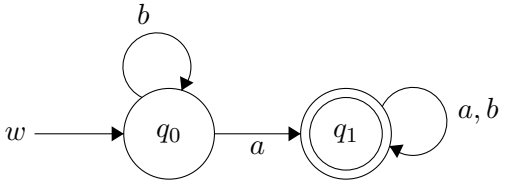
$$x \equiv_L y \iff [\forall z \in \Sigma^*, xz \in L \Leftrightarrow yz \in L]$$

Es decir, en $\equiv_L$ viven todas las parejas de palabras que les falta lo mismo para vivir en L.

a) Demuestre que $\equiv_L$ es una relación de equivalencia.

b) Considere ahora $L = \{w \in \Sigma^* : w$ contiene 001 como un substring $\}$. Calcule $\Sigma^*/\equiv_L$.

c) Dibuje una maquina de estados y transiciones que acepte al lenguaje L, donde una palabra w se ira leyendo de izquierda a derecha, caracter por caracter, y dependiendo del caracter leido se escoge a que estado ir de la maquina (esto lo definimos como una transición). La maquina termina de procesar cuando se termina el string, y se dice que la maquina acepta un string si la maquina termina en un estado de aceptación. Por ejemplo, la maquina para el lenguaje que contiene al menos un a es:



Donde el estado de aceptación es q_1.

Solución **P5**:

a)
- refleja: sea $x \in \Sigma^*$, se tiene que $\forall z \in \Sigma^*, xz \in L \Leftrightarrow xz \in L$ ya que $p \Leftrightarrow p$ es siempre verdadera, es decir $x \equiv_L x$.
- simetrica: sea $x, y \in \Sigma^*$ tal que $x \equiv y \iff \forall z \in \Sigma^*, xz \in L \Leftrightarrow yz \in L$. Esto ultimo es equivalente a que $yz \in L \Leftrightarrow xz \in L \iff y \equiv_L x$, ya que $p \Leftrightarrow q$ es equivalente a $q \Leftrightarrow p$. Como es equivalencia, en particular se tiene la implicancia. Por lo tanto es simetrica.
- Transitiva: sea $x, y, w \in \Sigma^*$ tal que $x \equiv_L y \wedge y \equiv_L w$ que equivale a que $\forall z \in \Sigma^*$ se tiene que $xz \in L \Leftrightarrow yz \in L$ y $yz \in L \Leftrightarrow wz \in L$. Luego se tiene que $xz \in L \Leftrightarrow wz \in L$ por transitivadd de la equivalencia, esto ultimo es que $x \equiv_L w$, por lo tanto es transitiva.

b) De acuerdo al enunciado, $x \equiv_L y$ significa que a x y a y le falta lo mismo para estar en L. Por lo que podemos pensar inmediatamente en la clase de equivalencia de las palabras que ya cumplen la condición de estar en L.

$$L_{[001]} = \{w \in \Sigma^* : w \text{ tiene 001 como un substring }\}$$

En este conjunto estan los elementos 001, 0001, 0...001, 1001, 11001101, etc. Ya que si comparo cualquier par de estos strings, ya estan en L por lo que directamente les falta lo mismo para estar en L (pues ya viven ahi).

Ahora nos falta ver las clases de equivalencia de las palabras que no estan en L. Como ya estamos en el caso de las palabras que no estan en L, podemos enfocarnos solo en las ultimos simbolos de los strings ya que solo dependen de la concatenación para llegar a estar en L.

Por ejemplo, notemos que $00 \equiv_L 1110100$ ya que ambas necesitan un 1 para pertenecer a L, esto se puede seguir haciendo y se puede concluir la siguiente clase de equivalencia.

$$L_{[00]} = \{w \in \Sigma^* : w \text{ termina en 00 y no tiene el substring 001}\}$$

Luego, siguiendo la misma lógica, podemos pensar en las palabras que terminan en 01 o en 10, luego de estudiar un poco, podemos notar que que 01 y 10 necesitan cosas distintas para terminar en L, por ejemplo 10 necesita un 01 para quedar en L pero 01 no le sirve a 01. por lo que son clases distintas. Entonces tenemos la clase de los que terminan en 0 y los que terminan en 1.

$$L_{[0]} = \{w \in \Sigma^* : w \text{ termina en 0 y su penultima letra no es 0 y no tiene el substring 001}\}$$

$$L_{[1]} = \{w \in \Sigma^* : w \text{ termina en 1 o } \varepsilon \text{ y no tiene el substring 001}\}$$

Notar que en la ultima clase se incluyo a las palabras vacías debido a que necesitan lo mismo que las palabras que terminan en 1.

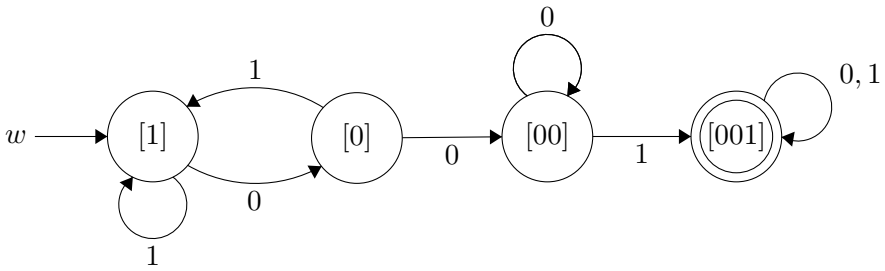
Queda de tarea comprobar que no hay más clases de equivalencias.

c) El string w entra a la maquina, la maquina irá leyendo desde izquierda a derecha cada carácter del string. Entra al estado [1], si lee un 1 (su primer carácter) se mantiene en [1] ya que el ultimo carácter visto es 1 y entra en la definición de las clases de equivalencia que vimos, si ve un 0 vamos a estar más cerca de estar en [001], por lo que avanzamos a [0].

Si estamos en [0] y el siguiente carácter es un 1 debemos retroceder ya que estaríamos en las mismas condiciones de al principio, es decir, el ultimo carácter es 1. En caso de ver un 0 entonces estamos más cerca de estar en [001] por lo que nos vamos al estado [00].

Si estamos en [00], solo nos falta un 1 para llegar al estado [001] por lo que nos iremos a este estado si leemos un 1. Por el contrario si leemos un 0 nos quedamos donde mismo pues aun tenemos 00 al final del string.

Finalmente, estando en el estado [001] que representa el estado de aceptación (si termina el string y el ultimo estado es este, la maquina ha aceptado al string), da lo mismo que simbolo venga, ya tenemos el substring 001 por lo que se quedará ahi por siempre.



Y con esto, hemos construido nuestro primer autómata finito determinista que da la base para estudiar la teoría de la computación y empezar a formalizar sobre que maquinas son más fuerte que otras y en particular, la maquina más fuerte y que jamás va a ser superada (y la cual tenemos hoy en nuestros computadores), la maquina de Turing.