Algoritmos, Azar y Autómatas
Resolución Ejercicios 7 y 8 (selección)

Manuel Panichelli

November 16, 2021

Resultados previos

Def. 1 (Selección por prefijos)**.** Sea $x = a_1 a_2 \dots$ una palabra infinita sobre el alfabeto A, y $L \subseteq A^*$ un conjunto de palabras finitas sobre ese alfabeto. Una **selección por prefijo** de x por L es

$$x \upharpoonright L = a_{i_1} a_{i_2} a_{i_3} \dots,$$

donde i_1, i_2, i_3 es una enumeración en orden creciente de todos los enteros i tales que $a_1 a_2 \dots a_{i-1} \in L$.

Ejemplos:

$x = 010010001000010000010000001000000001\dots$

- $L = (0^*1)^*$

  $x \upharpoonright L = 000000000\dots.$

- $\bar{L} = (A^* \setminus L) = (0^*1^*)^*0$

  $x \upharpoonright \bar{L} = 101010010001\dots$

Teorema 1 (Agafonov 1968)**.** Sea $x \in A^\omega$ normal, y $L \subset A^*$ regular. Luego $x \upharpoonright L$ es normal.

Prop. 1 (A la Champernowne)**.** La secuencia de la concatenación de todas las posibles palabras de longitud $1, 2, 3, \dots$ en orden lexicográfico para el alfabeto $\{0, 1\}$,

$$0\,1\quad 00\ 01\ 10\ 11\quad 000\ 001\ 010\ 011\ 100\ 101\ 110\ 111\quad 0000\dots$$

es normal.

Ejercicio 7

Dada una secuencia $a_1a_2\dots$ donde los a_i son símbolos del alfabeto, $pares(x)$ es la subsecuencia de x que se obtiene de tomar los símbolos en las posiciones pares de x. Es decir,

$$pares(x) = a_2a_4a_6\dots$$

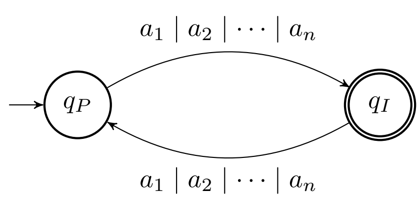
Demostrar que si x es una secuencia normal, entonces la subsecuencia $pares(x)$ es normal.

Proof. Usando el Teorema 1, nos basta con encontrar un lenguaje regular L tal que $x \upharpoonright L = pares(x)$ para probar que $pares(x)$ es regular.

Interpretando la definición de Selección por prefijos, el resultado de $x \upharpoonright L$ es la concatenación de los símbolos de las posiciones i-ésimas tales que el prefijo de longitud $i-1$ pertenece al lenguaje L.

Por lo tanto, para que $x \upharpoonright L = pares(x) = a_2a_4a_6\dots$, tienen que pertenecer a L los prefijos $a_1,\ a_1a_2a_3,\ a_1a_2a_3a_4a_5$, y así. Es decir, los de longitud impar. Entonces nos alcanza con dar un lenguaje regular para las cadenas del alfabeto A de longitud impar. El lenguaje es el aceptado por el autómata finito M, $L = \mathcal{L}(M)$. Suponiendo que $A = \{a_1, \dots a_n\}$,

$$M = \langle\{q_P, q_I\}, A, \delta, q_P, \{q_I\}\rangle, \text{ donde } \delta \text{ está dada por:}$$



Como encontré un autómata que reconoce el lenguaje L, L es regular. Y como $x \upharpoonright L = pares(x)$, por el Teorema 1, si x es normal, entonces $pares(x)$ también.

$\square$

Ejercicio 8

Indicar una forma de selección de subsecuencias tal que no siempre preserva normalidad. Dar un ejemplo de una secuencia normal donde no se preserve.

Para ello, mi estrategia será tomar la cadena normal de la Prop 1,

$$
\begin{aligned}
x = \;& 01 \\
& 00\ 01\ 10\ 11 \\
& 000\ 001\ 010\ 011\ 100\ 101\ 110\ 111 \\
& 0000\ 0001\ 0010\ 0011\ 0100\ 0101\ 0110\ 0111\ 1000\ 1001\ 1010\ 1011\ 1100\ 1101\ 1110\ 1111\ \ldots
\end{aligned}
$$

y mostrar que puedo dar un lenguaje L tal que $x \upharpoonright L$ no sea normal. Voy a llamar *bloque* a la concatenación de todas las cadenas de cierta longitud, por ej. 01 es el bloque de long 1 y 00011011 el de long 2. La idea de L entonces es que contenga los prefijos de las concatenaciones **completas** de todos los bloques que forman x, y de esa forma va a pertenecer el primer símbolo del siguiente bloque, que es siempre 0.

El lenguaje L será el aceptado por la máquina de Turing M expresada mediante un programa (en particular en Python). La función que lo computa es `recognize`,

```python
def recognize(word: str) -> bool:
    """
    Reconoce una cadena solo si es un prefijo que coincide con
    un fin de un "bloque" de una cadena "a la Champernowne".
    """

    n = 0
    current_champ = ""
    while len(current_champ) <= len(word):
        if word == current_champ:
            return True

        n += 1
        current_champ = champ_up_to(n)

    return False

def champ_up_to(n: int) -> str:
    """
    Genera la secuencia "a la Champernowne" hasta el bloque
    de tamano n.
    """

    seq = ""
    for n in range(1, n+1):
        seq += ''.join(all_words_of_length(n))

    return seq

```

```python
def all_words_of_length(n: int) -> List[str]:
    """
    Genera todas las cadenas de long n en orden lexicografico
    """

    if n == 1:
        return ["0", "1"]

    prev = all_words_of_length(n - 1)
    adding_zero = map(lambda word: word + "0", prev)
    adding_one = map(lambda word: word + "1", prev)
    return sorted(list(adding_zero) + list(adding_one))
```

Listing 1: Programa que reconoce las cadenas de L

La función `recognize` solamente retornará verdadero si la palabra es el prefijo que coincide con el fin de un bloque. Para ello sigue una estrategia muy simple, generar el prefijo y ver si son iguales. Esto es posible ya que la secuencia *a la Champernowne* es compresible por una máquina de Turing.

Como un prefijo estará contenido en el lenguaje solo si coincide con el fin de un bloque, el símbolo que le sigue será el inicio del próximo, que es siempre 0. Por lo tanto,

$$x \restriction L = 000000\ldots$$

$\Rightarrow x \restriction L$ no es normal.