# Teoría de Autómatas y Lenguajes Formales

# Practice 3: Turing Machine, recursive functions and WHILE language

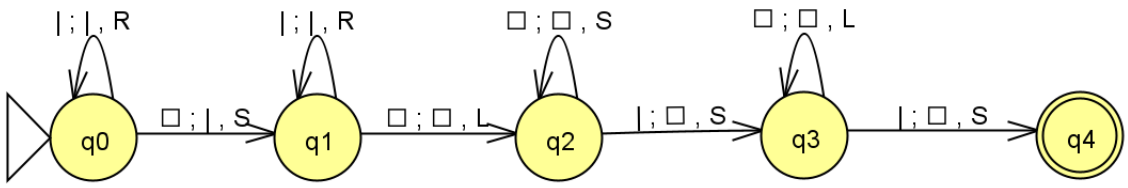
Pablo Fazio Arrabal

## Ejercicio 1 - Turing Machine

**Ejercicio 1.** Define the TM solution of **exercise 3.4** of the problem list and test its correct behaviour.

**Exercise 3.4.** Prove that the function $add(x, y) = x + y$, with $x$, $y \in \mathbb{N}$ is Turing-computable using the unary notation $\{|\}$. You have to create a TM with two arguments separated by a blank symbol that starts and ends behind the strings.

### Construcción de la TM en JFLAP

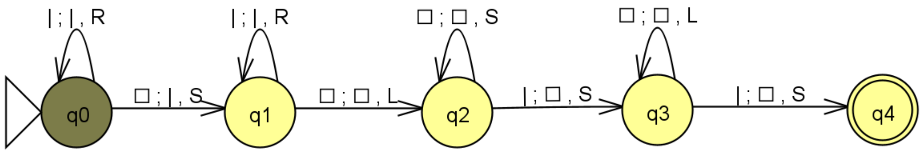


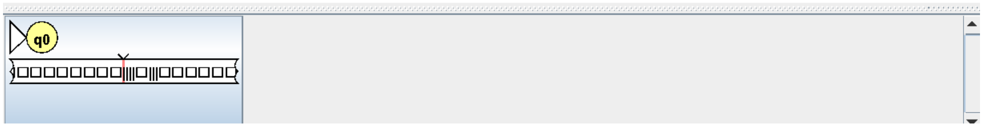

Lo más indicado es darnos cuenta que debemos unir ambas representaciones unarias de los números de la operación y borrar los | sobrantes. Para ello, como el cabezal de la cinta comienza al principio de la cadena, lo movemos al símbolo separador entre ambos números y en dicha posición escribimos '|'.

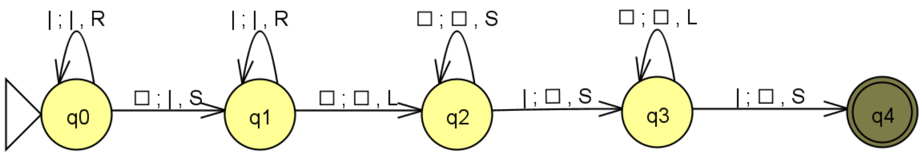
Una vez hecho esto, nos desplazamos hasta el final de la cadena esperando un nuevo símbolo por defecto y borramos los dos últimos símbolos de la cadena a la izquierda de este. Por último, paramos la computación y en la cinta estará representada la solución de la suma en unario.

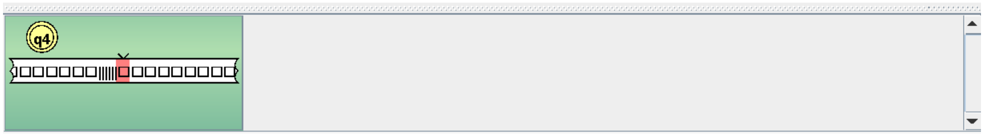
### Prueba de la TM en JFLAP

Veamos un ejemplo para comprobar que la función $add(x, y)$ está bien definida por esta TM. Dada la notación unaria $\{|\}$, sabemos que un natural $n$ está representado con $n + 1$ '|'. Luego veamos que ocurre con $add(3, 2) = 5$, representado por la cadena "||||□|||":




Como vemos en la imagen anterior, la cadena es aceptada por nuestra Máquina de Turing parando la cinta con la cadena "||||||", es decir la representación unaria del número 5, lo cual coincide con la solución.

## Ejercicio 2 - Recursive functions

**Ejercicio 2.** Define a recursive function for the sum of three values.

### Definición de la función recursiva

Vamos a definir nuestra función como una **recursión primitiva**, es decir:

**Definición.** Sea $k \geq 0$ y las funciones $g : \mathbb{N}^k \to \mathbb{N}$ y $h : \mathbb{N}^{k+2} \to \mathbb{N}$. Si la función $f : \mathbb{N}^{k+1} \to \mathbb{N}$ es:

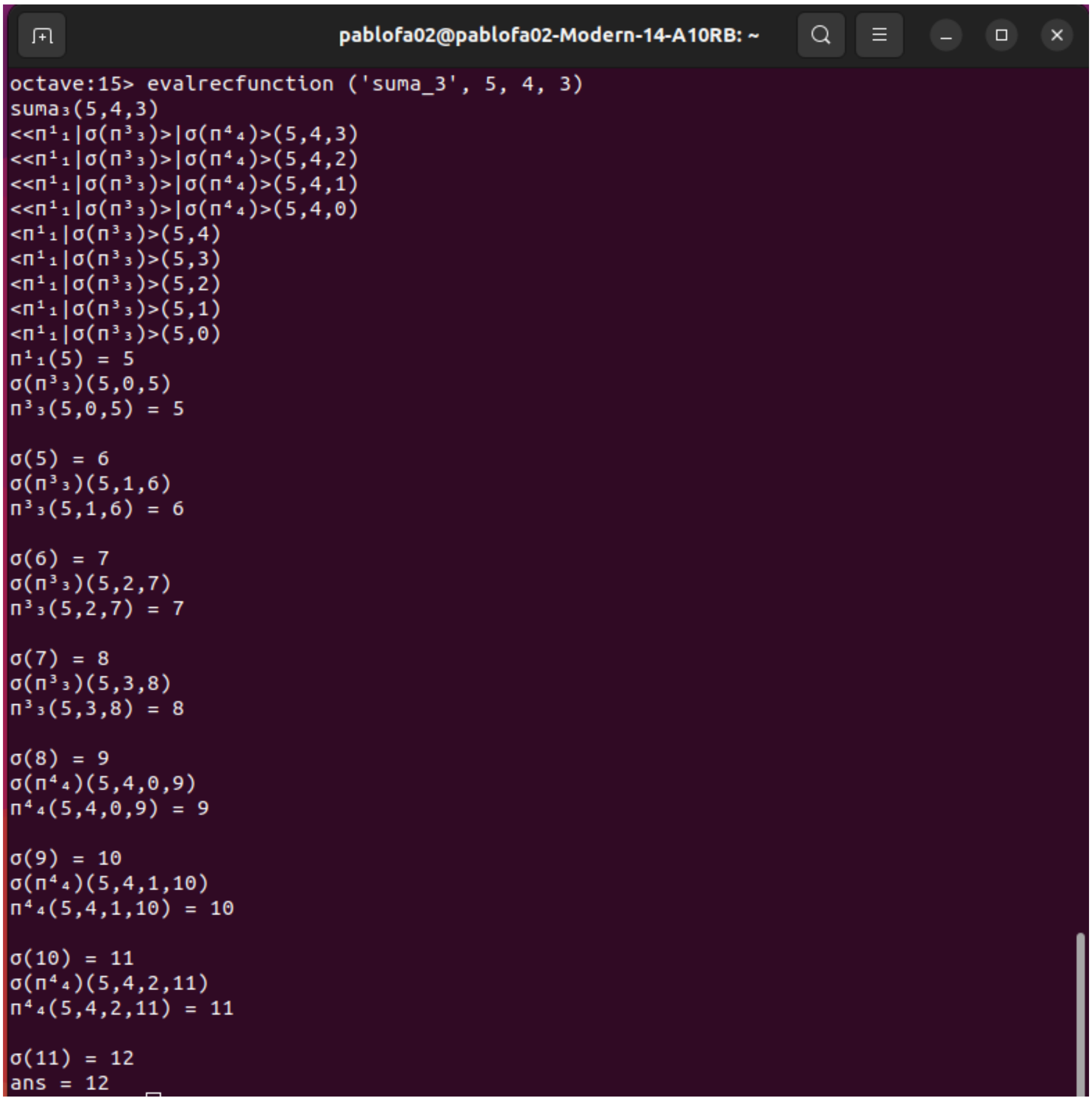
$$f(\underline{n}, m) = \begin{cases} g(\underline{n}) & \text{si} \quad m = 0 \\ h(\underline{n}, m-1, f(\underline{n}, m-1)) & \text{si} \quad m > 0 \end{cases}$$

entonces $f$ se obtiene de $g$ y $h$ por recursión primitiva.

Definamos la función $suma_3 : \mathbb{N}^3 \to \mathbb{N}$ como $suma_3(x, y, z) = x + y + z$. Por tanto, se definen $g : \mathbb{N}^2 \to \mathbb{N}$ y $h : \mathbb{N}^4 \to \mathbb{N}$, donde $g = add(x, y)$ y $h = \sigma(\pi_4^4)$ ambas funciones recursivas vistas en teoría.

Por lo visto en el **Ejemplo 8.1.1** de *talflecturenotes.pdf*, la función $add = \langle \pi_1^1 \mid \sigma(\pi_3^3) \rangle$ es recursiva. Luego, $suma_3(\underline{n}) = \langle \langle \pi_1^1 \mid \sigma(\pi_3^3) \rangle \mid \sigma(\pi_4^4) \rangle (\underline{n})$.

### Prueba en Octave



```
pablofa02@pablofa02-Modern-14-A10RB: ~
octave:15> evalrecfunction ('suma_3', 5, 4, 3)
suma₃(5,4,3)
<<π¹₁|σ(π³₃)>|σ(π⁴₄)>(5,4,3)
<<π¹₁|σ(π³₃)>|σ(π⁴₄)>(5,4,2)
<<π¹₁|σ(π³₃)>|σ(π⁴₄)>(5,4,1)
<<π¹₁|σ(π³₃)>|σ(π⁴₄)>(5,4,0)
<π¹₁|σ(π³₃)>(5,4)
<π¹₁|σ(π³₃)>(5,3)
<π¹₁|σ(π³₃)>(5,2)
<π¹₁|σ(π³₃)>(5,1)
<π¹₁|σ(π³₃)>(5,0)
π¹₁(5) = 5
σ(π³₃)(5,0,5)
π³₃(5,0,5) = 5

σ(5) = 6
σ(π³₃)(5,1,6)
π³₃(5,1,6) = 6

σ(6) = 7
σ(π³₃)(5,2,7)
π³₃(5,2,7) = 7

σ(7) = 8
σ(π³₃)(5,3,8)
π³₃(5,3,8) = 8

σ(8) = 9
σ(π⁴₄)(5,4,0,9)
π⁴₄(5,4,0,9) = 9

σ(9) = 10
σ(π⁴₄)(5,4,1,10)
π⁴₄(5,4,1,10) = 10

σ(10) = 11
σ(π⁴₄)(5,4,2,11)
π⁴₄(5,4,2,11) = 11

σ(11) = 12
ans = 12
```

# Ejercicio 3 - WHILE Language

**Ejercicio 3.** Implement a WHILE program that computes the sum of three values. You must use an auxiliary variable that accumulates the result of the sum.

## Construccion del código WHILE

$suma_3 = (3, s)$

---

$s$:
$X_4 := X_1$;
**while** $X_2 \neq 0$ **do**
    $X_4 := X_4 + 1$;
    $X_2 := X_2 - 1$;
**od**
**while** $X_3 \neq 0$ **do**
    $X_4 := X_4 + 1$;
    $X_3 := X_3 - 1$;
**od**

---

El código escrito es bastante sencillo de interpretar. Usaremos 3 argumentos de entrada en total, que representarán los números que queremos sumar, e iremos decrementando uno de ellos a favor de un argumento auxiliar $X_4$, que en un principio ya asignaremos como uno de estos números, hasta que se llegue a 0. Lo mismo haremos con el siguiente argumento, es decir, lo reducimos a favor del auxiliar hasta llegar a 0. Por tanto, al final, nos quedará el atributo $X_4$ como suma de los tres valores.