Def.

Automata A has 5 elements

1. n : Int — Dimension of A
2. I : Vector — Initial (has n elements)
3. A_0 : $n \times n$ Matric — Multiplier for Input = 0
4. A_1 : $n \times n$ Matric — Multiplier for Input = 1
5. F : Vector — Final (has n elements)

Automata คำนวณ Binary Input → เหลือแค่ : double

โดยใช้ ฟังชัน F, f

Function F : (Ffinite)

ถ้า Input $i = u_1 u_2 \dots u_n$ Binary , $u_i = \{0$ หรือ $1\}$

$$F_i = I \times A_i \times F$$

$$= I \times A_{u_1} \times \dots \times A_{u_n} \times F$$

$I \times [A_0 / A_n] \times [F] =$: double

(f_real)

$$f_i = \lim I \times A_i \times \underbrace{A_0 \dots}_{\infty} \times F$$

f_i คือค่าลิมิต เมื่อ $\times A_0$ ต่อท้ายไปเรื่อย

$$f_i = \begin{cases} \lim F_i & \text{if lim exists} \\ \text{undefined} & \text{if lim doesn't exist} \end{cases}$$

User stories

① Automata - Input

1) n: Int
2) I : double[n] — double , double , ... — , between Elements
3) A_0 : double[n][n] — double , double , ... ; — ; between row
   A_1 : double[n][n] — dobla , ...
   ...
4) F : double[n] — double , double , ...

② Computing : F , and then $f_{infinity}$

2.1 Compute Graph f for input u = 0000... (10x) - 111... (10x)

$f_{infinity}$ of u

u = Binary-String (from 0...0 (10) to 1...1 (10))

to Int (2^{10} points)

f := Calculation. f_read(String, Nat)

2.2 Extra Binary / String Input

Input
→ Binary → compute
→ String → to Binary → Compute

Compute : F_{Input} = ?

$f_{infinity}$ of F_{Input} = ?

③ Output

1. Show Graph F of Automata
2. Show F, for any String Input
   $f_{infinity}$

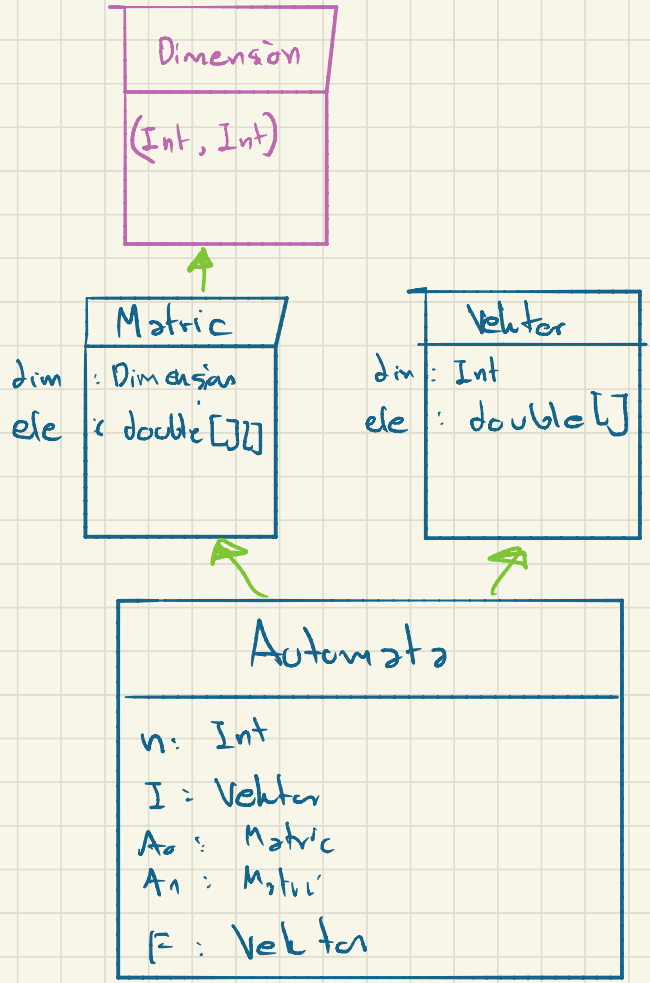
④ Check if Automata A minimal

1. find $B_1 B_2$ Orthogonal Basis for IA AF
2. Compute new A_L A_R
3. compare n of A_L / A_R with n of A
4. if minimal → final
   else → ⑤

:= Calculation. isMiniautomat

↓

minimize (Automata)



Calculation

: Vm M := Vektor x Matrix

: Mm M := Matrix x Matrix

: M m V := Matrix x Vektor

Input - Schemata

① For Vector

: , คั่นแต่ละ elements

exp. 1, 2, 3, 4

② For Matrix

, คั่นแต่ละ elements ใน row เดียวกัน

; คั่น row

exp. 1, 2, 3 ; 4, 5, 6 ; ...

③ String Input

if binary ⟶ compute

else ⟶ toBinary ⟶ compute