.

Back to Universal Programs

Alain Colmerauer

Sydney, September, 2008

Marseilles, France, http://www.alain.colmerauer.free.fr

Contents

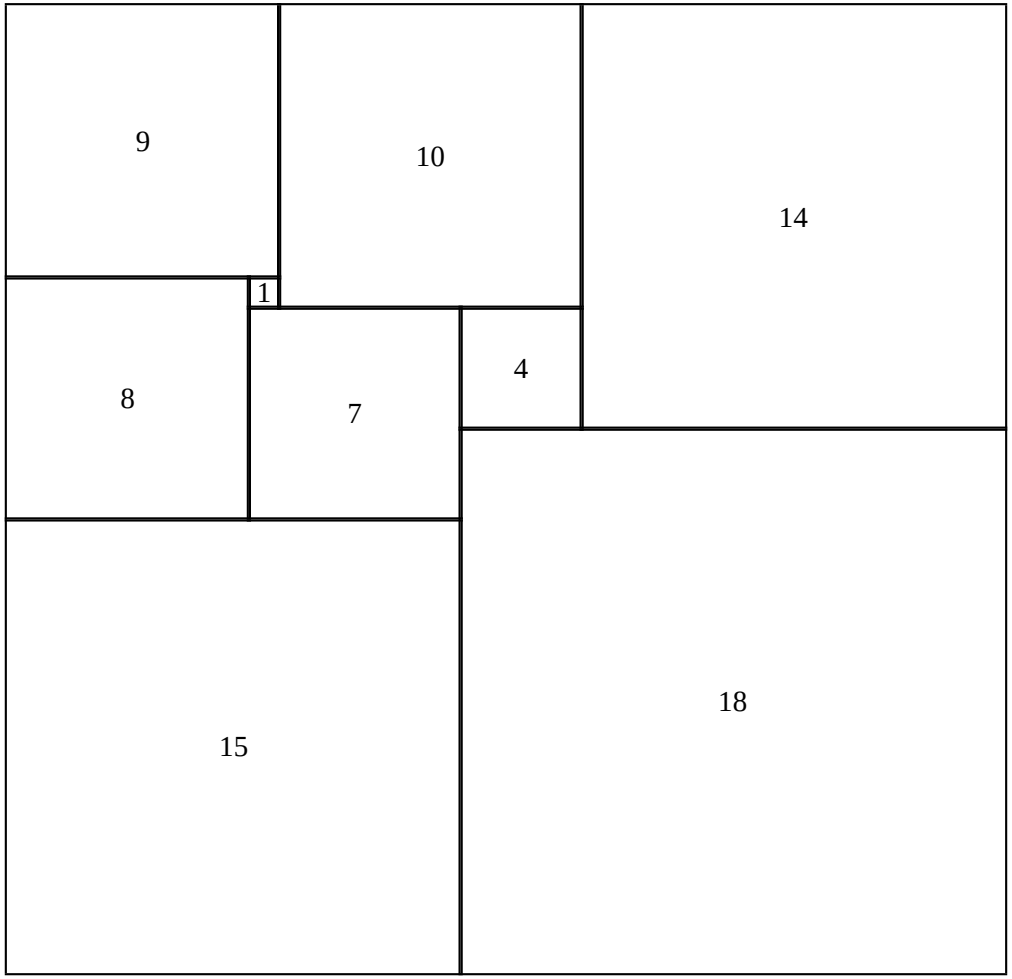
Complicated constraints 4
Decomposition of a rectangle into 9 squares 5
Decomposition of a rectangle into 9 squares, next 6
Decompostion into n squares with Prolog III or IV 7
Decomposition into 9 squares with Prolog III or IV, next 8
A complex constraint for Prolog IV 9
Optimal interval narrowing of the sortedness constraint 10
Optimal interval narrowing of the sortedness constraint $2n = 22$ 11
Optimal interval narrowing of the sortedness constraint $2n = 100$ . . . 12
Transition 13
What is a machine and a program? 14
A machine? 15
A machine programmed for translating 16
A machine programmed for multiplying 17
A programmed machine is a dynamic system 18
Formal definition of a programmed machine 19
How a programmed machine operates 20
Universal program and coding 21
Universal program for translating 22
Same universal program for multiplying 23
Definition of a universal pair 24
Universal program running on itself 25
Translating program 26
Universal program running on the translating program 27
Universal program running on the universal program 28
Definition of complexity and introspection coefficient 29
Existence and value of the introspection coefficient 30
Toward the labeling hypothesis 31
Labeling hypothesis 32
Vector $\boldsymbol{B}$ and Matrix $\boldsymbol{A}$ related to $\mathcal{B}$ and $\mathcal{A}$ 33
Vector $\boldsymbol{B}$ and Matrix $\boldsymbol{A}$ related to $\mathcal{B}$ and $\mathcal{A}$, next 34
Properties of $\boldsymbol{A}$ and $\boldsymbol{B}$ 35
Main theorem 36
Turing machine 37
Current configuration 38
Executed instruction 39
Next configuration 40
Executed instruction 41
Next configuration 42
Program example 43
Turing machine with internal direction 44
Program example 45
Graph of a program 46
Formal definition of a machine 47
Formal definition of a Turing machine (with internal direction) 48
Universal pair $(\boldsymbol{U}, \boldsymbol{code})$ for Turing machine (with internal direction) 49
Coding function $\boldsymbol{code}$ 50
Coding function $\boldsymbol{code}$, next 51
Coding function $\boldsymbol{code}$, next next 52
How program $\boldsymbol{U}$ operates 53
How program $\boldsymbol{U}$ operates, next 1 54
How program $\boldsymbol{U}$ operates, next 2 55
Architecture of program $\boldsymbol{U}$ 56
Graph of program $\boldsymbol{U}$ 57
Listing of program $\boldsymbol{U}$ 58
Listing of program $\boldsymbol{U}$, next 1 59
Listing of program $\boldsymbol{U}$, next 2 60
Complexity and introspection coefficient of $(\boldsymbol{U}, \boldsymbol{code})$ 61
Complexity of our Turing machine on examples 62
Introspection coefficient of our pair for the Turing machine 63
Arithmetic machine with indirect adressing 64
Iindirect adressing machine 65
Indirect addressing machine, formal definition 66
Universal pair $(\boldsymbol{U}, \boldsymbol{code})$ for arithmetic machine with indirect addressing . . . 67
Execution of $\boldsymbol{P}$ 68
Execution of $\boldsymbol{U}$ 68
Listing of program $\boldsymbol{U}$ 69
Listing of program $\boldsymbol{U}$, next 70

Complexity and introspection coefficient of $(\boldsymbol{U}, \boldsymbol{code})$ 71
Complexity of our pair for the indirect addressing machine 72
Introspection coefficient of our pair for the indirect addressing machine 73
Matrix $\boldsymbol{A}$ and vector $\boldsymbol{B}$ for $\boldsymbol{U}$ 74
Conclusion 75
Open problem 76
Open problem, next 77
A joke 78

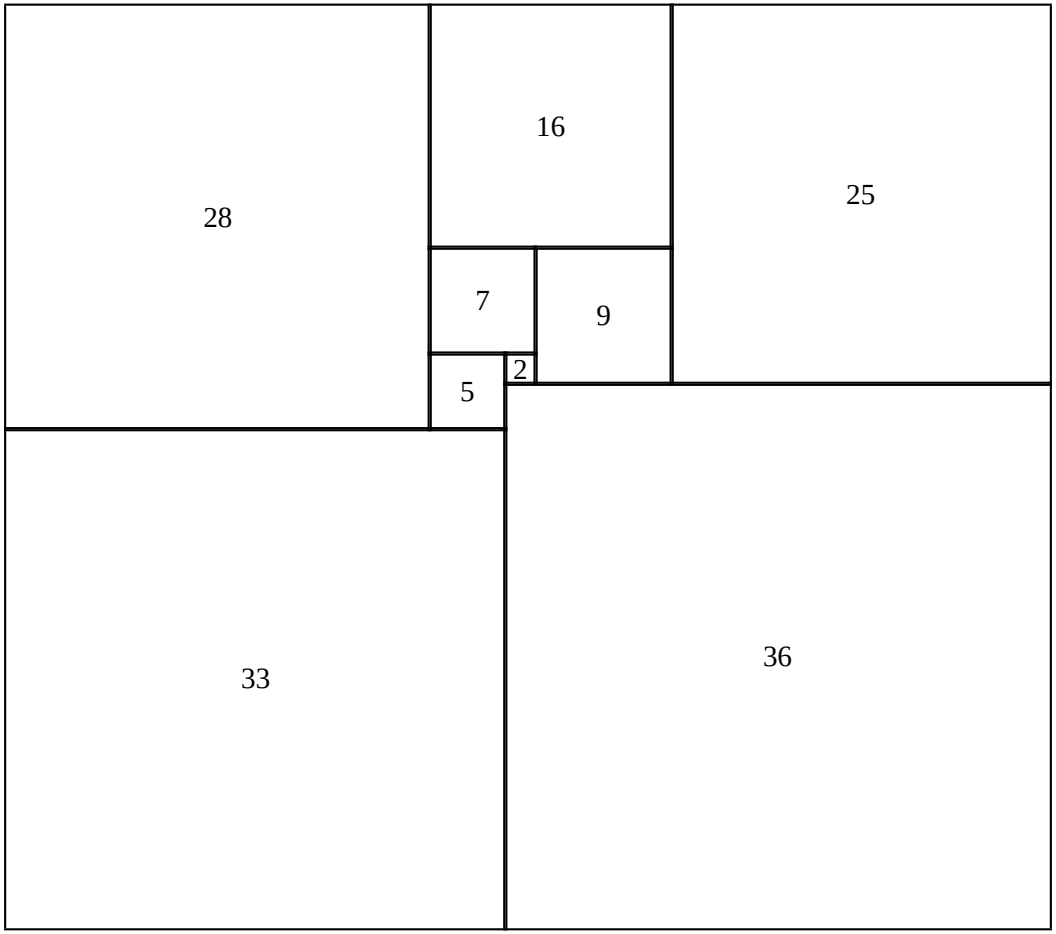
.

Complicated constraints

Decomposition of a rectangle into 9 squares



Decomposition of a rectangle into 9 squares, next



Alain Colmerauer, An Introduction to Prolog III, *Communications of the ACM*, 33(7): 68-90, 1990. A preliminary version is available in `http://alain.colmerauer@free.fr`.

Decompostion into n squares with Prolog III or IV

```
rectangle(A,C) :-
   gelin(A,1)
   distinctSizes(C),
   area([-1,A,1],L,C,[]).

distinctSizes([]).
distinctSizes([B|C]) :-
   gtlin(B,0),
   distinctSizes(C),
   out(B,C).

out(B,[]).
out(B,[Bp|C]) :-
   dif(B,Bp),
   out(B,C).
```

```
area([V|L],[V|L],C,C) :-
   gelin(V,0).
area([V|L],Lppp,[B|C],Cpp) :-
   ltin(V,0),
   square(B,L,Lp),
   area(Lp,Lpp,C,Cp),
   area([V+B,B|Lpp],Lppp,Cp,Cpp).

square(B,[H,0,Hp|L],Lp) :-
   gtlin(B,H),
   square(B,[H+Hp|L],Lp).
square(B,[H,V|L],[-B+V|L]) :-
   B=H.
square(B,[H|L],[-B,H-B|L]) :-
   ltlin(B,H).
```

$ltlin(x,y)$, $dif(x,y)$, $gelin(x,y)$, $gtlin(x,y)$ mean $x < y$, $x \neq y$, $x \geq y$, $x > y$.

Decomposition into 9 squares with Prolog III or IV, next

```
>> size(C)=9, rectangle(A,C).

A = 33/32,
C = [15/32,9/16,1/4,7/32,1/8,7/16,1/32,5/16,9/32];
A = 69/61,
C = [33/61,36/61,28/61,5/61,2/61,9/61,25/61,7/61,16/61];
A = 33/32,
C = [9/16,15/32,7/32,1/4,7/16,1/8,5/16,1/32,9/32];
A = 69/61,
C = [36/61,33/61,5/61,28/61,25/61,9/61,2/61,7/61,16/61];
A = 33/32,
C = [9/32,5/16,7/16,1/4,1/32,7/32,1/8,9/16,15/32];
A = 69/61,
C = [28/61,16/61,25/61,7/61,9/61,5/61,2/61,36/61,33/61];
A = 69/61,
C = [25/61,16/61,28/61,9/61,7/61,2/61,5/61,36/61,33/61];
A = 33/32,
C = [7/16,5/16,9/32,1/32,1/4,1/8,7/32,9/16,15/32].
```

$size(x)$ means *size of the list* x.

A complex constraint for Prolog IV

$$\exists u \exists v \exists w \exists x \left(\begin{array}{ll} & y \leq 5 \\ \wedge & v_1 = \cos v_4 \\ \wedge & \text{size}(u) = 3 \\ \wedge & \text{size}(v) = 10 \\ \wedge & u \bullet v = v \bullet w \\ \wedge & y \geq 2 + (3 \times x) \\ \wedge & x = (74 > \lfloor 100 \times v_1 \rfloor) \end{array} \right)$$

$y = 5$

```
>> U ex V ex W ex X ex
le(Y,5),
V:1 = cos(V:4),
size(U) = 3,
size(V) = 10,
U o V = V o W,
ge(Y,2.+.(3.*.X)),
X = bgt(74,floor(100.*.V:1)).

Y = 5.
```

Alain Colmerauer, *Prolog IV*, 1995, `http://alain.colmerauer@free.fr`, in French.

Optimal interval narrowing of the sortedness constraint

• **The sortedness constraint**

$$\textit{sort}(x_1, \ldots, x_n, x_{n+1}, \ldots, x_{2n}) \quad \equiv \quad \begin{cases} (x_{n+1}, \ldots, x_{2n}) \\ \text{is equal to} \\ (x_1, \ldots, x_n) \text{ sorted} \\ \text{in non-decreasing order.} \end{cases}$$

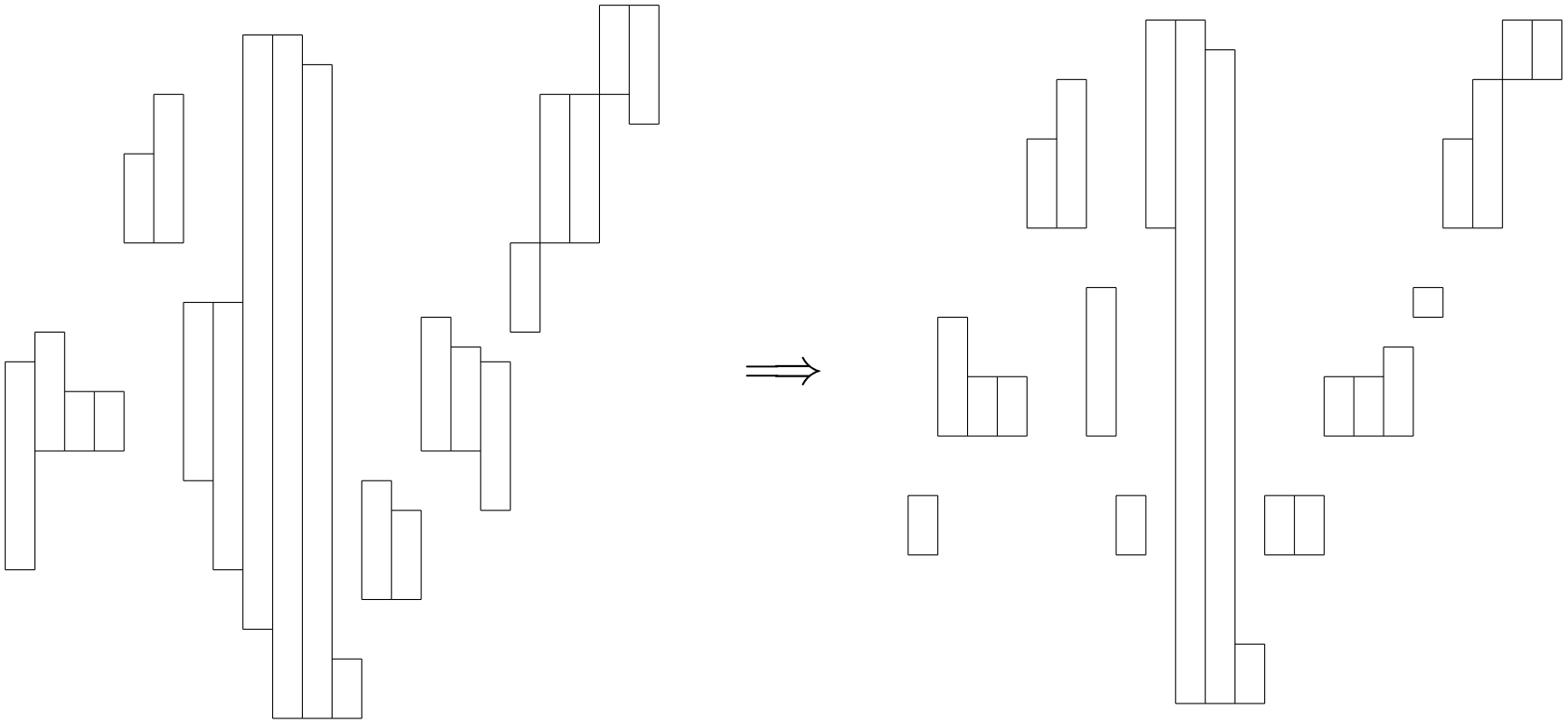
• **Problem** Given the intervals a_i, find the intervals a'_i, as small as possible, such that the two following constraints have the same solutions:

$$\textit{sort}(x_1, \ldots, x_{2n}) \ \wedge \ x_1 \in a_1 \ \wedge \ \cdots \ \wedge \ x_{2n} \in a_{2n}$$

$$\Leftrightarrow$$

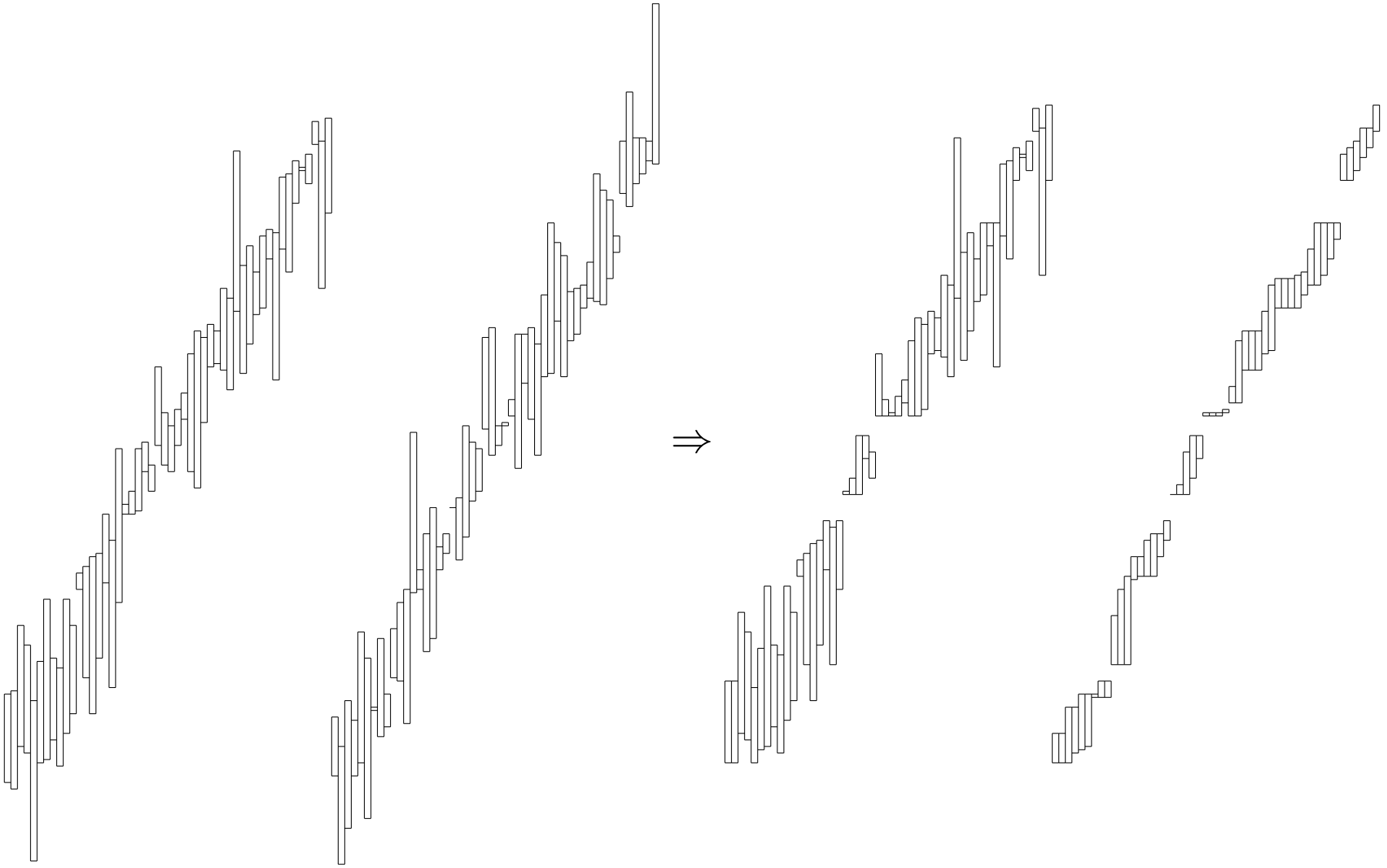
$$\textit{sort}(x_1, \ldots, x_{2n}) \ \wedge \ x_1 \in a'_1 \ \wedge \ \cdots \ \wedge \ x_{2n} \in a'_{2n}$$

Optimal interval narrowing of the sortedness constraint $2n = 22$



Noëlle Bleuzen and Alain Colmerauer, Optimal Narrowing of a Block of Sortings in Optimal time, *Constraints*, 5(1-2), pp 85-118, 2000. A preliminary version is available in `http://alain.colmerauer@free.fr`.

Optimal interval narrowing of the sortedness constraint $2n = 100$



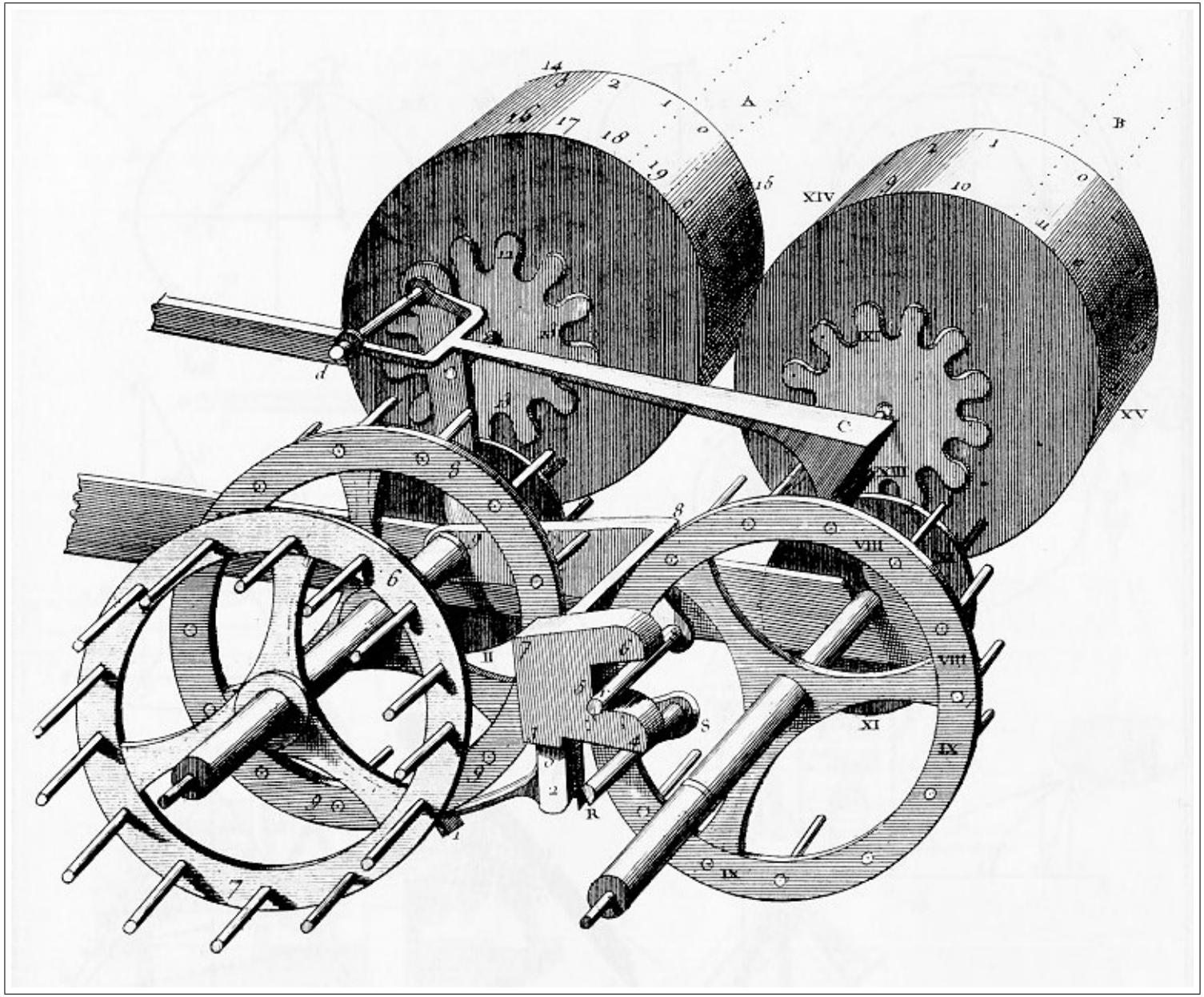
Transition

After the declarative aspect, the basic computatinal aspect.

.

What is a machine and a program?

A machine?

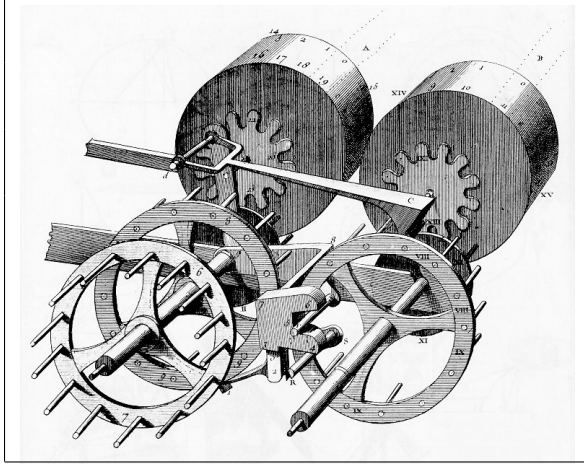


A machine programmed for translating

$\Sigma = \{\sqcup, \texttt{0}, \texttt{1}, \ldots, \texttt{9}, \texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$

`colorless␣green␣ideas␣sleeep␣furiously`

$\Downarrow$



Translating program

$\begin{Bmatrix} \text{instruction 1,} \\ \text{instruction 2,} \\ \ldots \\ \text{instruction } n \end{Bmatrix}$

$\Downarrow$

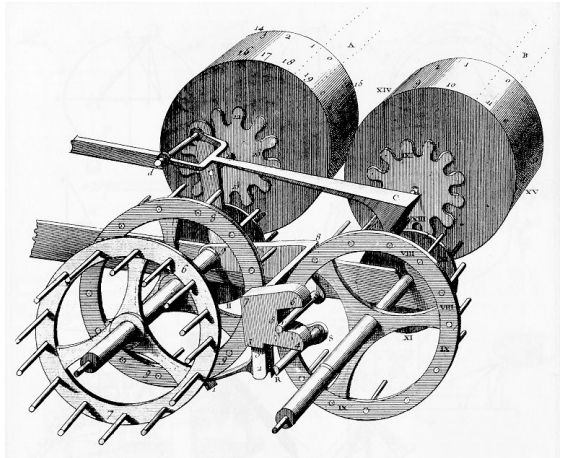
`des␣idees␣vertes␣incolores␣dorment␣furieusement`

A machine programmed for multiplying

$\Sigma = \{\sqcup, \texttt{0}, \texttt{1}, \ldots, \texttt{9}, \texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$

`26010x37979721`

$\Downarrow$



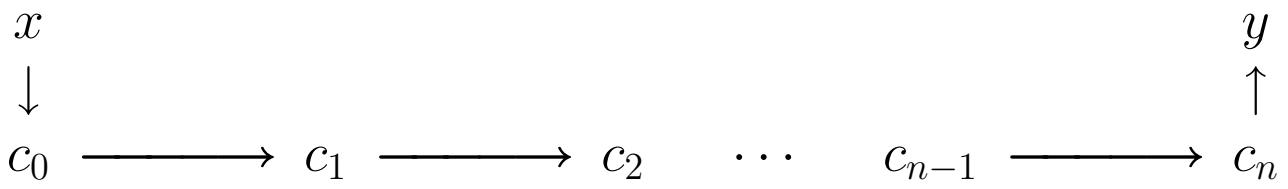
Multiplying program

$\begin{cases} \text{instruction 1,} \\ \text{instruction 2,} \\ \ldots \\ \text{instruction } n \end{cases}$

$\Downarrow$

`9876543210`

A programmed machine is a dynamic system



Formal definition of a programmed machine

- A *machine* M is a 5-tuple $(\Sigma, C, \alpha, \omega, I)$, where

Σ, *the alphabet* is a finite not empty set;

C, is a set, generally infinite, of *configurations*; the ordered pairs (c, c') of elements of C are called *transitions*;

α, *the input function*, maps each element x of $\Sigma^\star$ to a configuration $\alpha(x)$;

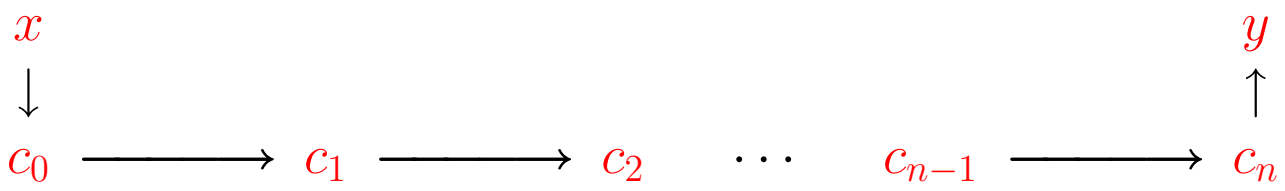
ω, *the ouput function*, maps each configuration c to an element $\omega(c)$ of $\Sigma^\star$;

I, is a countable set of *instructions*, an *instruction* being a set of determinist transitions.

- A *program* P for a machine M is a determinist finite subsets of the instructions set I of M.

How a programmed machine operates

- According to the figure



- for a program P and a finite sequence x of symbols, one defines

$$orbit(P,\, x) \;=\; \begin{cases} \text{the longest sequence } (c_0, c_1)\,(c_1, c_2)\,(c_2, c_3)\;\ldots \text{ with} \\ c_0 = \alpha(x) \text{ and each } (c_i, c_{i+1}) \text{ an element of } \cup P. \end{cases}$$

$$out(P,\, x) \;=\; \begin{cases} \nearrow, & \text{if } orbit(P,\, x) \text{ is infinite,} \\ \omega(c_n), & \text{if } orbit(P,\, x) \text{ ends with } (c_{n-1}, c_n). \end{cases}$$

.

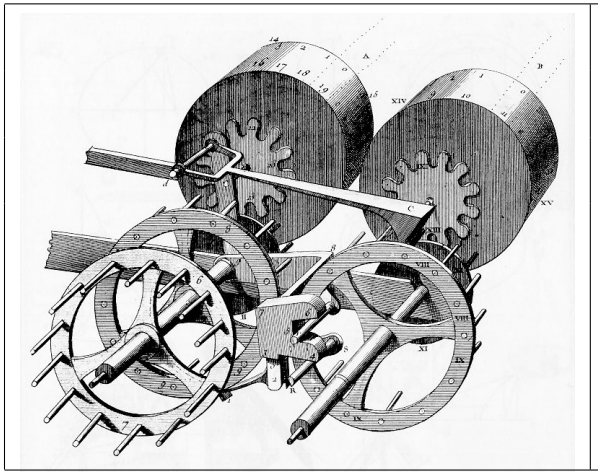
Universal program and coding

Universal program for translating

$\Sigma = \{\sqcup, \texttt{0}, \texttt{1}, \ldots, \texttt{9}, \texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$

Translating program	`colorless␣green␣ideas␣sleeep␣furiously`

$\Downarrow$

	Universal program $\left\{\begin{array}{l}\text{instruction 1,}\\ \text{instruction 2,}\\ \ldots\\ \text{instruction } n\end{array}\right\}$

$\Downarrow$

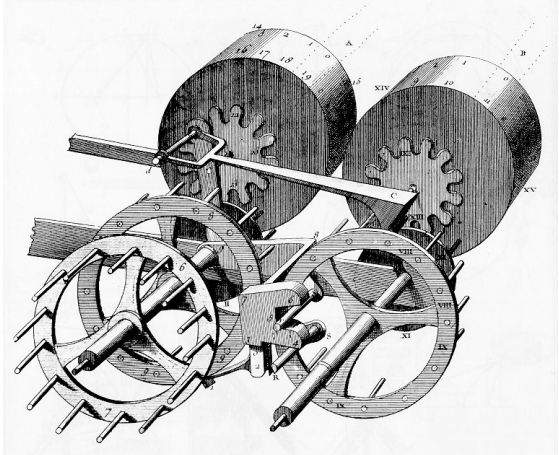
`des␣idees␣vertes␣incolores␣dorment␣furieusement`

Same universal program for multiplying

$\Sigma = \{\sqcup, \texttt{0}, \texttt{1}, \ldots, \texttt{9}, \texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$

Multiplying program	`26010x37979721`

$\Downarrow$



Universal program

$$\begin{Bmatrix} \text{instruction 1,} \\ \text{instruction 2,} \\ \ldots \\ \text{instruction } n \end{Bmatrix}$$

$\Downarrow$

`9876543210`

Definition of a universal pair

Let $M = (\Sigma, C, \alpha, \omega, I)$ be a machine and let us code each program P for M by a word $code(P)$ on Σ.

Definition The pair $(U,\ code)$, the program U and the coding function $code$, are said to be *universal* for M, if, for all programs P of M and for all $x \in \Sigma^\star$,

$$\boxed{out(U,\ code(P) \cdot x) = out(P,\ x)}.$$

If

$$code(U)^n = \overbrace{code(U) \cdot - - - \cdot code(U)}^{n}$$

and if in the above red formula we replace P by U, and x by $code(U)^n \cdot x$ we get :

$$out(U,\ code(U)^{n+1} \cdot x) = out(U,\ code(U)^n \cdot x)$$

and thus:

Property If $(U,\ code)$ is a universal pair, then for all $n \geq 0$ and $x \in \Sigma^\star$,

$$\boxed{out(U,\ code(U)^n \cdot x) = out(U,\ x)}.$$

.

Universal program running on itself

Translating program

1. Give to i the value 0.
2. Increase i by 1.
3. If i is greater than the number of words to be translated, go to 6.
4. Translate the i^{th} word.
5. Go to 2.
6. Stop.

```
colorless␣
green␣ideas
␣sleep␣fu
riously
```

Universal program running on the translating program

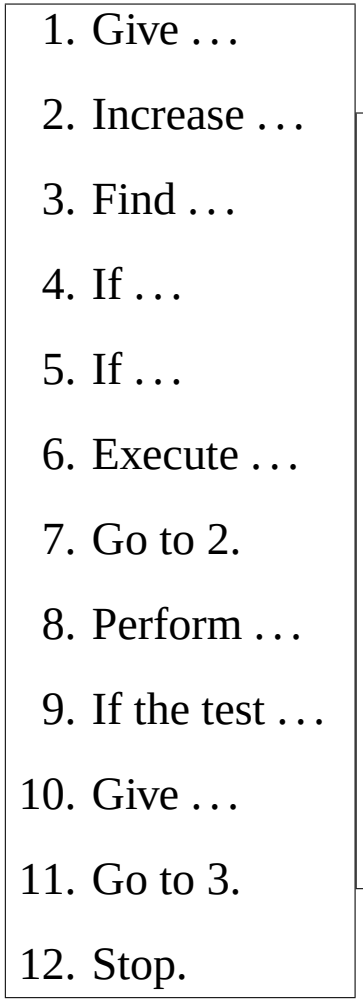
1. Give to n the value 0.
2. Increase n by 1.
3. Find instruction number n at the beginning of the input.
4. If the found instruction is a halting instruction, go to 12.
5. If the found instruction is of the form *"if test then go to* p*"*, go to 8.
6. Execute the found instruction.
7. Go to 2.
8. Perform the test required by the found instruction.
9. If the test succeed, go to 2.
10. Give to n the value p.
11. Go to 3.
12. Stop.

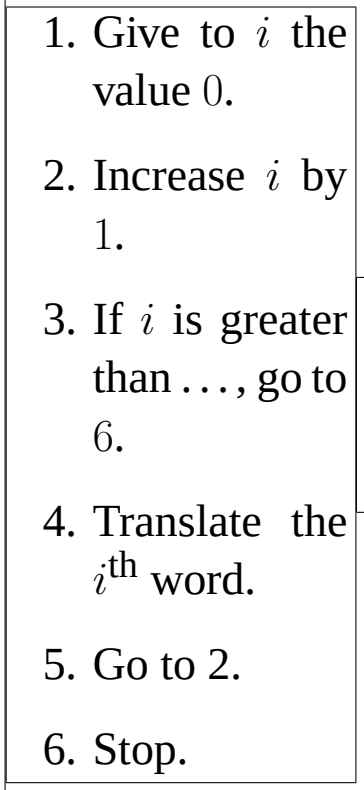
1. Give to i the value 0.
2. Increase i by 1.
3. If i is greater than $\ldots$, go to 6.
4. Translate the i^{th} word.
5. Go to 2.
6. Stop.

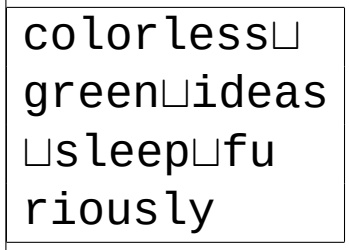
```
colorless␣
green␣ideas
␣sleep␣fu
riously
```

Universal program running on the universal program

1. Give to n the value 0.
2. Increase n by 1.
3. Find instruction number n at the beginning of the input.
4. If the found instruction is a halting instruction, go to 12.
5. If the found instruction is of the form *"if test then go to* p*"*, go to 8.
6. Execute the found instruction.
7. Go to 2.
8. Perform the test required by the found instruction.
9. If the test is not true, go to 2.
10. Give to n the value p.
11. Go to 3.
12. Stop.







Definition of complexity and introspection coefficient

Let $(U, code)$ be a universal pair for the machine $M = (C, I, \Sigma, \alpha, \omega)$.

Definition Given a program P for M and a word x on Σ such that $out(P, x) \neq \nearrow$, the *complexity* of $(U, code)$ is the real number defined by

$$\frac{|orbit(U, code(P) \cdot x)|}{|orbit(P, x)|}.$$

Definition If for all $x \in \Sigma^\star$, with $out(U, x) \neq \nearrow$, the real number

$$\boxed{\lim_{n \to \infty} \frac{|orbit(U, code(U)^{n+1} \cdot x)|}{|orbit(U, code(U)^{n} \cdot x)|}}$$

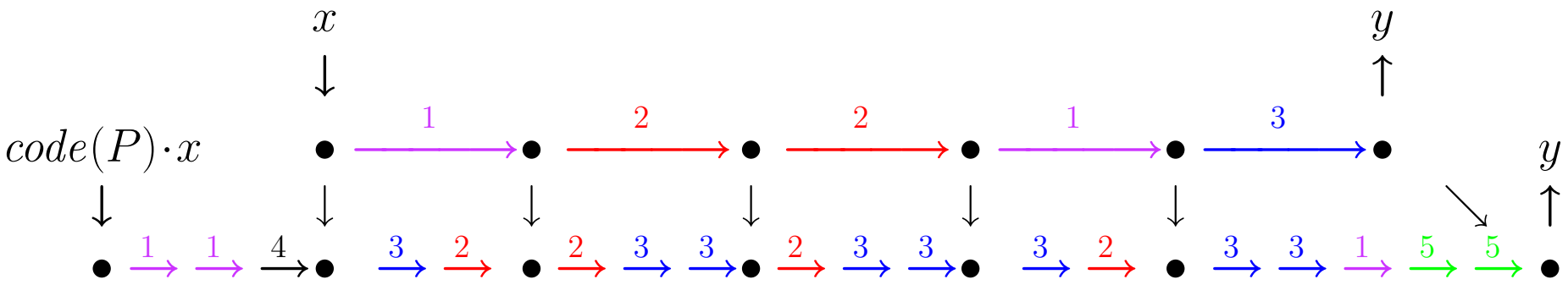
exists and does not depend on x, then this real number is the *introspection coefficient* of the universal pair $(U, code)$.

.

Existence and value of the introspection coefficient

Toward the labeling hypothesis

$$out(P,\, x) = y$$



$$out(U,\, code(P) \cdot x) = y$$

Labeling hypothesis

Hypothesis There exists $n, nb, \mathcal{A}, \mathcal{B}$ such that, for every pair of traces of the form

$$orbit(U,\ code(U)\cdot x),\quad orbit(U,\ x))$$

itself of the form

$$s_1 \cdots s_l,\quad r_1 \cdots r_k$$

we have

$$nb(s_1) \cdots nb(s_l) = \mathcal{B} \cdot \mathcal{A}(nb(r_1)) \cdots \mathcal{A}(nb(r_k)),$$

with n positive integer, with $nb(t) \in 1..n$ for each transtion t of U, with $\mathcal{A}(i)$ a finite sequence on $1..n$ for each $i \in 1..n$, with $\mathcal{B}$ a finite sequence on $1..n$.

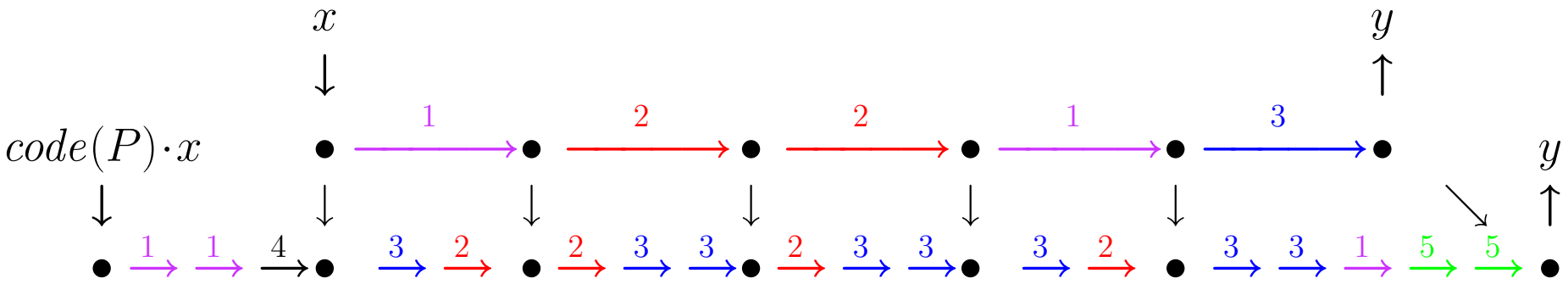
Vector B and Matrix A related to $\mathcal{B}$ and $\mathcal{A}$

$$B = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}, \qquad b_i = \text{number of occurrences of } i \text{ in } \mathcal{B}.$$

$$A = \begin{bmatrix} a_{11} \ldots a_{1n} \\ \vdots \quad\quad \vdots \\ a_{n1} \ldots a_{nn} \end{bmatrix}, \; a_{ij} = \text{number of occurrences of } i \text{ in } \mathcal{A}(j).$$

Vector B and Matrix A related to $\mathcal{B}$ and $\mathcal{A}$, next

- For example, for



- with $n = 5$, we get

$$B = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \qquad A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \end{bmatrix}.$$

Properties of A and B

Property For all $x \in \Sigma^\star$, with $out(U,\, x) \neq \nearrow$, by introducing the column matrix,

$$X^{(k)} = \begin{bmatrix} x_1^{(k)} \\ \vdots \\ x_n^{(k)} \end{bmatrix}, \qquad x_i^{(k)} = \begin{cases} \text{number of occurrences of } i \\ \text{in } nb^*(track(U,\, code(U)^k \cdot x)), \end{cases}$$

we have

$$||X^{(k)}|| = |orbit(U,\, code(U)^k \cdot x)|$$

and

$$X^{(k+1)} = AX^{(k)} + B$$

Main theorem

Theorem Suppose the matrix A admits a real eigenvalue λ, whose multiplicity is equal to 1, which is strictly greater to 1 and to the greatest modulus λ' of the other eigenvalue.

If α is a real number with $\lambda' < \alpha < \lambda$, if

$$X_0 = B,\ X_{n+1} = \frac{1}{\alpha} A X_n,$$

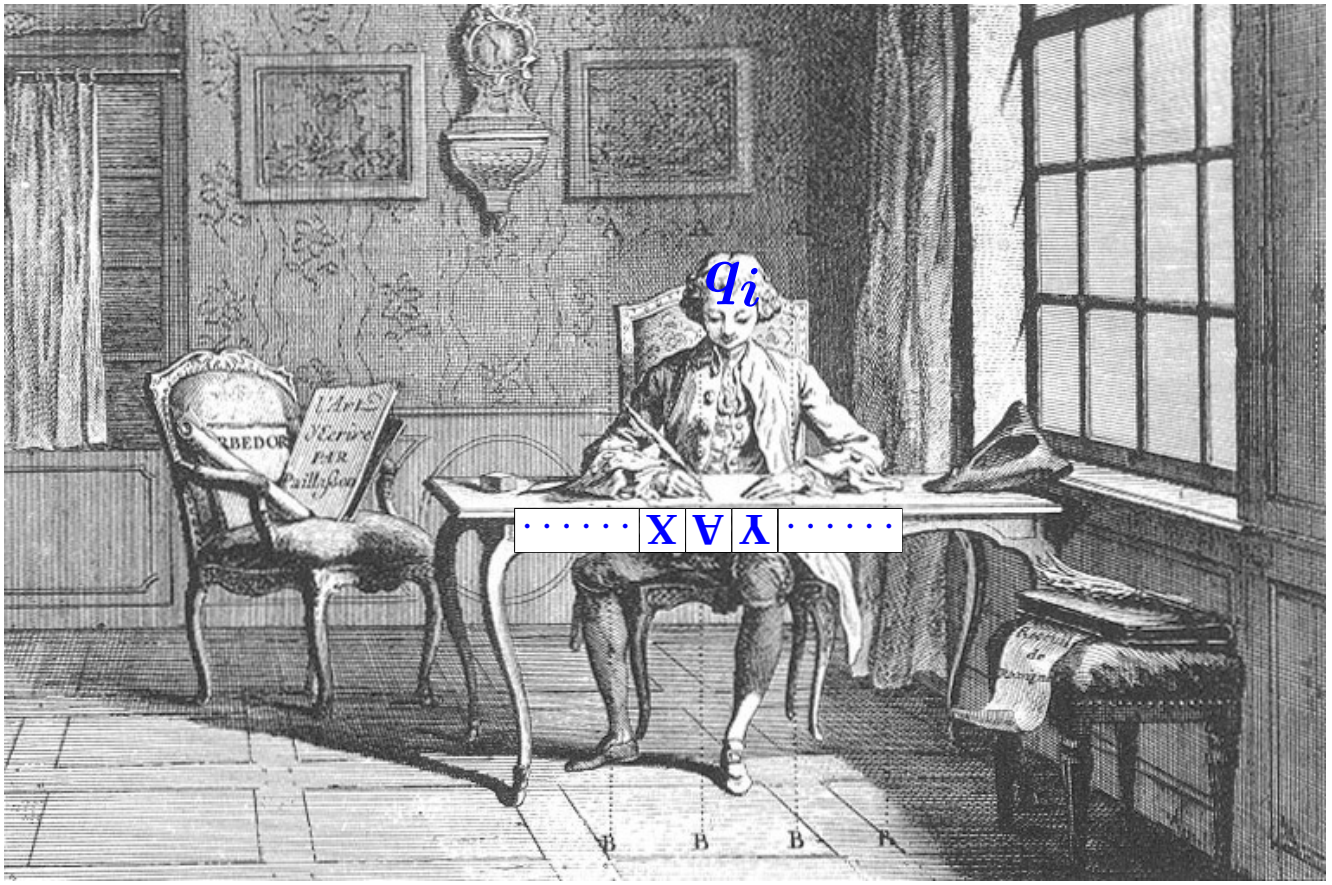
then, when $n \to \infty$, exactly one of the two properties holds:

1. $||X_n|| \to 0$,
2. $||X_n|| \to \infty$. In this case, λ is the inspection coefficient.

.

Turing machine

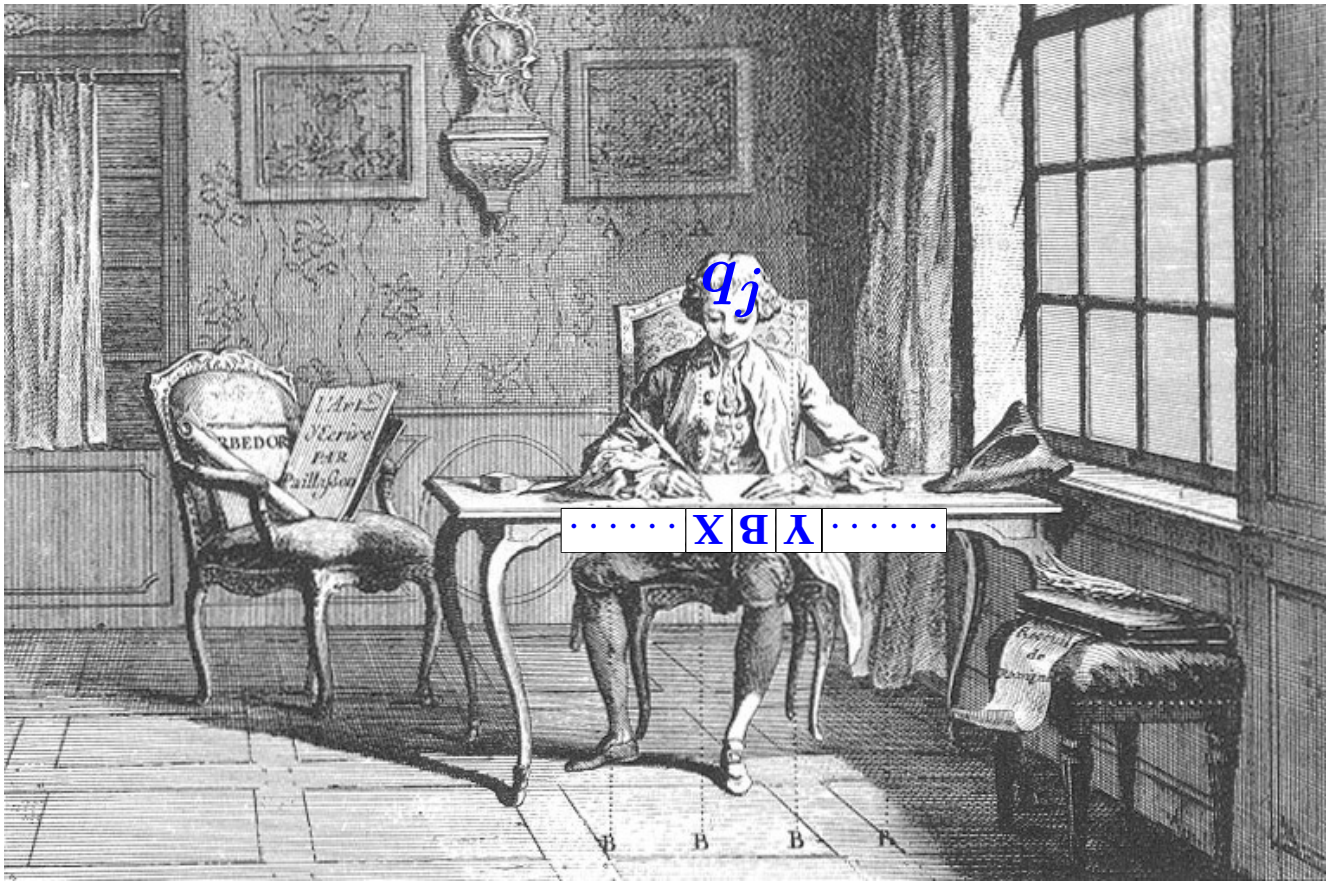
Current configuration



Executed instruction

$$[q_i, \mathrm{AB}R, q_j]$$

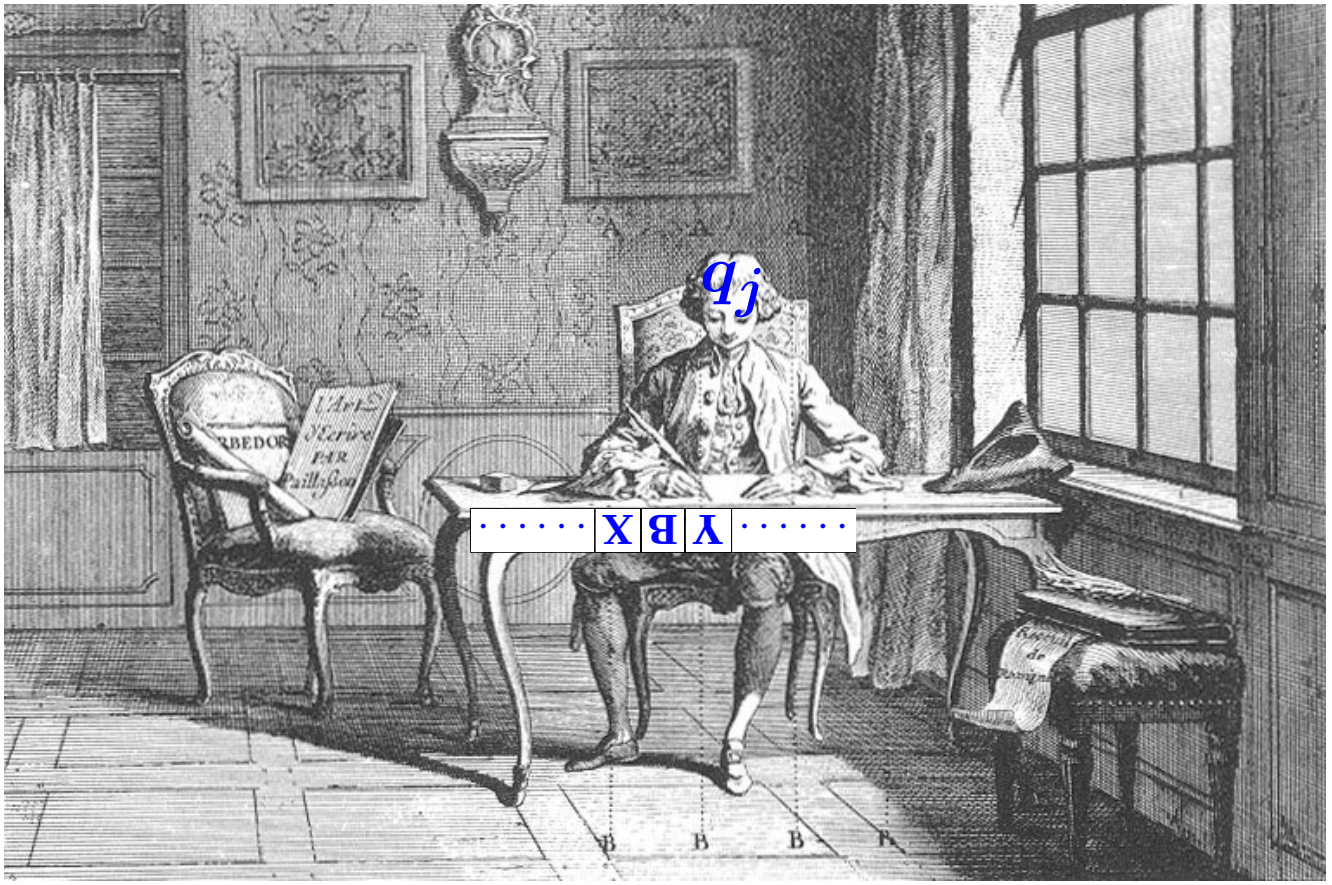
Next configuration



Executed instruction

$$[q_i, \mathrm{AB}L, q_j]$$

Next configuration



Program example

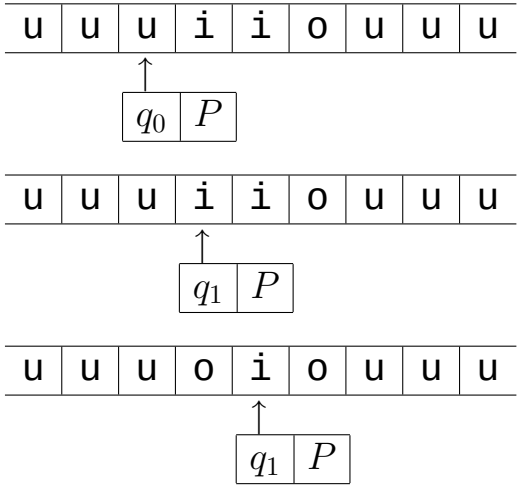
• With program

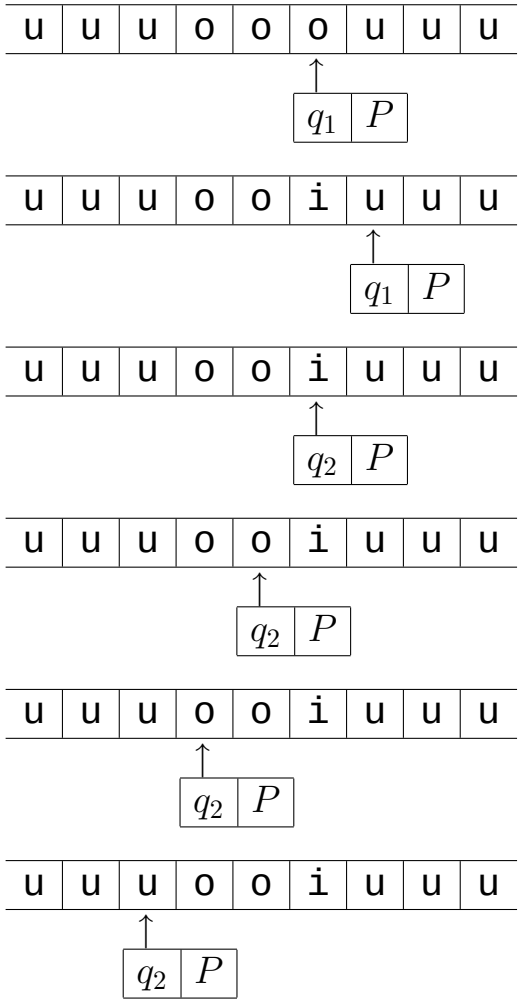
$$P = \{\begin{matrix}[q_0,\ \texttt{uu}R,\ q_1],\\ [q_1,\ \texttt{oi}R,\ q_1],\\ [q_1,\ \texttt{io}R,\ q_1],\\ [q_1,\ \texttt{uu}L,\ q_2],\\ [q_2,\ \texttt{oo}L,\ q_2],\\ [q_2,\ \texttt{ii}L,\ q_2]\end{matrix}\},$$

• and input

iio,

• the machine moves through the sequence of configurations





• and produces the result

ooi

.

Turing machine with internal direction

Program example

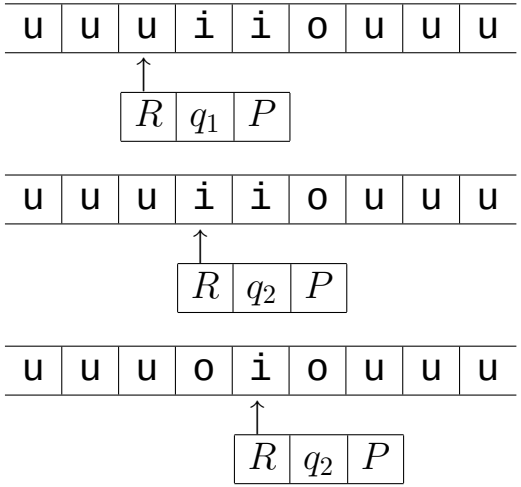
• With program

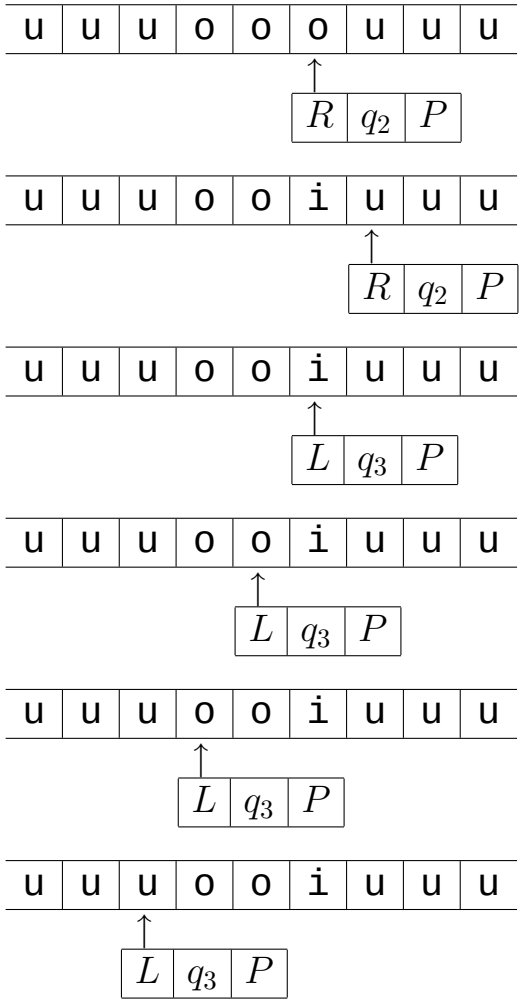
$$P = \{\begin{matrix}[q_1,\ \texttt{uu}+,\ q_2],\\ [q_2,\ \texttt{oi}+,\ q_2],\\ [q_2,\ \texttt{io}+,\ q_2],\\ [q_2,\ \texttt{uu}-,\ q_3],\\ [q_3,\ \texttt{oo}+,\ q_3],\\ [q_3,\ \texttt{ii}+,\ q_3]\end{matrix}\},$$

• and input

iio,

• the machine moves through the sequence of configurations



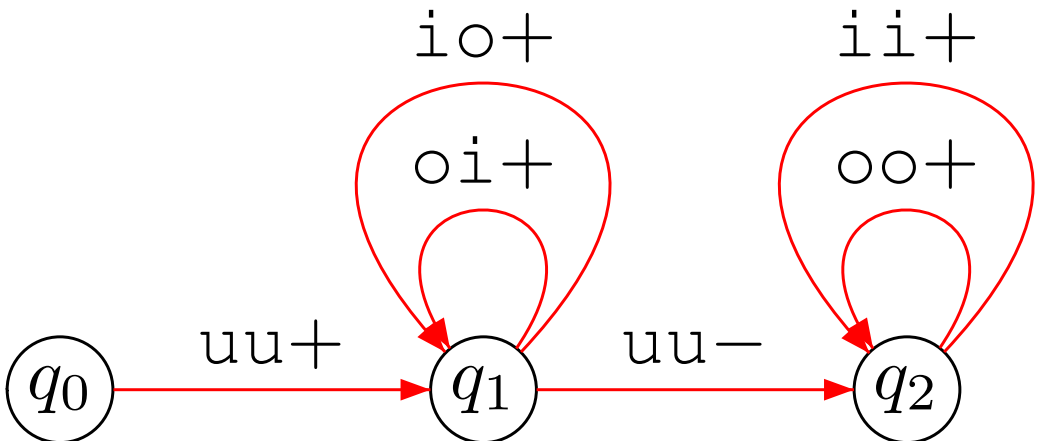


• and produce the result

ooi

Graph of a program

$$\left\{\begin{array}{l}[q_0,\ \mathtt{u},\ \mathtt{u},\ +1,\ q_1],\\ [q_1,\ \mathtt{o},\ \mathtt{i},\ +1,\ q_1],\\ [q_1,\ \mathtt{i},\ \mathtt{o},\ +1,\ q_1],\\ [q_1,\ \mathtt{u},\ \mathtt{u},\ -1,\ q_2],\\ [q_2,\ \mathtt{o},\ \mathtt{o},\ +1,\ q_2],\\ [q_2,\ \mathtt{i},\ \mathtt{i},\ +1,\ q_2]\end{array}\right.$$



Formal definition of a machine

A *machine* M is a 5-tuple $(\Sigma, C, \alpha, \omega, I)$, where

Σ, *the alphabet* is a finite not empty set;

C, is a set, generally infinite, of *configurations*; the ordered pairs (c, c') of elements of C are called *transitions*;

α, *the input function*, maps each element x of $\Sigma^\star$ to a configuration $\alpha(x)$;

ω, *the ouput function*, maps each configuration c to an element $\omega(c)$ of $\Sigma^\star$;

I, is a countable set of *instructions*, an *intruction* being a set of determinist transitions.

Formal definition of a Turing machine (with internal direction)

A Turing machine, with internal direction, has a 5-tuple of the form $(\Sigma, C, \alpha, \omega, I)$ where,

- Σ is a finite set not having u as an element,
- C is the set of 5-tuples of the form $[d, q_i, \cdot x, a, y\cdot]$, with $d \in \{L, R\}$, q_i being a state, x, y taken from $\Sigma^\star_{\mathsf{u}}$ and a taken from Σ_{u}, where $\Sigma_{\mathsf{u}} = \Sigma \cup \{\mathsf{u}\}$,
- $\alpha(x) = [R, q_1, \varepsilon, \mathsf{u}, x]$, for all $x \in \Sigma^\star$,
- $\omega([d, q_i, \cdot x, a, y\cdot])$ is the longest element of $\Sigma^\star$ beginning $y\cdot$,
- I is the set of instruction denoted and defined, for all states q_i, q_j, all elements a, b of Σ_{u} and all $s \in \{+, -\}$, by
  $[q_i,\ abs,\ q_j] \stackrel{\text{def}}{=}$
  $\{([d, q_i, \cdot xc, a, y\cdot],\ [L, q_j, \cdot x, c, by\cdot]) \mid\ (d, s) \in E_1 \text{ and } (x, c, y) \in F\}\} \cup$
  $\{([d, q_i, \cdot x, a, cy\cdot],\ [R, q_j, \cdot xb, c, y\cdot]) \mid\ (d, s) \in E_2 \text{ and } (x, c, y) \in F\}$,
  wth $E_1 = \{(L, +), (r, -)\}$, $E_2 = \{(L, +), (R, -)\}$ and $F = \Sigma^\star_{\mathsf{u}} \times \Sigma_{\mathsf{u}} \times \Sigma^\star_{\mathsf{u}}$.

.

Universal pair $(U, code)$ for Turing machine (with internal direction)

Coding function *code*

• Let P be a program for M. We take $code(P)$ as the word on $\{\mathtt{o}, \mathtt{i}, \mathtt{z}\}$

$$code(P) = \mathtt{z}I_{4n}\mathtt{z}\ldots\mathtt{z}I_{k+1}\mathtt{z}I_k\mathtt{z}I_{k-1}\mathtt{z}\ldots\mathtt{z}I_1\mathtt{zoi}\ldots\mathtt{izz}.$$

• Integer n is the number of states of P and the I_k's are the coded instructions.

• The size of the *shuttle* $\mathtt{oi}\ldots\mathtt{iz}$ is equal to the longest size of the I_k's minus 5.

Coding function *code*, next

In order to assign a position to each coded instruction I_k of $[q_i, abs, q_j]$, we introduce the number:

$$\pi(i,a) = 4(i-1) + \begin{cases} 1, & \text{if } a = \texttt{u} \\ 2, & \text{if } a = \texttt{o} \\ 3, & \text{if } a = \texttt{i} \\ 4, & \text{if } a = \texttt{z} \end{cases}.$$

Coding function *code*, next next

For all $a \in \Sigma_{\mathtt{u}}$ and $i \in 1..n$,

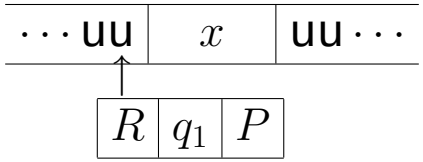
$$I_{\pi(i,a)} = \begin{cases} \overline{[q_i, abs, q_j]}, & \text{if there exists } b, s, j \text{ with } [q_i, abs, q_j] \in P, \\ \mathtt{oi}, & \text{otherwise,} \end{cases}$$

- with $\overline{[q_i, a, b, s, q_j]} = \begin{cases} \mathtt{i}a_m \dots a_2\mathtt{o}, & \text{if } \pi(i,a) < \frac{1}{2}(\pi(j,\mathtt{u}) + \pi(j,\mathtt{z})), \\ \mathtt{o}a_2 \dots a_m\mathtt{i}, & \text{if } \pi(i,a) > \frac{1}{2}(\pi(j,\mathtt{u}) + \pi(j,\mathtt{z})), \end{cases}$,
- with a_2a_3 equal to $\mathtt{io}$, $\mathtt{oi}$, $\mathtt{ii}$, depending whether b equals $\mathtt{u}$, $\mathtt{o}$, $\mathtt{i}$, $\mathtt{z}$,
- with $a_4 = \mathtt{o}$ or $a_4 = \mathtt{i}$ depending whether $s = +$ or $s = -$ and
- with $\mathtt{i}a_m \dots a_5$ a binary number ($\mathtt{o}$ for 0 and $\mathtt{i}$ for 1) whose value is equal to

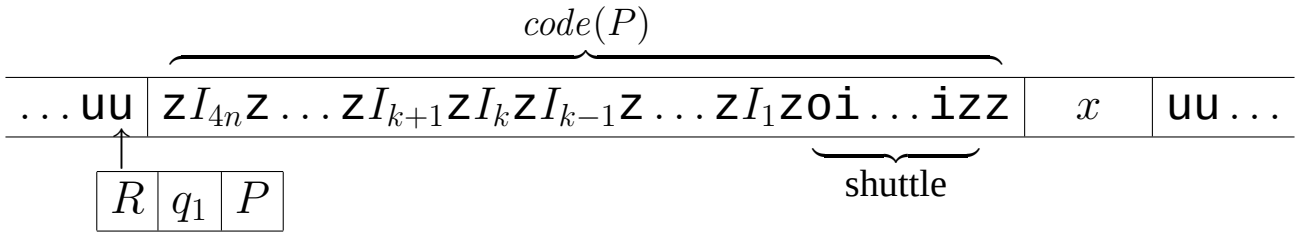
$$|\pi(j) - \pi(i,a)| + \frac{3}{2}.$$

How program U operates

- **Initial configuration of P**

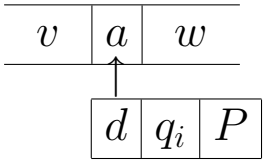


- **Initial corresponding configuration of U**

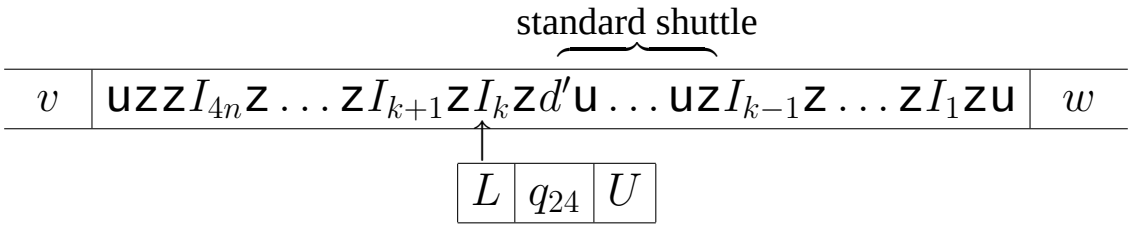


How program U operates, next 1

• **Current configuration of P**

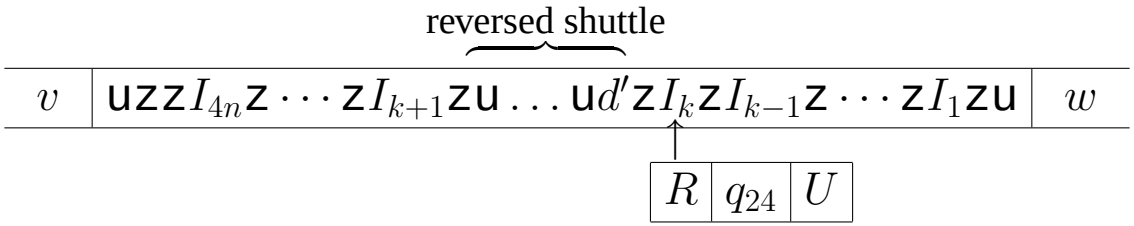


• **Current corresponding configuration of U**



(1)

or

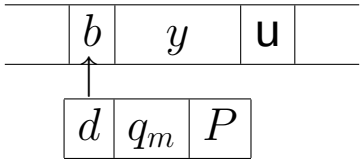


(2)

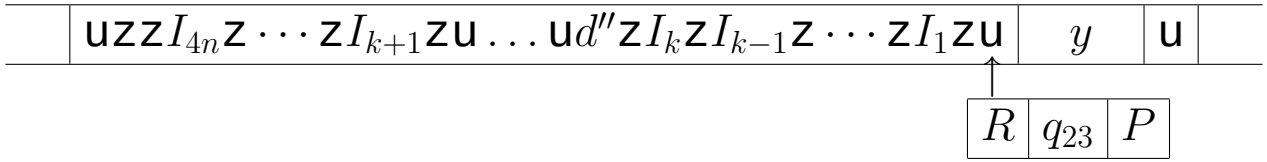
depending whether I_k, with $k = \pi(i, a)$, is in the *standard* form $\mathtt{i}a_m \ldots a_2\mathtt{o}$ or in the *reversed* form $\mathtt{i}a_m \ldots a_2\mathtt{o}$. The read-write points to a_3 or to the z which follows I_k when I_k is the empty instruction oi. Depending whether d is equal to L or R, the symbol d' is equal to u or o, if I_k is standard, and to o or u, if I_k is reversed.

How program U operates, next 2

- **Final configurations of P**

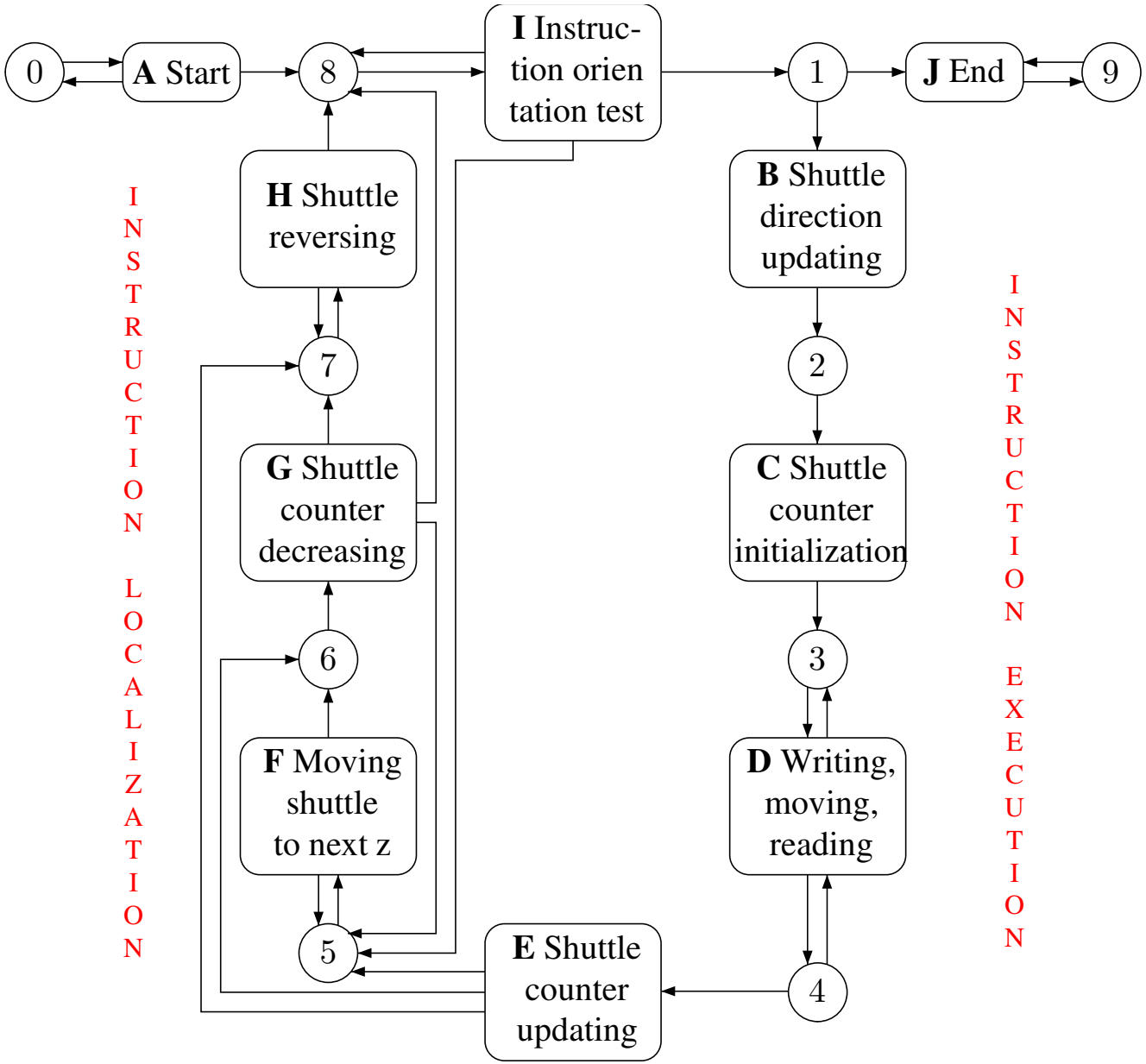


- **Final corresponding configurations of U**

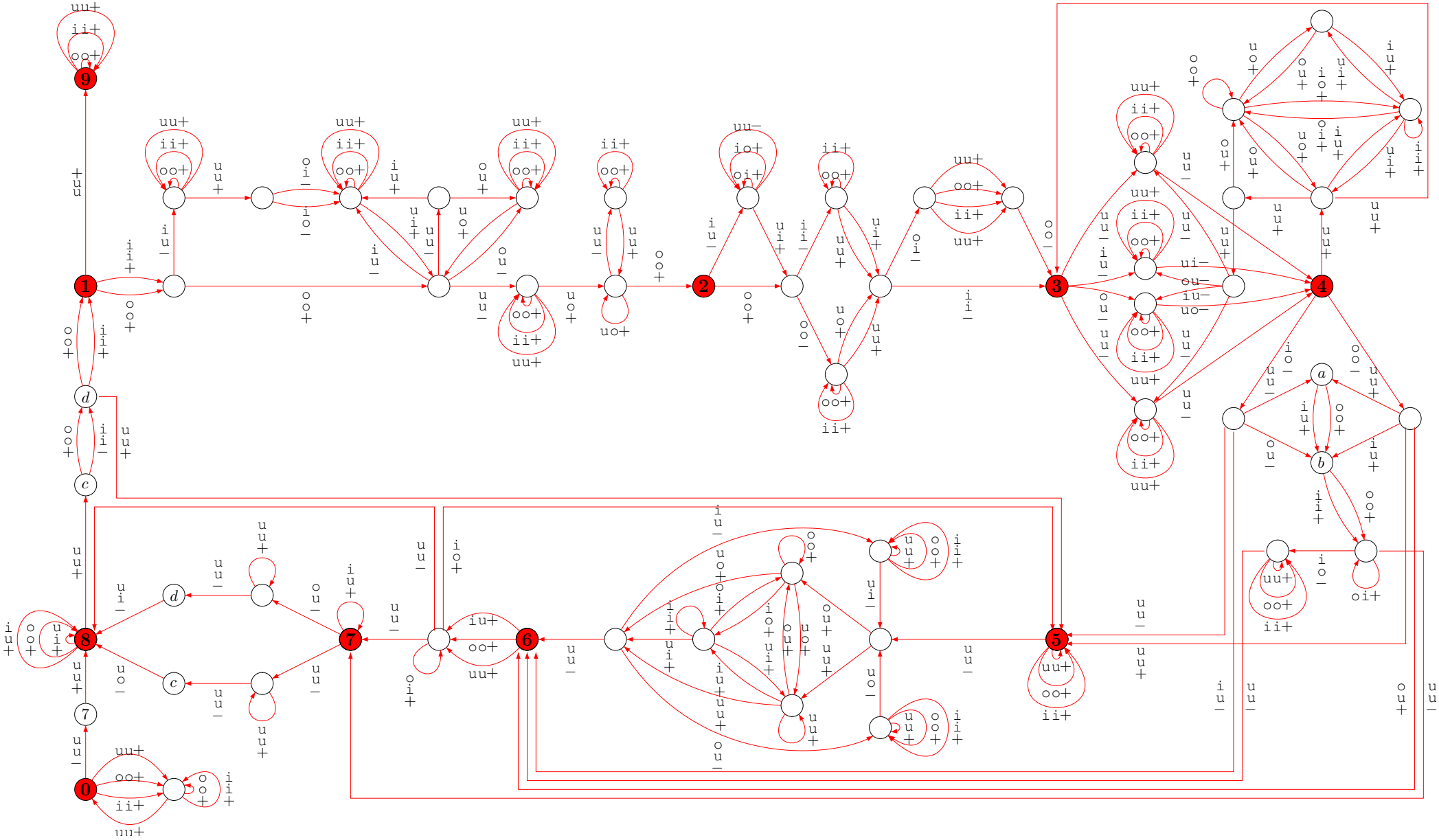


with $I_k = \texttt{oi}$, $k = \pi(m, b)$ and d' equal to o or u, depending whether d equals L or R.

Architecture of program *U*



Graph of program U



Listing of program U

54 states and 184 instructions

```
# A BEGINNING
[X0,uz+,A1],   [X0,oo+,A1],   [X0,ii+,A1],   [X0,zu-,X7],
               [A1,oo+,A1],   [A1,ii+,A1],   [A1,zz+,X0],
                                             [X7,zz+,X8],

# B INSTRUCTION TAIL COPYING
               [X1,oo+,B1],   [X1,ii+,B1],
               [B1,oo+,B5],   [B1,iu-,B2],
               [B2,oo+,B2],   [B2,ii+,B2],   [B2,zz+,B3],
               [B3,oi-,B4i],  [B3,io-,B4i],
[B4o,uo+,B5],  [B4o,oo+,B4o], [B4o,ii+,B4o], [B4o,zz+,B4o],
[B4i,ui+,B5],  [B4i,oo+,B4i], [B4i,ii+,B4i], [B4i,zz+,B4i],
[B5,uu-,B6],   [B5,ou-,B4o],  [B5,iu-,B4i],  [B5,zz-,B7],
               [B6,oo+,B4o],  [B6,ii+,B4i],
# Replacement of the remaining u’s by o’s
[B7,uo+,B9],   [B7,oo+,B7],   [B7,ii+,B7],   [B7,zz+,B7],
[B9,uo+,B9],   [B9,oo+,X2],                  [B9,zz-,B10],
               [B10,oo+,B10], [B10,ii+,B10], [B10,zz+,B9],

# C INSTRUCTION HEAD COPYING
# Creating the symbol to be written
               [X2,oo+,C2],   [X2,iu-,C1],
[C1,ui+,C2],   [C1,oi+,C1],   [C1,io+,C1],   [C1,zu-,C1],
               [C2,oo-,C3o],  [C2,ii-,C3i],
[C3o,uo+,C4],  [C3o,oo+,C3o], [C3o,ii+,C3o], [C3o,zu+,C4],
[C3i,uz+,C4],  [C3i,oo+,C3i], [C3i,ii+,C3i], [C3i,zi+,C4],
# Taking in account the direction
               [C4,oi-,C5],   [C4,ii-,X3],
[C5,uu+,C6],   [C5,oo+,C6],   [C5,ii+,C6],   [C5,zz+,C6],
               [C6,oo-,X3],
```

Listing of program U, next 1

```
# D WRITING, MOVING AND READING
# Writing and reading
[X3,uu-,D1u],  [X3,ou-,D1o],  [X3,iu-,D1i],  [X3,zu-,D1z],
[D0,uu-,D1z],  [D0,ou-,D1i],  [D0,iu-,D1o],  [D0,zu-,D1u],
[D1u,uu-,X4],  [D1u,oo+,D1u], [D1u,ii+,D1u], [D1u,zz+,D1u],
[D1o,uo-,X4],  [D1o,oo+,D1o], [D1o,ii+,D1o], [D1o,zz+,D1o],
[D1i,ui-,X4],  [D1i,oo+,D1i], [D1i,ii+,D1i], [D1i,zz+,D1i],
[D1z,uz-,X4],  [D1z,oo+,D1z], [D1z,ii+,D1z], [D1z,zz+,D1z],
# Moving
                                             [X4,zu+,D2z],
               [D2u,ou+,D2o], [D2u,iu+,D2i],
[D2o,uo+,D2u], [D2o,oo+,D2o], [D2o,io+,D2i], [D2o,zo+,D2z],
[D2i,ui+,D2u], [D2i,oi+,D2o], [D2i,ii+,D2i], [D2i,zi+,D2z],
[D2z,uz+,X3],  [D2z,oz+,D2o], [D2z,iz+,D2i], [D2z,zz+,D2zz],
[D2zz,uz+,D0], [D2zz,oz+,D2o],

# E SHUTTLE UPDATING
# Beginning of the updating
               [X4,oo-,E1b],  [X4,io-,E1a],
[E1a,uz-,E2a], [E1a,oz-,E2b], [E1a,iz-,X6],  [E1a,zz-,X5],
[E1b,uz+,X5],  [E1b,oz+,X6],  [E1b,iz+,E2b], [E1b,zz+,E2a],
# End of the updating
               [E2a,oo+,E2b], [E2a,iu+,E2b],
               [E2b,oo+,E4],  [E2b,ii+,E4],
               [E4,oi+,E4],   [E4,io-,E5],   [E4,zz-,X7],
[E5,uu+,E5],   [E5,oo+,E5],   [E5,ii+,E5],   [E5,zz-,X6],
```

Listing of program U, next 2

```
# F SUTTLE MOVING TO NEXT z
[X5,uu+,X5],   [X5,oo+,X5],   [X5,ii+,X5],   [X5,zz-,F1],
[F1,uz+,F2u],  [F1,oz+,F2o],
[F2u,uu+,F2u], [F2u,ou+,F2o], [F2u,iu+,F2i], [F2u,zu+,F3],
[F2o,uo+,F2u], [F2o,oo+,F2o], [F2o,io+,F2i], [F2o,zo+,F3],
[F2i,ui+,F2u], [F2i,oi+,F2o], [F2i,ii+,F2i], [F2i,zi+,F3],
               [F3,oz-,F4o],  [F3,iz-,F4i],  [F3,zz-,X6],
[F4o,uu+,F4o], [F4o,oo+,F4o], [F4o,ii+,F4o], [F4o,zo-,F1],
[F4i,uu+,F4i], [F4i,oo+,F4i], [F4i,ii+,F4i], [F4i,zi-,F1],

# G SHUTTLE DECREASING
[X6,uu+,G1],   [X6,oo+,G1],   [X6,iu+,G1],
[G1,uu-,X7],   [G1,oi+,G1],   [G1,io+,X5],   [G1,zz-,X8],

# H SHUTTLE REVERSING AFTER BLANK SYMBOLS INTRODUCTION
[X7,uu-,E2a],  [X7,ou-,E2b],  [X7,iu+,X7],
[E2a,uu+,E2a],                                [E2a,zz-,I1],
[E2b,uu+,E2b],                                [E2b,zz-,I2],
[I1,uo-,X8],
[I2,ui-,X8],

# I INSTRUCTION ORIENTATION TEST AFTER BLANK SYMBOLS INTRODUCTION
[X8,ui+,X8],   [X8,oo+,X8],   [X8,iu+,X8],   [X8,zz+,I1],
               [I1,oo+,I2],   [I1,ii-,I2],
               [I2,oo+,X1],   [I2,ii+,X1],   [I2,zz+,X5],

# J END OF THE PROGRAM
                                              [X1,zz+,X9],
               [X9,oo+,X9],   [X9,ii+,X9],   [X9,zz+,X9]];
```

.

Complexity and introspection coefficient of $(\boldsymbol{U}, \boldsymbol{code})$

Complexity of our Turing machine on examples

• First we have chosen a reversing program P such that, for all $n \geq$, one gets $out(P,\ a_1 a_2 \ldots a_n) = a_n \ldots a_2 a_1$, with the a_i's taken from $\{\texttt{o}, \texttt{i}, \texttt{z}\}$. The program P has 32 instructions and 9 states. We have $|code(P)| = 265$ and $|code(U)| = 1552$.

• We have obtained the following results for the pair $(U,\ code)$:

x	$\lvert orbit(P, x)\rvert$	$\lvert orbit(U,$ $code(P)\cdot x)\rvert$	$\lvert orbit(U,$ $code(U)\cdot code(P)\cdot x)\rvert$	$\frac{\lvert orbit(U,\ code(U)\cdot code(P)\cdot x)\rvert}{\lvert orbit(U,\ code(P)\cdot x)\rvert}$
ε	2	5 927	22 974 203	3 876.19
o	6	13 335	51 436 123	3 857.23
oi	12	23 095	88 887 191	3 848.76
oiz	20	35 377	136 067 693	3 846.22
oizo	30	49 663	190 667 285	3 839.22

• It can be seen that we have succeeded in running the universal program U on its own code and thus to compute a first approximation of the introspection coefficient.

Introspection coefficient of our pair for the Turing machine

• After having computed the column vector B of size $184 \times 4 = 736$ and the matrix A of size 736, using the main Theorem, we have verified that U admits an introspection coefficient and computed its value: for all words x on Σ such that $orbit(P,\ x) \neq \nearrow$,

$$\boxed{\lim_{n \to \infty} \frac{|orbit(U,\ code_1(U)^{n+1} \cdot x)|}{|orbit(U,\ code(U)^n \cdot x)|} = 3\,672.98}$$

• There, it is also proven that a more classical Turing machine, with 361 instructions and 106 states, has the same introspection coefficient.

• Alain Colmerauer, On the complexity of universal programs, *Machine, Computations and Universality (Saint-Petersburg 2004).* Lecture Notes in Computer, pp 18-35, 2005. A preliminary version is available in `http://alain.colmerauer@free.fr`.

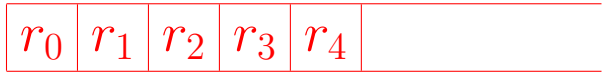
.

Arithmetic machine with indirect adressing

Iindirect adressing machine

- The input-output alphabet $\Sigma = \{c_1, \ldots, c_m\}$, where the c_i's are any symbols.

- The machine is made of an infinity of registers r_i:



Each register contains an natural unbounded integer.

- The possible instructions are

$$
\begin{aligned}
[i, cst, j, k] &: \text{ if } r_0 = i \text{ then } r_j := k \text{ and } r_0 := r_0 + 1,\\
[i, plus, j, k] &: \text{ if } r_0 = i \text{ then } r_j := r_j + r_k \text{ and } r_0 := r_0 + 1,\\
[i, sub, j, k] &: \text{ if } r_0 = i \text{ then } r_j := \max\{0, r_j - r_k\} \text{ and } r_0 := r_0 + 1,\\
[i, from, j, k] &: \text{ if } r_0 = i \text{ then } r_j := r_{r_k} \text{ and } r_0 := r_0 + 1,\\
[i, to, j, k] &: \text{ if } r_0 = i \text{ then } r_{r_j} := r_k \text{ and } r_0 := r_0 + 1,\\
[i, ifzero, j, k] &: \text{ if } r_0 = i \text{ then } r_0 := \begin{cases} r_k + 1, & \text{if } r_j = 0,\\ r_0 + 1, & \text{if } r_j \neq 0. \end{cases}
\end{aligned}
$$

Indirect addressing machine, formal definition

Definition An indirect addressing arithmetic machine has a 5-tuple $(\Sigma, C, \alpha, \omega, I)$ where,

- $\Sigma = \{c_1, \ldots, c_m\}$, where the c_i's are any symbols,
- C is the set of infinite sequences on natural integers of the form $r = (r_0, r_1, r_2, \ldots)$,
- $\alpha(a_1 \ldots a_n) = (0, 25, 1, \ldots, 1, r_{24+1}, \ldots, r_{24+n}, 0, 0, \ldots)$, with r_{24+i} equal to $1, \ldots, m$ depending whether a_i equals $c_1, \ldots, c_m$,
- $\omega(r_0, r_1, \ldots) = a_1 \ldots a_n$, with a_i equal to $c_1, \ldots, c_m$ depending whether a_i equals $1, \ldots, m$, and satisfying the condition that n is the greatest integer such that $r_{r_1}, \ldots, r_{r_1+n}$ are elements of $\{1, \ldots, m\}$,
- I is the set of instructions denoted and defined, for all natural integers i, j, k, by:

$$
\begin{aligned}
[i, cst, j, k] &\stackrel{\text{def}}{=} \{(r,\ s) \in C^2 \,|\, r_0 = i, s := r,\ s_j := k,\ s_0 := s_0 + 1\},\\
[i, plus, j, k] &\stackrel{\text{def}}{=} \{(r,\ s) \in C^2 \,|\, r_0 = i, s := r,\ s_j := s_j + s_k,\ s_0 := s_0 + 1\},\\
[i, sub, j, k] &\stackrel{\text{def}}{=} \{(r,\ t) \in C^2 \,|\, r_0 = i,\ s := r,\ s_j := \max\{0, s_j - s_k\},\ s_0 := s_0 + 1\},\\
[i, from, j, k] &\stackrel{\text{def}}{=} \{(r,\ t) \in C^2 \,|\, r_0 = i,\ s := r,\ s_j := s_{s_k},\ s_0 := s_0 + 1\},\\
[i, to, j, k] &\stackrel{\text{def}}{=} \{(r,\ t) \in C^2 \,|\, r_0 = i,\ s := r,\ s_{r_j} = r_k,\ s_0 := s_0 + 1\},\\
[i, ifzero, j, k] &\stackrel{\text{def}}{=} \{(r,\ t) \in C^2 \,|\, r_0 = i,\ s := r, \text{ if } s_j = 0 \text{ then } s_0 := s_k,\ s_0 := s_0 + 1\}.
\end{aligned}
$$

.

Universal pair $(U, code)$ for arithmetic machine with indirect addressing

Execution of P

- Current configuration

r_0	r_1	r_2	
0	3	2	

$$P = \left\{ \begin{array}{l} [0, plus, 2, 1], \\ [1, cst, 11, 1], \\ [2, from, 5, 2], \\ [3, ifzero, 5, 8], \\ [4, sub, 5, 11], \\ [5, to, 2, 5], \\ [6, plus, 2, 11], \\ [7, cst, 0, 1] \end{array} \right\}$$

- Next current configuration

r_0	r_1	r_2	
1	3	5	

Execution of U

- Current corresponding configuration

r_0	r_1	r_2					
50			P	0	3	2	

$$U = \left\{ \begin{array}{l} [0, cst, 8, 0], \\ [1, cst, 10, 2], \\ [2, cst, 11, 11], \\ \ldots \\ [99, cst, 0, 49], \\ [100, plus, 9, 1], \\ [101, from, 9, 9], \\ [102, plus, 1, 9] \end{array} \right\}$$

- Next current corresponding configuration

r_0	r_1	r_2					
50			P	1	3	5	

Listing of program U

```
# INITIALISATION
# OF THE REGISTERS
[0,cst,8,0],
[1,cst,10,2],
[2,cst,11,11],
[3,cst,12,20],
[4,cst,13,26],
[5,cst,14,33],
[6,cst,15,67],
[7,cst,16,68],
[8,cst,17,70],
[9,cst,18,76],
[10,cst,19,82],
[11,cst,20,88],
[12,cst,21,94],

# ENCODING OF THE
# EMULATED PROGRAM
# INITIALISATION OF THE
# SOURCE POSITION R[1] AND
# THE BOOLEEN VALUE c
[13,cst,2,24],
[14,cst,5,1],
[15,cst,0,16],
# INCREASING THE
# SOURCE POSITION R[1]
[16,plus,1,9],
# CASE STUDY ACCORDING
# TO THE VALUE a OF R[R[1]]
[17,from,3,1],
[18,to,1,8],
[19,plus,3,11],
[20,from,0,3],
# CASE WHERE a=1
[21,ifzero,5,24],
[22,cst,5,0],
[23,cst,4,0],
[24,plus,4,4],
[25,plus,4,9],
[26,cst,0,15],
# CASE WHERE a=2
[27,ifzero,5,30],
[28,cst,5,0],
[29,cst,4,0],
[30,plus,4,4],
[31,plus,4,9],
[32,plus,4,9],
[33,cst,0,15],
# CASE WHERE a=3
[34,ifzero,5,36],
[35,cst,0,40],
[36,plus,2,9],
[37,sub,4,9],
[38,to,2,4],
[39,cst,5,1],
[40,cst,0,15],
# END
[41,to,1,8],
[42,cst,4,1],
[43,plus,4,1],
[44,cst,6,25],
[45,to,4,6],

# PROGRAM EMULATION
# SKIP INCREMENTATION
# OF THE INSTRUCTION COUNTER
[46,cst,0,49],
# INCREASING
# THE INSTRUCTION COUNTER
[47,from,6,1],
[48,plus,6,9],
[49,to,1,6],
```

Listing of program U, next

```
# COMPUTING THE POSITION
# OF INSTRUCTION NB ZERO
[50,from,7,1],
[51,cst,6,25],
[52,plus,6,7],
[53,plus,6,7],
[54,plus,6,7],
# HALTING TEST
[55,cst,7,0],
[56,plus,7,1],
[57,sub,7,6],
[58,ifzero,7,100],
# COMPUTING a:=R[R[6]],
[59,from,3,6],
# COMPUTING b:=R[R[6]]+R[1]
[60,plus,6,9],
[61,from,4,6],
[62,plus,4,1],
# COMPUTING c:=R[R[4]+2];
[63,plus,6,9],
[64,from,5,6],
# CASE STUDY ACCORDING
# TO THE VALUE OF a
[65,cst,6,15],
[66,plus,6,3],
[67,from,0,6],
# NO INSTRUCTION
[68,cst,0,99],
# CONSTANT INSTRUCTION
[69,to,4,5],
[70,cst,0,46],
# PLUS INSTRUCTION
[72,plus,5,1],
[73,from,5,5],
[74,plus,6,5],
[75,to,4,6],
[76,cst,0,46],
# MINUS INSTRUCTION
[77,from,6,4],
[78,plus,5,1],
[79,from,5,5],
[80,sub,6,5],
[81,to,4,6],
[82,cst,0,46],
# FROMINDIRECT INSTRUCTION
[83,plus,5,1],
[84,from,5,5],
[85,plus,5,1],
[86,from,5,5],
[87,to,4,5],
[88,cst,0,46],
# TOINDIRECT INSTRUCTION
[89,from,4,4],
[90,plus,4,1],
[91,plus,5,1],
[92,from,5,5],
[93,to,4,5],
[94,cst,0,46],
# IFZERO INSTRUCTION
[95,from,4,4],
[96,ifzero,4,98],
[97,cst,0,46],
[98,to,1,5],
[99,cst,0,49],
# END
[100,plus,9,1],
[101,from,9,9],
[102,plus,1,9].
```

.

Complexity and introspection coefficient of $(\boldsymbol{U}, \boldsymbol{code})$

Complexity of our pair for the indirect addressing machine

• On particular examples we have obtained the following results:

x	$\lvert orbit(P, x)\rvert$	$\lvert orbit(U,$ $code(P)\cdot x)\rvert$	$\lvert orbit(U,$ $code(U)\cdot code(P)\cdot x)\rvert$	$\frac{\lvert orbit(U,\, code(U)\cdot code(P)\cdot x)\rvert}{\lvert orbit(U,\, code(P)\cdot x)\rvert}$
ε	12	2 372	72 110	30.40
`o`	16	2 473	74 758	30.23
`oi`	31	2 860	84 916	29.69
`oiz`	35	2 961	87 564	29.57
`oizo`	50	3 348	97 722	29.19

• where P is a reversing program of 21 instructions, with $\lvert code(P)\rvert = 216$, such that, for all $n \geq 0$ one obtains $out(P,\, a_1a_2 \ldots a_n) = a_n \ldots a_2a_1$, with the a_i's taken from $\{\texttt{o}, \texttt{i}, \texttt{z}\}$. For information, $\lvert code(U)\rvert = 1042$.

Introspection coefficient of our pair for the indirect addressing machine

$$\boxed{\lim_{n \to \infty} \frac{|orbit(U,\ code(U)^{n+1} \cdot x)|}{|orbit(U,\ code(U)^{n} \cdot x)|} = 26,27}$$

Matrix A and vector B for U

$$A = \begin{bmatrix}
4 & 4 & 4 & 4 & 4 & 4 & 4 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\
9 & 11 & 10 & 11 & 11 & 8 & 9 & 10 & 12 & 11 & 12 & 12 & 9 & 10 & 10 & 12 & 11 & 12 & 12 & 9 & 10 \\
1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 \\
6 & 8 & 8 & 8 & 8 & 6 & 7 & 7 & 9 & 9 & 9 & 9 & 7 & 8 & 7 & 9 & 9 & 9 & 9 & 7 & 8 \\
2 & 2 & 2 & 2 & 2 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\qquad
B = \begin{bmatrix}
1967 \\ 4085 \\ 310 \\ 2041 \\ 1329 \\ 708 \\ 311 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0
\end{bmatrix}$$

.

Conclusion

Open problem

• Given a first universal pair $(U, code)$ for a Turing machine M, by "cheating", it is possible to construct a second universal pair $(U', code')$ for M with introspection coefficient equal to 1.

• A first way of "cheating" consists of taking $U' = U$ and

$$code'(P) = \begin{cases} \varepsilon, & \text{if } P = U, \\ code(P), & \text{if } P \neq U. \end{cases}$$

Then

$$\frac{|orbit(U',\ code(U')^{n+1} \cdot x)|}{|orbit(U',\ code(U')^{n} \cdot x)|} = \frac{orbit(U',\ x)}{orbit(U',\ x)} = 1$$

and $(U', code')$ is a universal pair with introspection coefficient equal to 1.

Open problem, next

• A second way of "cheating" consists in keeping $code' = code$ and constructing a program U', which, after having erased as many times as possible a given word z occurring as prefix of the input, behaves as U on the remaining input. According to the recursion theorem it is possible to take z equal to $code(U')$ and thus to obtain a universal program U' such that, for all $y \in \Sigma^\star$ having not $code(U)'$ as prefix,

$$|orbit(U',\ code(U')^n \cdot y)| = nk_1 + k_2(y), \tag{3}$$

where k_1 and $k_2(y)$ are positive integers, with k_1 not depending on y. Thus we have :

$$\frac{|orbit(U',\ code(U')^{n+1} \cdot y)|}{|orbit(U',\ code(U')^n \cdot y)|} = \frac{|orbit(U,\ x)| + (n+1)k_1 + k_2(y)}{|orbite(U,\ x)| + nk_1 + k_2(y)} =$$

$$1 + \frac{k_1}{|orbit(U,\ x)| + k_2(y) + nk_1}.$$

By letting n tend toward infinity we obtain an introspection coefficient equal to 1 for the pair $(U', code')$.

• **Open problem** How to express in the definition of the introspection coefficient that the function $code$ and the program U should not distinguish the case $P = U$ from the case $P \neq U$.

A joke

• Given the fact that a Turing machine with a universal program models the way our brain operates,
• given the fact that we should think about what we intend to say,
• given the fact that our best introspection coefficient of a universal Turing program is 3673,
• don't say:
you should count to 10 before you speak,
• but say:
you should count to 3673 before you speak.