Complexity of Universal Programs

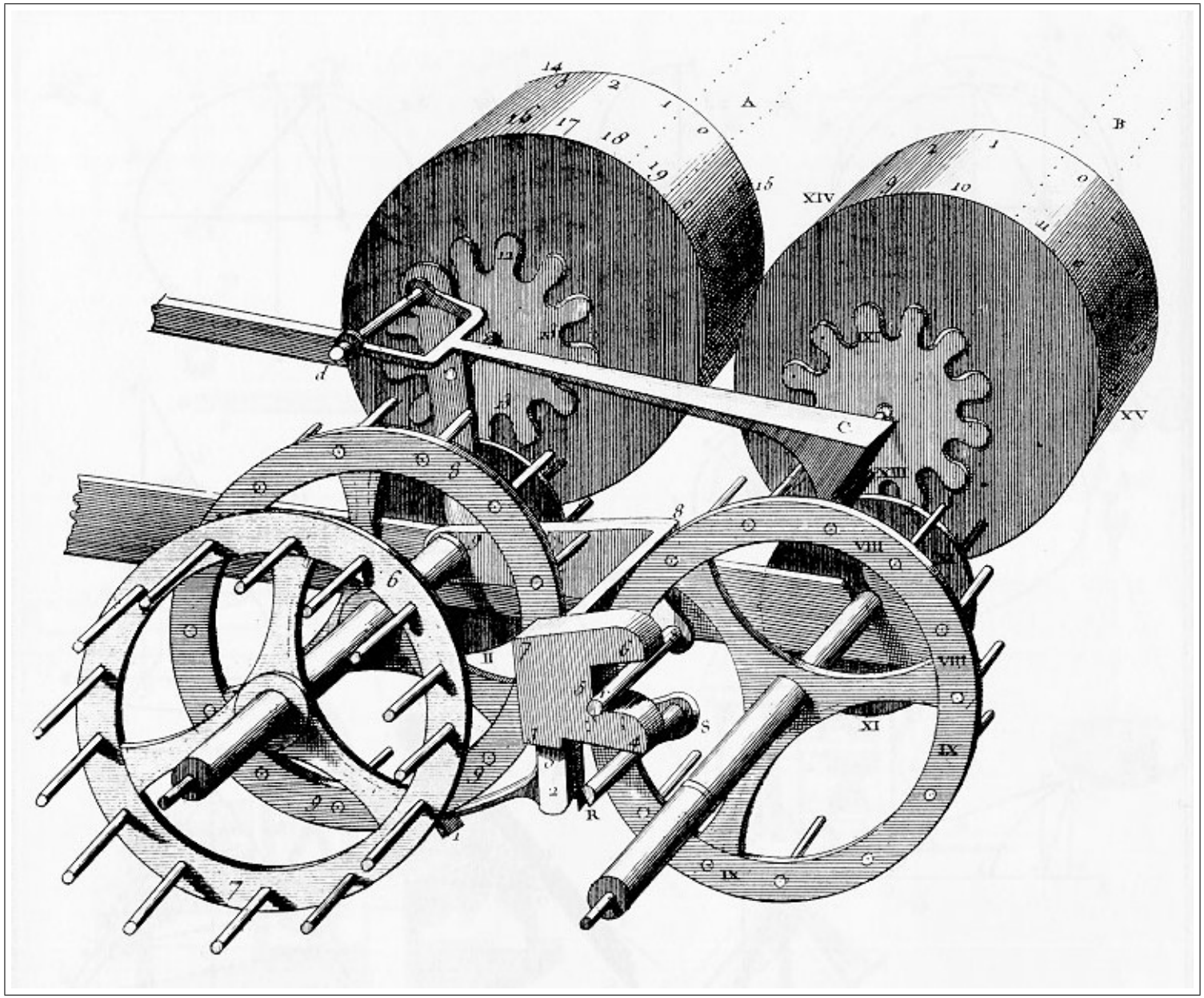
Alain Colmerauer

St. Petersburg, September 22, 2004

Laboratoire d'Informatique de Marseille
CNRS, Universités de Provence
et de la Méditerranée

Contents

What is a machine and a program?

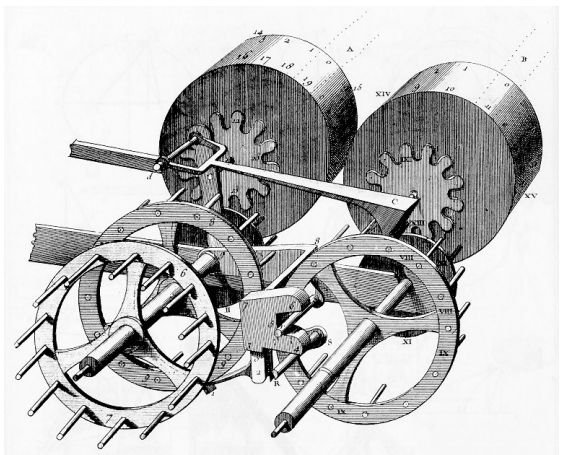

A
B
C
XIV
XV
VIII
XI
IX
S
R

A machine programmed for translating

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

`colorless␣green␣ideas␣sleeep␣furiously`

$\Downarrow$



Translating program

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

$\Downarrow$

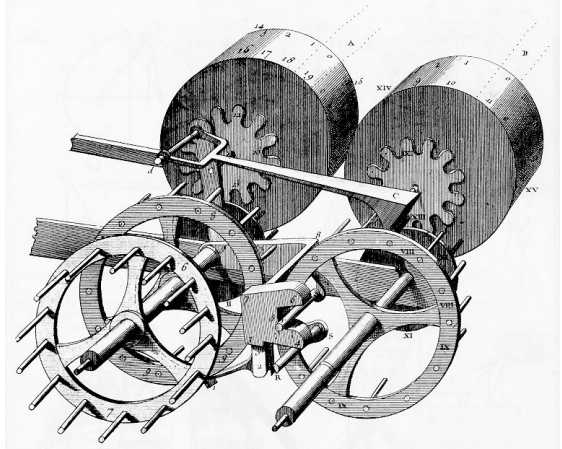
`des␣idees␣vertes␣incolores␣dorment␣furieusement`

A machine programmed for multiplying

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

`26010x37979721`

$\Downarrow$



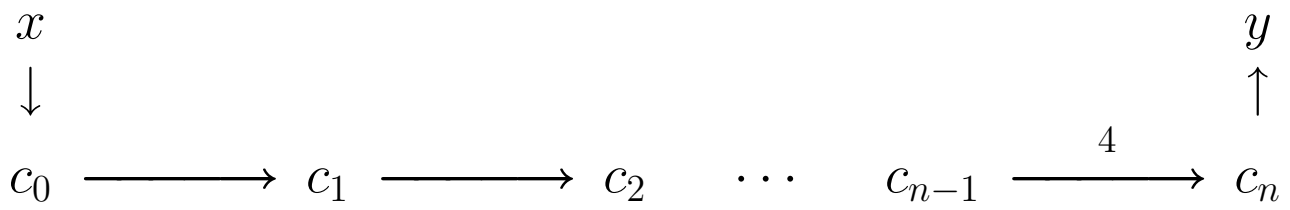
Multiplying program

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

$\Downarrow$

`9876543210`

A programmed machine is a dynamic system



Formal definition of a programmed machine

• A *programmed machine* is an ordered pair $(\mathcal{M}, P)$, where $\mathcal{M}$ is a *machine* and P a *program* for $\mathcal{M}$.

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

Σ, *the alphabet* of $\mathcal{M}$, 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 $\Sigma^\star$;

$\mathcal{I}$, is a countable set of *instructions*, an *intruction* being a set of *compatibles* transitions, that is to say, whose first components are all distinct.

• A *program* P for a machine $\mathcal{M}$ is a finite subsets of the instructions set $\mathcal{I}$ of $\mathcal{M}$, such that the transitions of $\cup P$ are compatible. The set of configurations occurring in P is denoted by C_P. A configuration c of $\mathcal{M}$ is *final for* P, if there exists no transition of $\cup P$ starting with c.

The main functions of a programmed machine

Let $\mathcal{M} = (\Sigma, C, \alpha, \omega, \mathcal{I})$ be a machine and P a program for $\mathcal{M}$. For all words x on Σ and configurations c, c' of C:

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

$$out_{\mathcal{M}}(P,\, x) = \begin{cases} \nearrow, & \text{if } orbit(P,\, x) \text{ is infinite,} \\ \omega(c_n), & \text{if } orbit(P,\, x) \text{ ends with } c_n \end{cases}$$

$$track_{\mathcal{M}}(P,\, c,\, c') = \begin{cases} \text{the shortest sequence of the form} \\ (c_0, c_1)\,(c_1, c_2)\,(c_2, c_3)\,\ldots\,(c_{n-1}, c_n), \\ \text{with } c = c_0, \text{ each } (c_i, c_{i+1}) \in \cup P,\ c_n = c', \\ \nearrow, \text{ if this shortest sequence does not exist,} \end{cases}$$

$$track_{\mathcal{M}}(P,\, x) = \begin{cases} \text{the longest sequence of the form} \\ (c_0, c_1)\,(c_1, c_2)\,(c_2, c_3)\,\ldots \\ \text{with } orbit(P,\, x) = c_0, c_1, c_2, \ldots. \end{cases}$$

$$cost_{\mathcal{M}}(P,\, x) = \begin{cases} \infty, & \text{if } track(P,\, x) \text{ is infinite,} \\ |track(P,\, x)|, & \text{if } track(P,\, x) \text{ is finite,} \end{cases}$$

Simulation of a machine by another

• Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two machines of same alphabet Σ and let $\mathcal{P}_1$ and $\mathcal{P}_2$ be the sets of their programs.

• **Definition** Two programmed machines of the form $(\mathcal{M}_1, P_1)$ and $(\mathcal{M}_2, P_2)$ are *equivalent* if, for all $x \in \Sigma^\star$,

$$\begin{aligned} out_{\mathcal{M}_1}(P_1,\, x) &= out_{\mathcal{M}_2}(P_2,\, x), \\ cost_{\mathcal{M}_1}(P_1,\, x) &= cost_{\mathcal{M}_2}(P_2,\, x). \end{aligned}$$

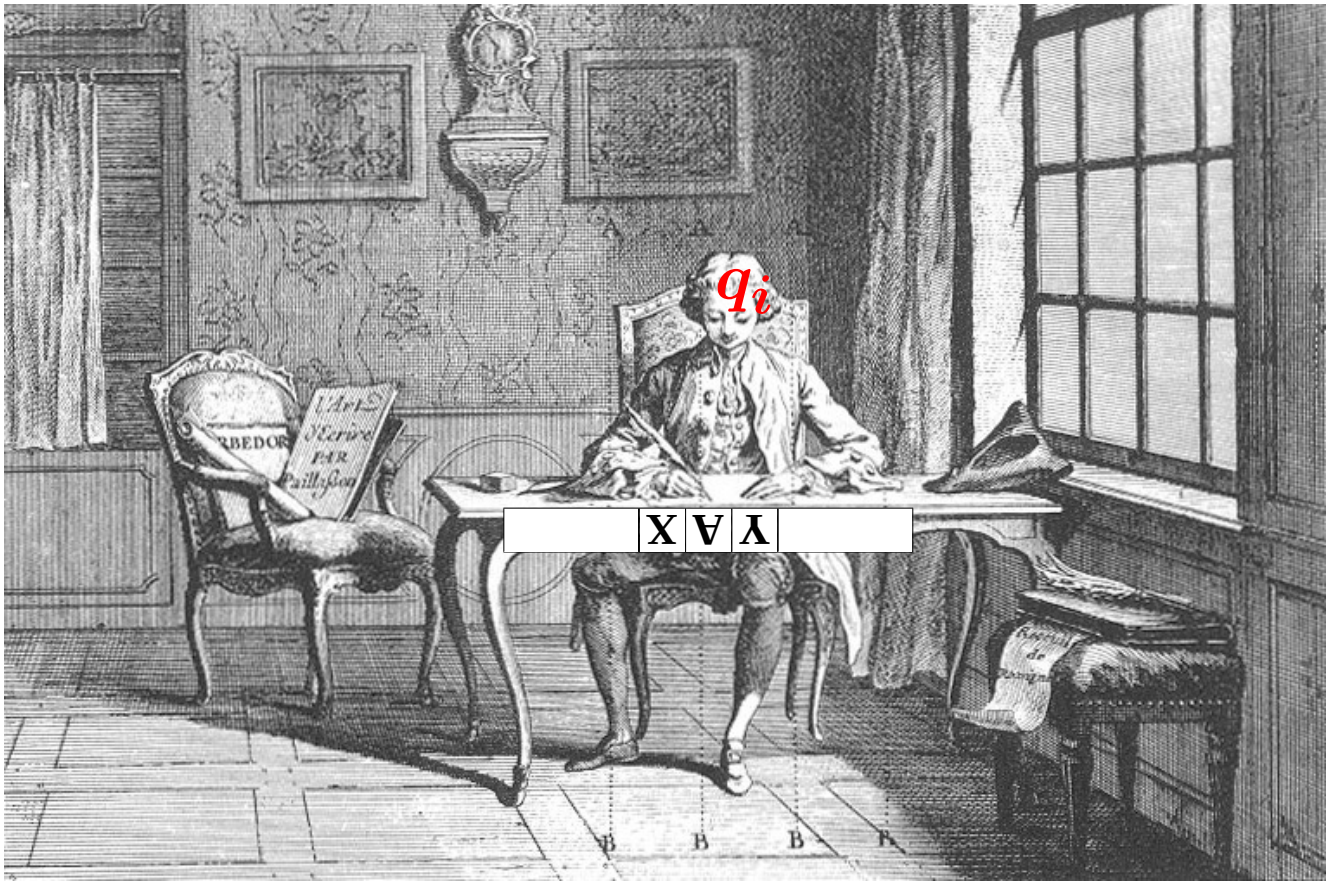
• **Definition** The program transformation $f_1 : \mathcal{P}_1 \to \mathcal{P}_2$ *simulates* $\mathcal{M}_1$ on $\mathcal{M}_2$ if, for all $P_1 \in \mathcal{P}_1$, the programmed machines $(\mathcal{M}_1, P_1)$ and $(\mathcal{M}_2, f_1(P_1))$ are equivalent.

Turing machine

Alan Turing

Alan M. Turing, "On computable numbers with an application to the Entsheidungsproblem", *Proc. London Math. Society*, 2, 42, pp. 230–265, 1936.

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

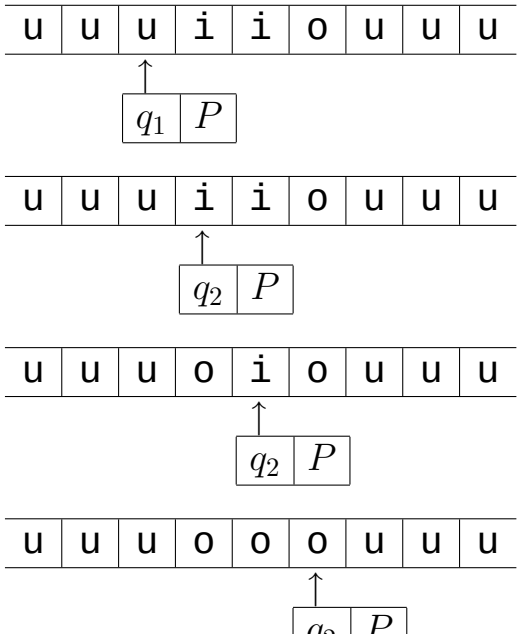
• With program

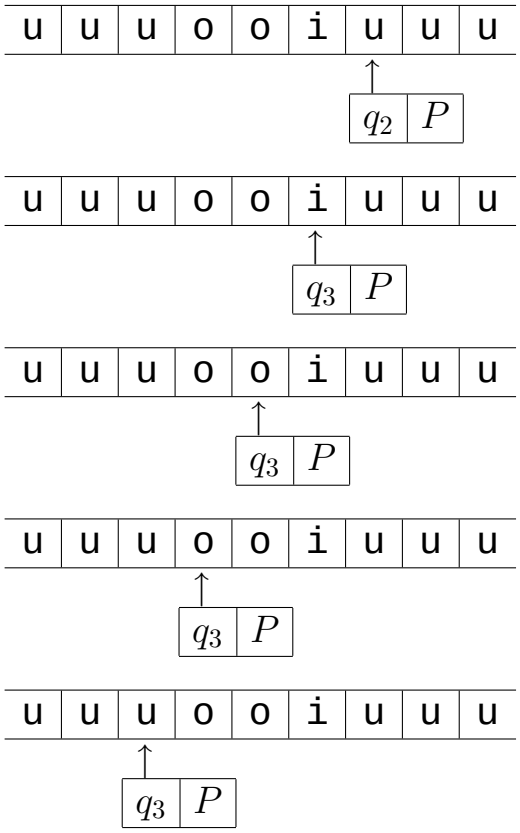
$$P = \left\{ \begin{array}{l} [q_1,\ \texttt{uu}R,\ q_2], \\ [q_2,\ \texttt{oi}R,\ q_2], \\ [q_2,\ \texttt{io}R,\ q_2], \\ [q_2,\ \texttt{uu}L,\ q_3], \\ [q_3,\ \texttt{oo}L,\ q_3], \\ [q_3,\ \texttt{ii}L,\ q_3] \end{array} \right\},$$

• and input

iio,

• the machine moves through the sequence of configurations





• and produces the result

ooi

Formal definition of a Turing machine

A Turing machine has a 5-tuple of the form $(\Sigma, C, \mathcal{I}, \alpha, \omega)$ where,

- $\Sigma \neq \Sigma_{\mathsf{u}}$, with $\Sigma_{\mathsf{u}} = \Sigma \cup \{\mathsf{u}\}$,
- C is the set of 4-tuples of the form $[q_i, \cdot x, a, y\cdot]$, with q_i being any state, x, y taken from $\Sigma^\star_{\mathsf{u}}$ and a taken from Σ_{u},
- $\alpha(x) = [q_1, \varepsilon, \mathsf{u}, x]$, for all $x \in \Sigma^\star$,
- $\omega([q_i, \cdot x, a, y\cdot])$ is the longest element of $\Sigma^\star$ beginning $y\cdot$,
- $\mathcal{I}$ is the set of instructions denoted and defined, for all sates q_i, q_j and elements a, b of Σ_{u}, by
  $[q_i,\ abL,\ q_j] \stackrel{\text{def}}{=} \{([q_i, \cdot xc, a, y\cdot],\ [q_j, \cdot x, c, by\cdot]) \mid (x, c, y) \in E\}$,
  $[q_i,\ abR,\ q_j] \stackrel{\text{def}}{=} \{([q_i, \cdot x, a, cy\cdot],\ [q_j, \cdot xb, c, y\cdot]) \mid (x, c, y) \in E\}$,
  with $E = \Sigma^\star_{\mathsf{u}} \times \Sigma_{\mathsf{u}} \times \Sigma^\star_{\mathsf{u}}$.

Turing machine with internal direction

Program example

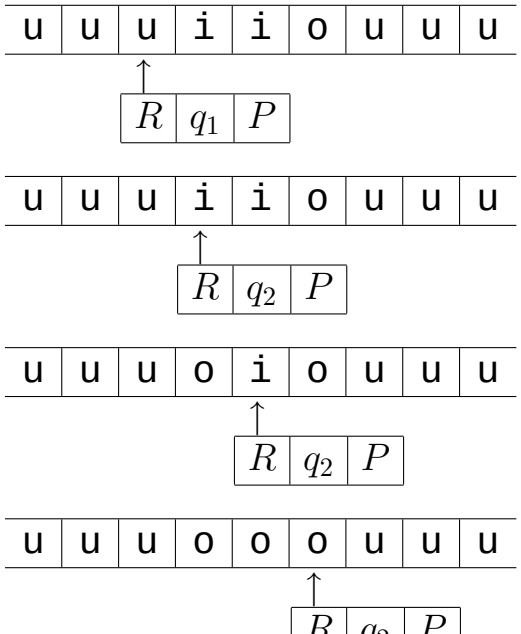
• With program

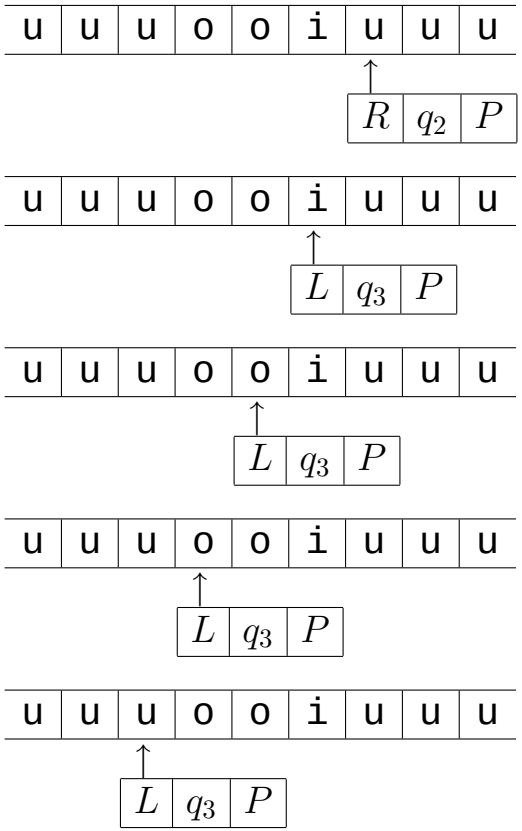
$$P = \left\{ \begin{array}{l} [q_1,\ \texttt{uu}+,\ q_2], \\ [q_2,\ \texttt{oi}+,\ q_2], \\ [q_2,\ \texttt{io}+,\ q_2], \\ [q_2,\ \texttt{uu}-,\ q_3], \\ [q_3,\ \texttt{oo}+,\ q_3], \\ [q_3,\ \texttt{ii}+,\ q_3] \end{array} \right\},$$

• and input

$$\texttt{iio},$$

• the machine moves through the sequence of configurations



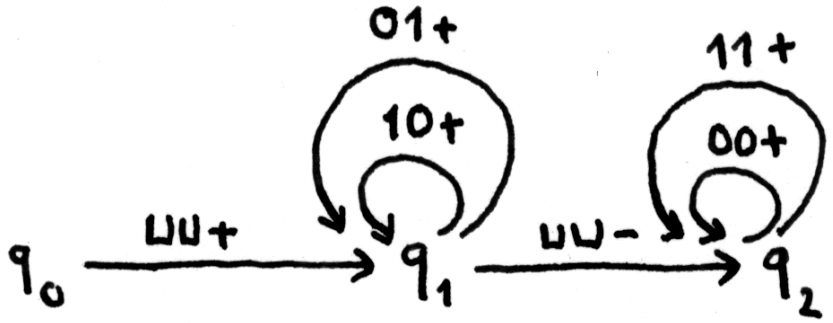


• and produce the result

$$\texttt{ooi}$$

Graph of a program

$$\left\{\begin{array}{l}[q_1, \texttt{uu}+, q_2],\\ [q_2, \texttt{oi}+, q_2],\\ [q_2, \texttt{io}+, q_2],\\ [q_2, \texttt{uu}-, q_3],\\ [q_3, \texttt{oo}+, q_3],\\ [q_3, \texttt{ii}+, q_3]\end{array}\right\}$$



Formal definition of a Turing machine with internal direction

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

- $\Sigma \neq \Sigma_{\mathsf{u}}$, avec $\Sigma_{\mathsf{u}} = \Sigma \cup \{\mathsf{u}\}$,
- 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},
- $\alpha(x) = [R, q_1, \varepsilon, \mathsf{u}, x]$, for all $x \in \Sigma^{\star}$,
- $\omega([d, q_i, \cdot x, a, y\cdot])$ is the longest element of $\Sigma^{\star}$ beginning $y\cdot$,
- $\mathcal{I}$ 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 $F = \Sigma^{\star}_{\mathsf{u}} \times \Sigma_{\mathsf{u}} \times \Sigma^{\star}_{\mathsf{u}}$.

Relationship between the two types of Turing machines

Program transformations f_1 and f_2

• Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be machines of same alphabet Σ, the first one of type Turing and the second one of type Turing with internal direction and let $\mathcal{P}_1$ and $\mathcal{P}_2$ denote the sets of their programs.

• **Property** The program transformations $g_1 : \mathcal{P}_1 \to \mathcal{P}_2$ and $g_2 : \mathcal{P}_2 \to \mathcal{P}_1$, defined below, strongly simulate $\mathcal{M}_1$ on $\mathcal{M}_2$ and $\mathcal{M}_2$ on $\mathcal{M}_1$:

$$g_1(P_1) \stackrel{\text{def}}{=} \{[q_{2i-h},\, abs,\, q_{2j-k}] \mid [q_i,\, abd,\, q_j] \in P_1 \text{ and } (h,k,d,s) \in E\},$$
$$g_2(P_2) \stackrel{\text{def}}{=} \{[q_{2i-h},\, abs,\, q_{2j-k}] \mid [q_i,\, abd,\, q_j] \in P_2 \text{ and } (h,k,s,d) \in E\},$$

with

$$E \stackrel{\text{def}}{=} \begin{Bmatrix}(1,\,1,\,+,R)\\(1,\,0,\,-,L)\end{Bmatrix} \cup \begin{Bmatrix}(0,\,1,\,-,R)\\(0,\,0,\,+,L)\end{Bmatrix}.$$

• **Theorem 1** The program transformations $f_1 : \mathcal{P}_1 \to \mathcal{P}_2$ and $f_2 : \mathcal{P}_2 \to \mathcal{P}_1$, defined, for $\ell = 1$ and $\ell = 2$, by $f_\ell = clean \circ g_\ell$, with g_ℓ defined above, simulate $\mathcal{M}_1$ on $\mathcal{M}_2$ and $\mathcal{M}_2$ on $\mathcal{M}_1$ and are such that

$$f_2 \circ f_1 \circ f_2 \circ f_1 = f_2 \circ f_1, \quad f_1 \circ f_2 \circ f_1 \circ f_2 = f_1 \circ f_2.$$

Arithmetic machine with indirect adressing

Arithmetic machine with indirect addressing

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

- $\Sigma = \{c_1, \ldots, c_m\}$,
- 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 satisfting the condition that n is the greatest integer such that $r_{r_1}, \ldots, r_{r_1+n}$ are elements of $\{1, \ldots, m\}$,
- $\mathcal{I}$ is the set of instructions denoted and defined, for all natural integers i, j, k, by:

$$
\begin{aligned}
[i, cst, j, k] &\stackrel{\text{def}}{=} \{(r,\ s') \in C^2 \,|\, r_0 = i,\ s_j = k \text{ and } s_i = r_i \text{ elsewhere}\},\\
[i, plus, j, k] &\stackrel{\text{def}}{=} \{(r,\ s') \in C^2 \,|\, r_0 = i,\ s_j = r_j + r_k \text{ and } s_i = r_i \text{ elsewhere}\},\\
[i, sub, j, k] &\stackrel{\text{def}}{=} \{(r,\ s') \in C^2 \,|\, r_0 = i,\ s_j = r_j \div r_k \text{ and } s_i = r_i \text{ elsewhere}\},\\
[i, from, j, k] &\stackrel{\text{def}}{=} \{(r,\ s') \in C^2 \,|\, r_0 = i,\ s_j = r_{r_k} \text{ and } s_i = r_i \text{ elsewhere}\},\\
[i, to, j, k] &\stackrel{\text{def}}{=} \{(r,\ s') \in C^2 \,|\, r_0 = i,\ s_{r_j} = r_k \text{ and } s_i = r_i \text{ elsewhere}\},\\
[i, ifzero, j, k] &\stackrel{\text{def}}{=} \{(r,\ s') \in C^2 \,|\, r_0 = i,\ s_0 = \begin{cases} r_k, & \text{if } r_j = 0,\\ r_0 + 1, & \text{if } r_j \neq 0 \end{cases} \text{ and } s_i = r_i \text{ elsewhere}\},
\end{aligned}
$$

with s' equal to s except that $s'_0 = s_0 + 1$ and with $r_j \div r_k = \max\{0, r_j - r_k\}$.

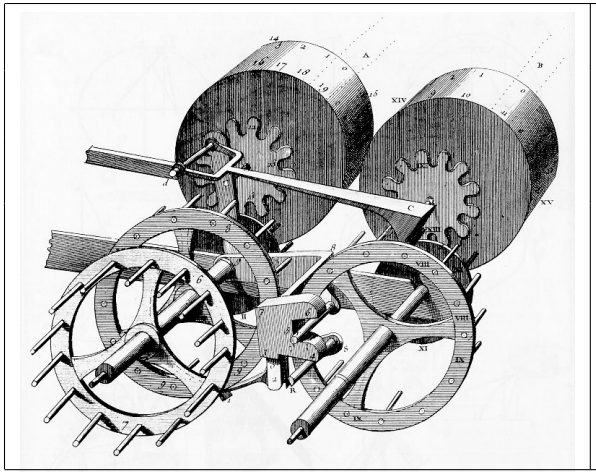
Universal program and coding

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

$\Downarrow$

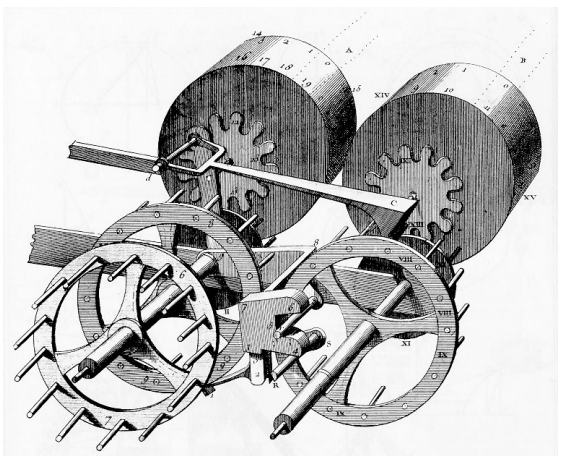
`des␣idees␣vertes␣incolores␣dorment␣furieusement`

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

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

$\Downarrow$

`9876543210`

Definition of a universal pair

• Let $\mathcal{M} = (\Sigma, C, \alpha, \omega, \mathcal{I})$ be a machine and let us code each program P for $\mathcal{M}$ 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 $\mathcal{M}$, if, for all programs P of $\mathcal{M}$ and for all $x \in \Sigma^\star$,

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

• If in the above 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 $x \in \Sigma^\star$,

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

Example of an 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 tranlated, go to 6.
4. Translate the i^{th} word.
5. Go to 2.
6. Stop.

Input: English sentence

```
colorless␣
green␣idea
s␣sleep␣fu
riously
```

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 succeed, go to 2.
10. Give to n the value p.
11. Go to 3.
12. Stop.

Input: translating program + English sentence

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␣furiously
```

Same 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.

Input: universal program + translating program + English sentence

1. Give ...
2. Increase ...
3. Find ...
4. If ...
5. If ...
6. Execute ...
7. Go to 2.
8. Perform ...
9. If the test ...
10. Give ...
11. Go to 3.
12. Stop.

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

```
colorless␣green␣ideas␣sleep␣furiously
```

Complexity and introspection coefficient

Definition of complexity and introspection coefficient

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

Definition Given a program P for $\mathcal{M}$ and a word x on Σ such that $cost(P,\, x) \neq \infty$, the *complexity* of $(U,\ code)$ is the real number denoted and defined by

$$\boxed{\lambda(P,\, x) = \frac{cost(U,\ code(P) \cdot x)}{cost(P,\, x)}.}$$

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

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

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

Keeping the same complexity and introspection coefficient

• Let $\mathcal{M}_1$ be a Turing machine and $\mathcal{M}_2$ a Turing machine with internal direction and same alphabet Σ.

• Let f_1 and f_2 be the program transformations of Theorem 1.

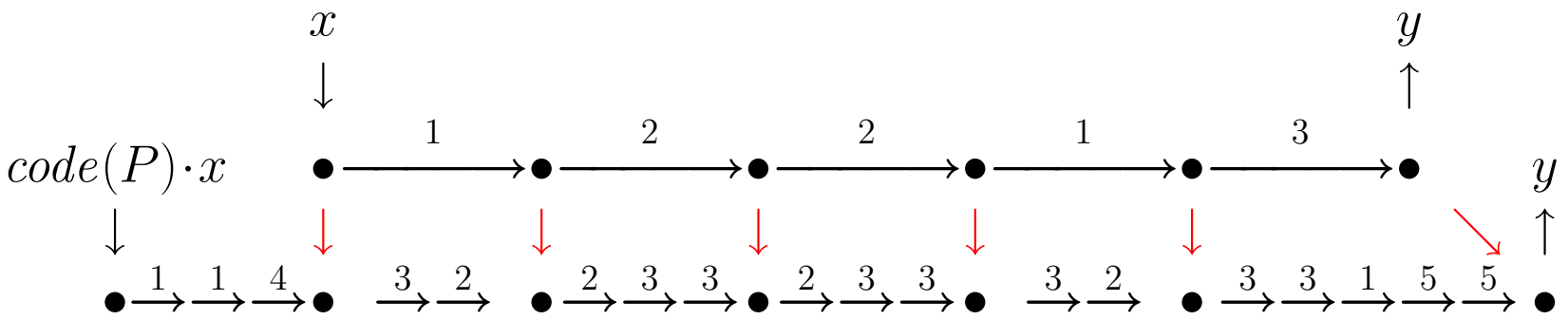
• **Property** If $P_1 = f_2(P_2)$ or $P_2 = f_1(P_1)$ and if $(U_2, code_2)$ is a universal pair for $\mathcal{M}_2$ then the pair $(U_1, code_1) \stackrel{\text{def}}{=} (f_2(U_2),\ code_2 \circ f_1)$

1. is universal for $\mathcal{M}_1$,
2. has a complexity $\lambda_1(P_1, x)$, not defined or equal to the complexity $\lambda_2(P_2, x)$ of the pair $(U_2, code_2)$, depending whether the real number $\lambda_2(P_2, x)$ is not defined or defined,
3. admits the same introspection coefficient as $(U_2, code_2)$ or does not admit an introspection coefficient, depending whether $(U_2, code_2)$ admits or does not admit one,
4. is such that $code_1(P_1) = code_2(P_2)$.

Existence and value of the introspection coefficient

Toward the labeling hypothesis

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



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

Labeling hypothesis

Let $\mathcal{M} = (\Sigma, C, \alpha, \omega, \mathcal{I})$ be a machine and $(U, code)$ a universal pair for $\mathcal{M}$

Hypothesis There exist

- *a synchronization function* $\Phi : C_U \to C_U$, with $\Phi(c)$ final for U when c is final for U,
- *a labeling function* $\mu : (\cup U)^\star \to (1..n)^\star$ with n a positif integer and μ being surjective, such that $\mu(t_1 \dots t_k) = \mu(t_1) \dots \mu(t_k)$, for all $t_1 \dots t_k \in (\cup U)^\star$,
- *an initial sequence of labels* $\delta \in (1..n)^\star$, independent of x, such that

$$\delta = \mu(track(U,\, \alpha(code(U) \cdot x),\, \Phi(\alpha(x)))), \text{ for all } x \in \Sigma^\star,$$

- *a label rewriting* $\varphi : 1..n \to (1..n)^\star$ such that, for all $(c, c') \in \cup U$,

$$\varphi(\mu(c, c')) = \mu(track(U,\, \Phi(c),\, \Phi(c'))).$$

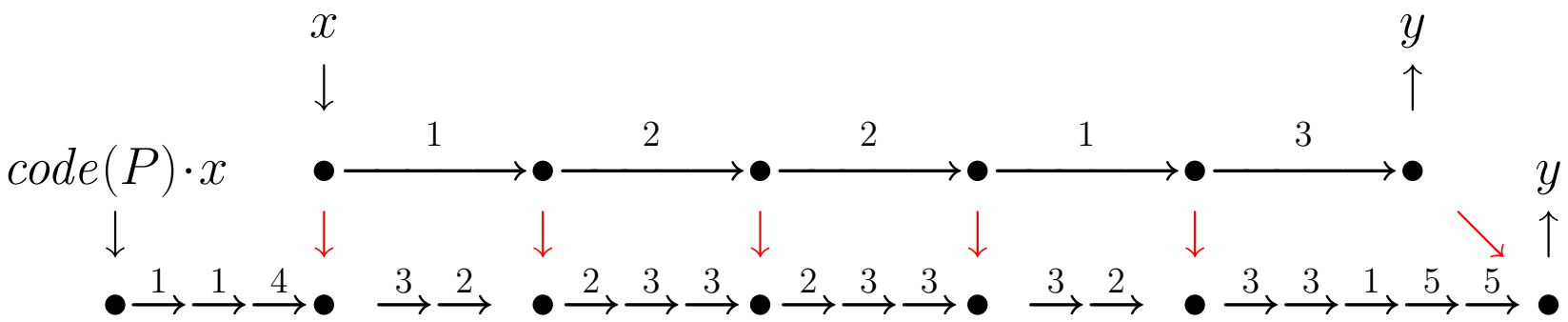
Vector B and Matrix A related to δ and φ

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

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

Vector B and Matrix A related to δ and φ, 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,

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

we have

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

• **Thus**

$$cost(U,\, code(U)^k \cdot x) = ||X^{(k)}||, \quad \text{and} \quad \frac{cost(U,\, code(U)^{k+1} \cdot x)|}{cost(U,\, code(U)^k \cdot x)} = \frac{||X^{(k+1)}||}{||X^{(k)}||}.$$

Main theorem

Here B and A are the vector and the matrix associated with β and φ.

Theorem 2 If the matrix A admits a real eigenvalue λ, whose multiplicity is equal to 1 and whose value is strictly greater than 1 and strictly greater then the absolute value λ' of the other eigenvalues of A then:

- the limit matrix $\bar{A} = \lim_{n\to\infty} (\frac{1}{\lambda}A)^n$ exists, and is not everywhere zero,
- if $||\bar{A}B|| \neq 0$, the introspection coefficient of U exists and is equal to λ,
- if ℓ is a real numbers such that $\lambda' < \ell < \lambda$ and the X_i's are column vectors defined by

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

then, when $n \to \infty$,

$$||X_n|| \to \begin{cases} 0, & \text{if } ||\bar{A}B|| = 0, \\ \infty, & \text{if } ||\bar{A}B|| \neq 0. \end{cases}$$

Universal pair $(U_2, code_2)$ for Turing machine with internal direction

Presentation of $(U_2, code_2)$

• $(U_2, code_2)$ is a universal pair for the Turing machine with internal direction $\mathcal{M}_2$ with alphabet

$$\Sigma = \{\mathtt{o}, \mathtt{i}, \mathtt{z}\}.$$

• In order to assign a position to each instruction $[q_i, abs, q_j]$ of a program, we introduce the numbers:

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

defined for all positive integers i and symbols $a \in \{\mathtt{u}, \mathtt{o}, \mathtt{i}, \mathtt{z}\}$.

Coding function $code_2$

• Let P_2 be a program for $\mathcal{M}_2$ and let $P_2' = f_1(f_2(P_2))$. We take $code_2(P_2) = code_2'(P_2')$, with $code_2'(P_2')$ the word on $\{\mathtt{o}, \mathtt{i}, \mathtt{z}\}$:

$$\mathtt{z}I_{4n}\mathtt{z}\ldots\mathtt{z}I_{k+1}\mathtt{z}I_k\mathtt{z}I_{k-1}\mathtt{z}\ldots\mathtt{z}I_1\mathtt{zoi}\ldots\mathtt{izz},$$

where n is the number of states of P_2', and the size of the *shuttle* $\mathtt{oi}\ldots\mathtt{iz}$ equals the longest size of the I_k's minus 5,

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

$$I_{\pi(i,a)} = \begin{cases} \overline{[q_i, abs, q_j]}, & \text{if there exists } b, s, j \text{ with } [q_i, abs, q_j] \in P_2', \\ \mathtt{oi}, & \text{otherwise,} \end{cases}$$

• with,

$$\overline{[q_i, a, b, s, q_j]} = \begin{cases} \mathtt{i}a_m \ldots a_2\mathtt{o}, & \text{if } \pi(j) - h(i,a) > 0, \\ \mathtt{o}a_2 \ldots a_m\mathtt{i}, & \text{if } \pi(j) - h(i,a) < 0, \end{cases}$$

• with a_2a_3 equal to $\mathtt{io}$, $\mathtt{oi}$, $\mathtt{ii}$, depending whether b equals $\mathtt{u}$, $\mathtt{o}$, $\mathtt{i}$, $\mathtt{z}$,

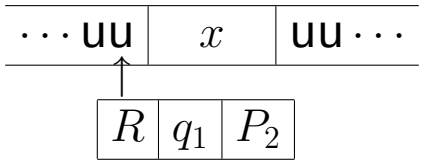
• with $a_4 = \mathtt{o}$ or $a_4 = \mathtt{i}$ depending whether $s = +$ or $s = -$,

• with $\mathtt{i}a_m \ldots a_5$ a binary number ($\mathtt{o}$ for 0 and $\mathtt{i}$ for 1) whose value is equal to

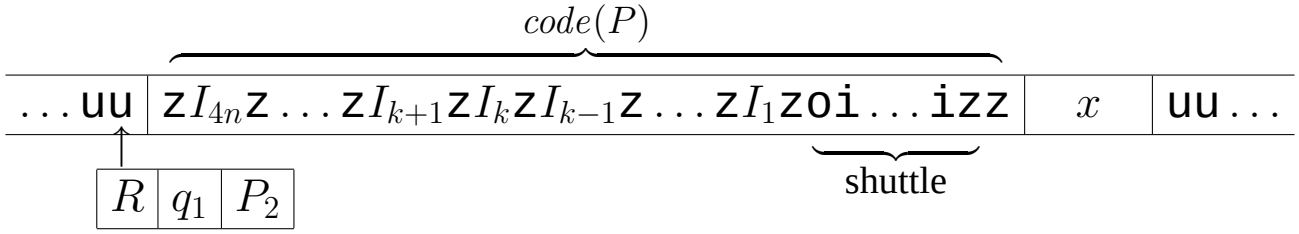
$$|\pi(j) - \pi(i,a)| + \frac{3}{2}.$$

How program U_2 operates

- **Initial configuration of P_2**

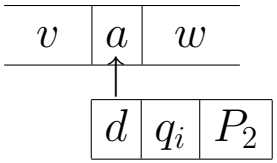


- **Initial corresponding configuration of U_2**

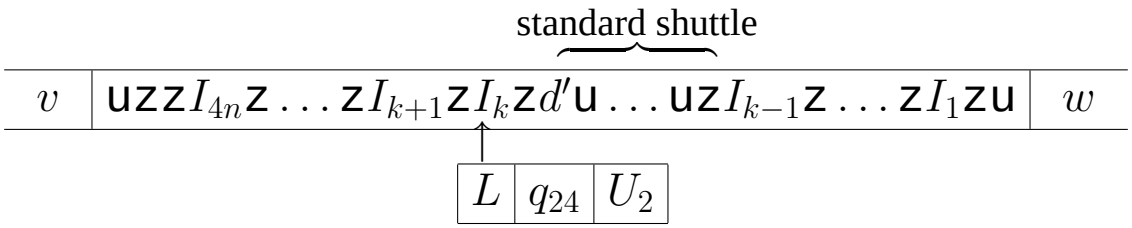


How program U_2 operates, next 1

• **Current configuration of P_2**

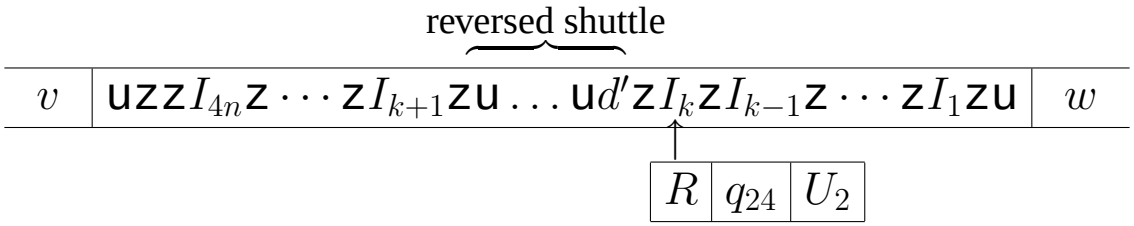


• **Current corresponding configuration of U_2**



(1)

or

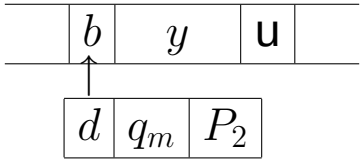


(2)

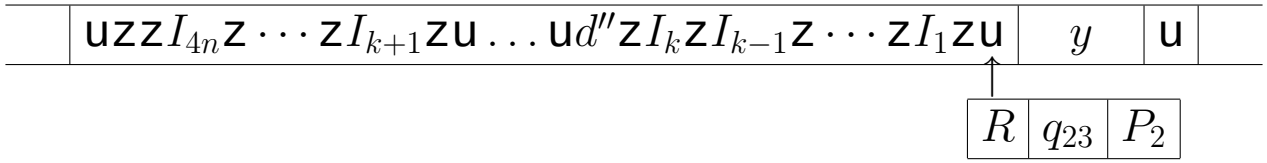
depending whether I_k, with $k = \pi(i, a)$, is in the *standard* form i$a_m \ldots a_2$o or in the *reversed* form i$a_m \ldots a_2$o. The read-write points to a_3 or to the z which follows I_k when I_k is the empty instruction oi. Depending whether d is equal to L or R, the symbol d' is equal to u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

How program U_2 operates, next 2

• **Final configurations of P_2**

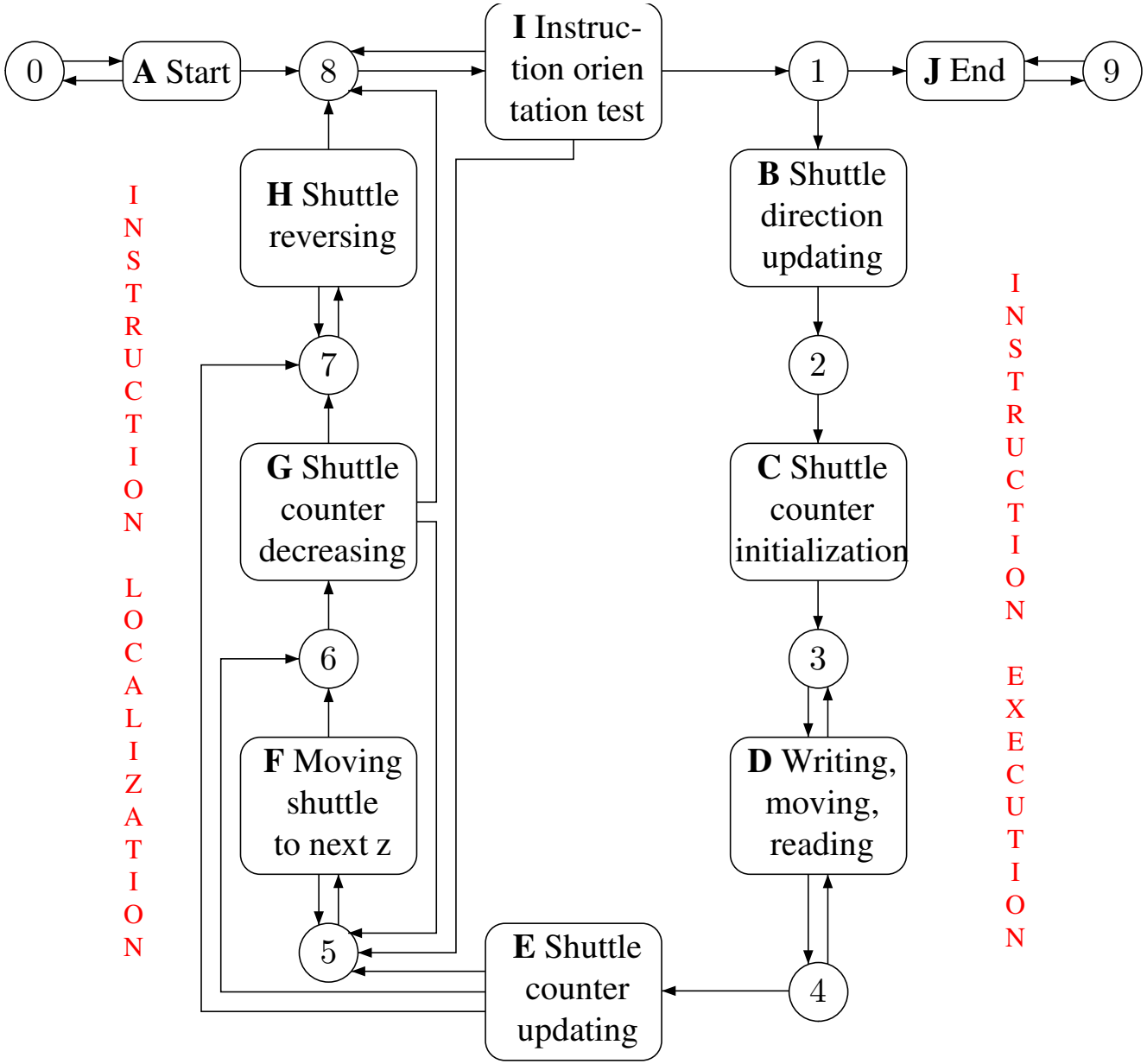


• **Final corresponding configurations of U_2**

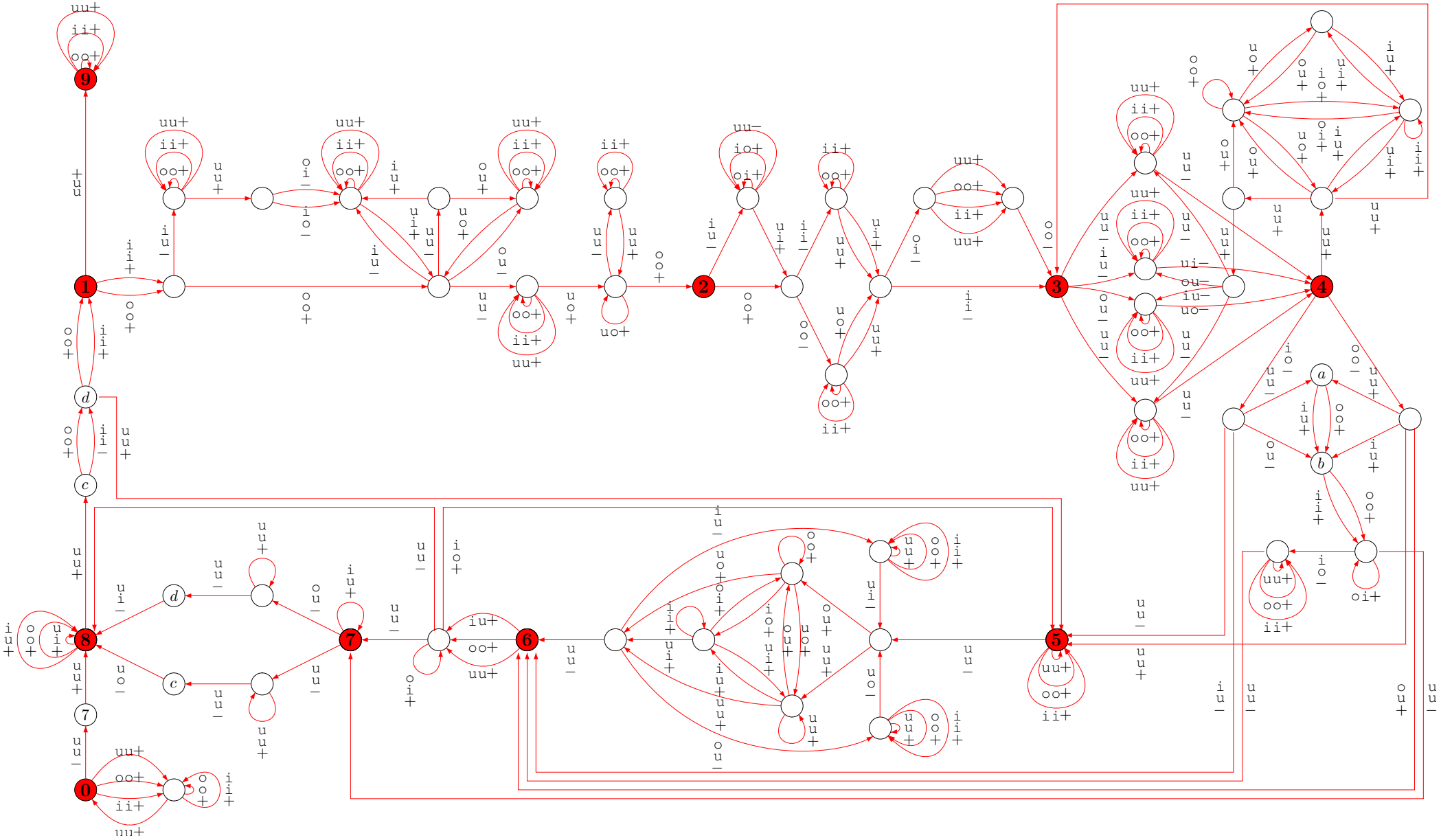


with $I_k = \texttt{oi}$, $k = \pi(m, b)$ and d' equal to o or u, depending whether d equals L or R.

Architecture of program U_2



Graph of program U_2



Listing of program U_2

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_2, 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_2, 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 $(\boldsymbol{U_1}, \boldsymbol{code_1})$ and $(\boldsymbol{U_2}, \boldsymbol{code_2})$

General complexity of $(U_1, code_1)$ and $(U_2, code_2)$

• Let $(U_1, code_1)$ be the universal pair $(f_2(U_2), code_2 \circ f_1)$ for the Turing machine $\mathcal{M}_1$ of same alphabet $\Sigma = \{\mathtt{o}, \mathtt{i}, \mathtt{z}\}$ then $\mathcal{M}_2$.

• U_1 has 361 instructions and 106 states, while U_2 has 184 instructions and 54 states.

• For $\ell = 1$ and $\ell = 2$, and any program P_ℓ for $\mathcal{M}_\ell$,

$$\boxed{|code_\ell(P_\ell)| = \mathcal{O}(n \log n)},$$

where n is the number of states of $clean(P_\ell))$.

• There exists a positive real number k, not dependent on $x \in \Sigma^\star$, such that

$$\boxed{\lambda_\ell(P_\ell, x) \leq n \log^2 n}$$

• If $m = |code_\ell(P_\ell)|$, then there exists a positive real number k, not dependent on $x \in \Sigma^\star$, such that

$$\boxed{\lambda_\ell(P_\ell, x) \leq m \log m}$$

Complexity of $(U_1, code_1)$ and $(U_2, code_2)$ on examples

• For $\ell = 1$ and $\ell = 2$.

x	$cost(P_\ell, x)$	$cost(U_\ell, code_\ell(P_\ell)\cdot x)$	$cost(U_\ell, code_\ell(U_\ell)\cdot code_\ell(P_\ell)\cdot x)$	$\lambda_\ell(P_\ell, x)$	$\lambda_\ell(U_\ell, code_\ell(P_\ell)\cdot x)$
ε	2	5 927	22 974 203	2 963.50	3 876.19
`o`	6	13 335	51 436 123	2 222.50	3 857.23
`oi`	12	23 095	88 887 191	1 924.58	3 848.76
`oiz`	20	35 377	136 067 693	1 768.85	3 846.22
`oizo`	30	49 663	190 667 285	1 655.43	3 839.22

•

• Here P_ℓ, with $P_1 = f_2(P_2)$, is a reversing program of 32 instructions and 9 states, such that, for all $n \geq$,

$$out(P_2,\ a_1a_2 \ldots a_n) = a_n \ldots a_2a_1,$$

with the a_i's taken from $\{\texttt{o}, \texttt{i}, \texttt{z}\}$.

• For $\ell = 1$ and $\ell = 2$, $|code_\ell(P_\ell)| = 265$ and $|code_\ell(U_\ell)| = 1552$.

Introspection coefficient of $(U_1, code_1)$ and $(U_2, code_2)$

• In order to satisfy our Hypothesis, $code_2(U_2) = f_1(f_2(code_2(U_2)))$ and the labeling function μ is defined so that, for all transitions (c_1, c_2) and (c'_1, c'_2) of $\cup\mathcal{M}_2$, the integers $\mu(c_1, c_2)$ and $\mu(c'_1, c'_2)$ are equal if and only if together:

- the states of c_1 and c'_1 are equal,
- the symbols pointed by the read-write heads in c_1 and c'_1 are equal,
- the symbols pointed by the read-write heads in c_2 and c'_2 are equal,
- the directions in c_2 and c'_2 are the same.

• The function Φ is defined, for all configuration of U_2, with $P_2 = U_2$ by,

$$\Phi(c) = \begin{cases} \text{current configuration corresponding to } c, \text{ if } c \text{ is not final for } P_2, \\ \text{final configuration corresponding to } c, \text{ if } c \text{ is final for } P_2. \end{cases}$$

• After having computed the column vector B and the matrix A, using Theorem 2, we have verified that U_2 admits an introspect coefficient and computed its value: for $\ell = 2$ and all words x on Σ such that $cost(P,\, x) \neq \infty$,

$$\boxed{\lim_{n\to\infty} \frac{cost(U_\ell,\, code_\ell(U_\ell)^{n+1} \cdot x)}{cost(U_\ell,\, code(U_\ell)^n \cdot x)} = 3672.98}$$

• This result also holds for $\ell = 1$.

Universal pair $(U_3, code_3)$ for arithmetic machine with indirect adressing

Execution of P_3

• Current configuration

R_0	R_1	R_2	
0	3	2	

$$P_3 = \left\{ \begin{array}{l} [0, plus, 2, 1], \\ [1, cst, 11, 1], \\ [2, from, 5, 2], \\ [3, ifzero, 5, 8], \\ [4, sub, 5, 11], \\ [5, to, 2, 5], \\ [6, plus, 2, 11], \\ [7, cst, 0, 1] \end{array} \right\}$$

• Next current configuration

R_0	R_1	R_2	
1	3	5	

Execution of U_3

• 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_3


```
# 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_3, 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 $(U_3, code_3)$

• **Property** There exists a positive number k such that, for all programs P_3 and word x on Σ, with $cost(P_3,\, x) \neq \infty$,

$$\boxed{\lambda(P_3,\, x) = \frac{cost(U_3,\, code_3(P_3)\cdot x)}{cost(P_3, x)} \leq 35 + k\frac{|code(P)|}{cost(P_3, x)}}$$

• On particular examples we have obtained the following results:

x	$cost(P_3, x)$	$cost(U_3, code_3(P_3)\cdot x)$	$cost(U_3, code_3(U_3)\cdot code_3(P_3)\cdot x)$	$\lambda(P_3, x)$	$\lambda(U_3, code_3(P_3)\cdot x)$
ε	12	2 372	72 110	197.66	30.40
`o`	16	2 473	74 758	154, 56	30.23
`oi`	31	2 860	84 916	92.26	29.69
`oiz`	35	2 961	87 564	84.60	29.57
`oizo`	50	3 348	97 722	66.96	29.19

where P_3 is a reversing program of 21 instructions such that, for all $n \geq 0$

$$out(P_3,\, a_1a_2 \ldots a_n) = a_n \ldots a_2a_1,$$

with the a_i's taken from $\{\texttt{o}, \texttt{i}, \texttt{z}\}$. For information, $|code_3(P_3)| = 216$ and $|code_3(U_3)| = 1042$.

• As introspection coefficient we have obtained:

$$\boxed{\lim_{n\to\infty} \frac{cost(U_3,\, code(U_3)^{n+1}x)}{cost(U_3,\, code(U_3)^{n}\cdot x)} = 26, 27}$$

Matrix A and vector B for U_3

$$A = \begin{bmatrix}
4 & 4 & 4 & 4 & 4 & 4 & 4 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\
9 & 11 & 10 & 11 & 11 & 8 & 9 & 10 & 12 & 11 & 12 & 12 & 9 & 10 & 10 & 12 & 11 & 12 & 12 & 9 & 10 \\
1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 \\
6 & 8 & 8 & 8 & 8 & 6 & 7 & 7 & 9 & 9 & 9 & 9 & 7 & 8 & 7 & 9 & 9 & 9 & 9 & 7 & 8 \\
2 & 2 & 2 & 2 & 2 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$$B = \begin{bmatrix}
1967 \\ 4085 \\ 310 \\ 2041 \\ 1329 \\ 708 \\ 311 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0
\end{bmatrix}$$

Conclusion

Open problem

• Given a first universal pair $(U, code)$ for a Turing machine $\mathcal{M}$, by "cheating", it is possible to construct a second universal pair $(U', code')$ for $\mathcal{M}$ with instrospection 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

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

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,

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

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

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

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

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 $P \neq U$.

Final conclusion

- 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 the introspection coefficient of our universal Turing program is 3673,

- do not think twice before you speak, but 3673 times.