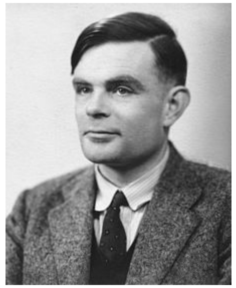
Partie 1 :
Architecture matérielle

Chapitre 3:
Machine de Turing

Alan Turing

Alan Mathison Turing (23 juin 1912 - 7 juin 1954)
Mathématicien anglais
Princeton, puis Cambridge

- Machine de Turing, calculabilité, cryptographie
- Enigma, Colossus, mark I
- Test de Turing (IA)



Machine de Turing: Principe

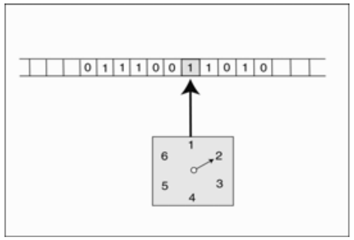
Un ruban séquentiel (données):

- des symboles
- une tête de lecture/écriture (position courante)
- des opérations sur le ruban: déplacement (D, G, N)

Un automate à états finis (programme):

- État courant, état initial, états finaux
- Fonction de transition:

  Etat, entrée -> Etat, sortie, Déplacement



Machine de Turing : formalisation

Formellement:

$$M=(Q, \Gamma, O , \delta)$$

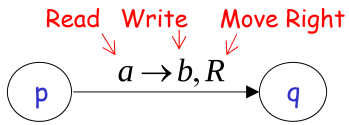
- *Q est un ensemble fini d'états*
  - *Contient l'état initial q_0 et les états finaux $F \subseteq Q$*
- *Γ est l'alphabet du ruban*
  - *contient le symbole blanc (B, ou # ou _ ou …)*
  - *$\Sigma \subseteq \Gamma$ est l'alphabet d'entrée (données initiales)*
- *O est l'ensemble des opérations sur le ruban (tête lecture/écriture)*
  - *Ex: O={L,R,N} , déplacement à gauche (L), à droite (R), aucun (N)*
- *δ : Q-F x $\Gamma \rightarrow$ Q x Γ x O est la fonction de transition (le programme)*
  - *$\delta(q,a) \rightarrow (p,b,R)$*

  Instruction: si état q et lecture a, alors nouvelle état p, écriture b, déplacer tête à droite (R)

  - *Equivalent à la ligne d'un programme:*

  q: if read=a then write b, move R, goto p:

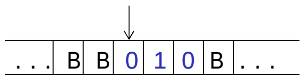
  - *Equivalent au graphe de transition:*



Machine de Turing : exemple

Cette machine calcule le *complementaire* d'un nombre binaire.

Le ruban contient le nombre binaire u sous la forme d'une sequence de caracteres 0,1. Le curseur est sur le 1er chiffre à gauche.



De gauche à droite, chaque 0 (resp. 1) est remplacé par un 1 (resp. 0) .

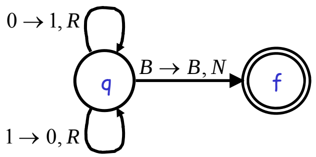
Arret au dernier chiffre à droite (avant le caractere blanc).

- States : Q = {q (start), f (final)}.
- Input symbols: $\Sigma = \{0, 1\}$.
- Tape symbols: $\Gamma = \{0, 1, B\}$.
- Operations: O = {L, R, N}

Transition function (program)

δ: $\{q\} \times \{0,1\} \rightarrow \{q,f\} \times \{0, 1, B\} \times \{L, R, N\}$

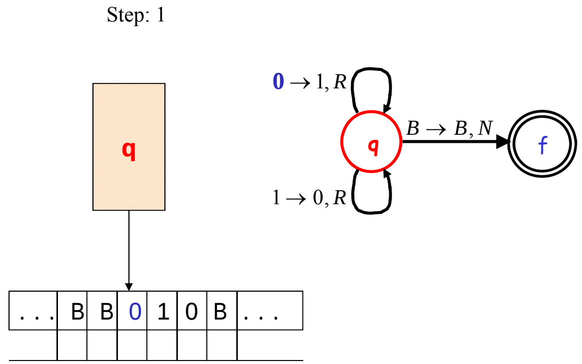
$\delta(q, 0) = (q, 1, R)$.
$\delta(q, 1) = (q, 0, R)$.
$\delta(q, B) = (f, B, N)$.



Transition graph

Machine de Turing : exemple

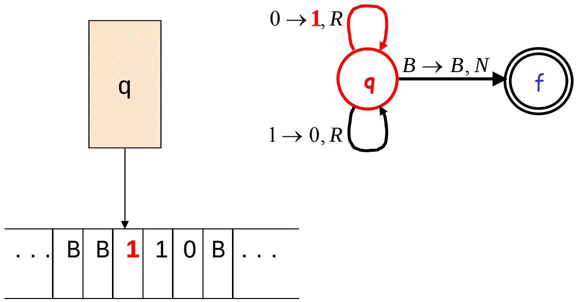
$\delta(q, 0) = (q, 1, R)$
$\delta(q, 1) = (q, 0, R)$
$\delta(q, B) = (f, B, N)$



Machine de Turing : exemple

$\delta(q, 0) = (q, 1, R)$
$\delta(q, 1) = (q, 0, R)$
$\delta(q, B) = (f, B, N)$

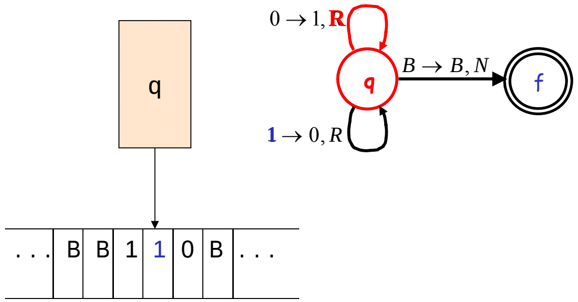
Step: 1



Machine de Turing : exemple

$\delta(q, 0) = (q, 1, R)$
$\delta(q, 1) = (q, 0, R)$
$\delta(q, B) = (f, B, N)$

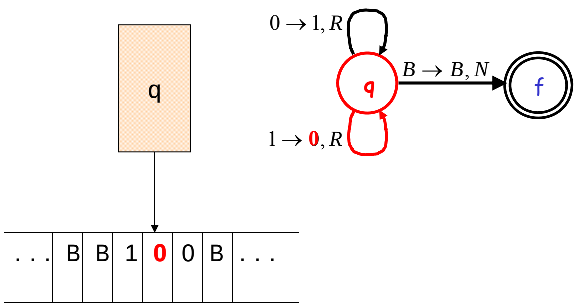
Step: 2



Machine de Turing : exemple

δ(q, 0) = (q, 1, R)
δ(q, 1) = (q, 0, R)
δ(q, B) = (f, B, N)

Step: 2

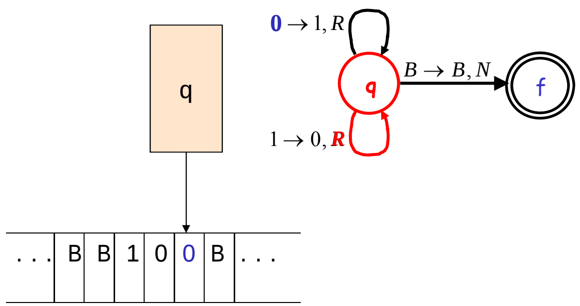


9

Machine de Turing : exemple

δ(q, 0) = (q, 1, R)
δ(q, 1) = (q, 0, R)
δ(q, B) = (f, B, N)

Step: 3

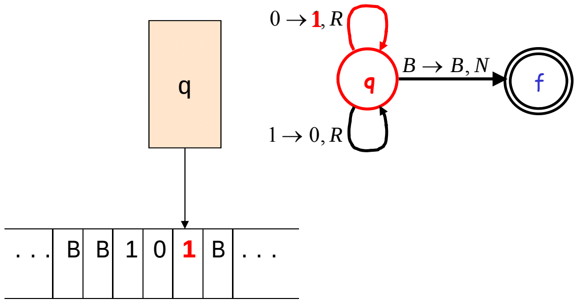


10

Machine de Turing : exemple

$\delta(q, 0) = (q, 1, R)$
$\delta(q, 1) = (q, 0, R)$
$\delta(q, B) = (f, B, N)$

Step: 3

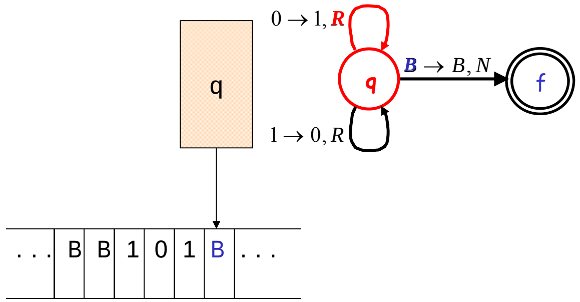


11

Machine de Turing : exemple

$\delta(q, 0) = (q, 1, R)$
$\delta(q, 1) = (q, 0, R)$
$\delta(q, B) = (f, B, N)$

Step: 3

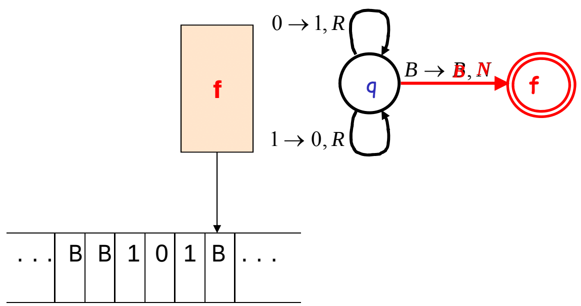


12

Machine de Turing : exemple

$\delta(q, 0) = (q, 1, R)$
$\delta(q, 1) = (q, 0, R)$
$\delta(q, B) = (f, B, N)$

Step: 3

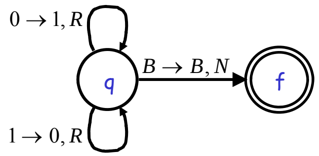


Machine de Turing : exemple

Complémentation

Trace d'exécution:

```
1   BBB010BBBB
       q
2   BBB110BBBB
        q
3   BBB100BBBB
         q
4   BBB101BBBB
          f
```



Machine de Turing : exemple

Representations équivalentes :

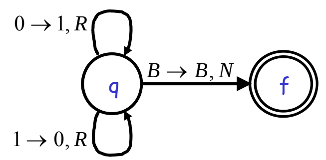
- fonction de transition δ

  δ(q, 0) = (q, 1, R)
  δ(q, 1) = (q, 0, R)
  δ(q, B) = (f, B, N)

- Programme en pseudo-code

  q: *if read=0 then write 1, move R, goto q:*
  if read=1 then write 0, move R, goto q:
  if read=B then - , move L, goto f:
  f: *halt*

- Graphe de transition



- Matrice de transition

Etat	Entrée	Sortie	Opération	Etat suivant	Commentaire
q	0	1	R	q	État initial
q	1	0	R	q	
q	B	B	N	f	
f					État final

Machine de Turing : Exercice 1

Incrémenter : ajouter 1 à un nombre binaire

u => u+1 1011 + 1 = 1100
(11) (12)

Ruban : BBB1011BBBB

Idée:

- Aller au 1er chiffre à gauche (unité)
- Tant que 1, écrire 0 et continuer à gauche (retenue de 1)
- Si Blanc, écrire 1, fin
- Si 0 écrire 1 fin

Ruban : BBB1011BBBB
=> BBB1010BBBB
BBB1000BBBB
BBB1100BBBB

Graphe de transition:

?

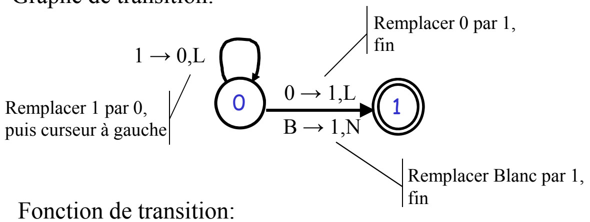
Machine de Turing : Exercice 1

Incrémenter : ajouter 1 à un nombre binaire

Ruban : BBB1011BBBB

- States : Q = {0 (start), 1 (final)}.
- Input symbols: Σ = {0, 1}.
- Tape symbols: Γ = {0, 1, B}.
- Operations: O = {L, R, N}

Graphe de transition:



Fonction de transition:

δ(0, 1) = (0, 0, L)
δ(0, 0) = (1, 1, L)
δ(0, B) = (1, 1, N)

Matrice de transition:

Etat	Entrée	Sortie	Opération	Etat suivant	Commentaire
0	0	1	L	1	État initial
0	1	0	L	0	
0	B	1	N	1	
1					État final

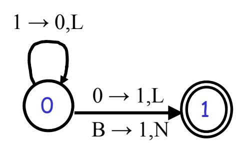
Machine de Turing : Exercice 1

Incrémenter : ajouter 1 à un nombre binaire

Trace :

```
1  BBB1011BBBB
          0
2  BBB1010BBBB
         0
3  BBB1000BBBB
        0
4  BBB1100BBBB
       1
```

Graphe de transition:



Une autre trace:

```
1  BBB1111BBBB
          0
2  BBB1110BBBB
         0
3  BBB1100BBBB
        0
4  BBB1000BBBB
       0
5  BBB0000BBBB
      0
6  BB10000BBBB
      1
```

Machine de Turing : Exercice 2

Décrémenter: soustraire 1 à un nombre binaire

u => u-1

1000 - 1 = 0111
(8) (7)

Ruban : BBB1000BBBB

Idée:

- Aller au 1er chiffre à gauche (unité)
- Tant que 0, écrire 1 et continuer à gauche (retenue de -1)
- Si Blanc, erreur résultat négatif, fin
- Si 1 écrire 0 fin

Ruban : BBB1000BBBB
=> BBB1001BBBB
BBB1011BBBB
BBB1111BBBB
BBB0111BBBB

Graphe de transition:

?

Machine de Turing : Exercice 3

Somme de 2 nombres unaires

11111 + 11111111 = 111111111111
(5) (8) (12 = 5+8 -1)
4 + 7 = 11

Ruban : BBB11111B11111111BBBB

Idée : concaténer les 2 nombres

- Remplacer le symbole blanc séparateur B par 1
- Effacer les 2 chiffres 1 à gauche par B

BBB11111B11111111BBBB
=> BBBBB111111111111BBBB

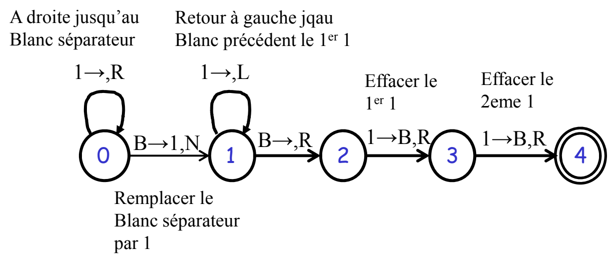
Graphe de transition:

?

Machine de Turing : Exercice 3

Somme de 2 nombres unaires

Graphe de transition:



Fonction de transition:

δ(0, 1) = (0, 1, R)
δ(0, B) = (1, 1, N)
δ(1, 1) = (1, 1, L)
δ(1, B) = (2, B, R)
δ(2, 1) = (3, B, R)
δ(3, 1) = (4, B, R)

Matrice de transition:

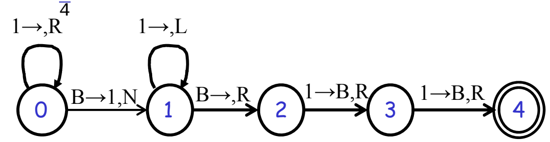
Etat	Entrée	Sortie	Opération	Etat suivant	Commentaire
0	1	1	R	0	État initial
0	B	1	N	1	
1	1	1	L	1	
1	B	B	R	2	
2	1	B	R	3	
3	1	B	R	4	
4					État final

Machine de Turing : Exercice 3

Somme de 2 nombres unaires

Trace :

```
1  BBB11111B11111111BBBB
      0
2  BBB11111B11111111BBBB
       0
3  BBB11111B11111111BBBB
        0
4  BBB11111B11111111BBBB
         0
5  BBB11111B11111111BBBB
       0
6  BBB11111B11111111BBBB
           0
7  BBB11111111111111BBBB
          1
8  BBB11111111111111BBBB
         1
9  BBB11111111111111BBBB
        1
10 BBB11111111111111BBBB
       1
11 BBB11111111111111BBBB
      1
12 BBB11111111111111BBBB
     1
13 BBB11111111111111BBBB
      2
14 BBBB1111111111111BBBB
       3
15 BBBBB111111111111BBBB
        4
```



Machine de Turing : Exercice 4

Somme de 2 nombres binaires

1011 + 110 = 10001
11 + 6 = 17

↓

Ruban : `BBB1011#110BBBB`

X#Y

Idée : X + 1 * Y
Y => Y – 1 (décrémenter Y)
X => X + 1 (incrémenter X)
jqa Y = 0
le résultat X + Y est dans X

```
   BBB1011#110BBBB
=> BBB1011#101BBBB  (Y-1)
   BBB1100#101BBBB  (X+1)
   BBB1100#100BBBB  (Y-1)
   BBB1101#100BBBB  (X+1)
   …
   BB10001#000BBBB (X+Y)
```

Graphe de transition:

?

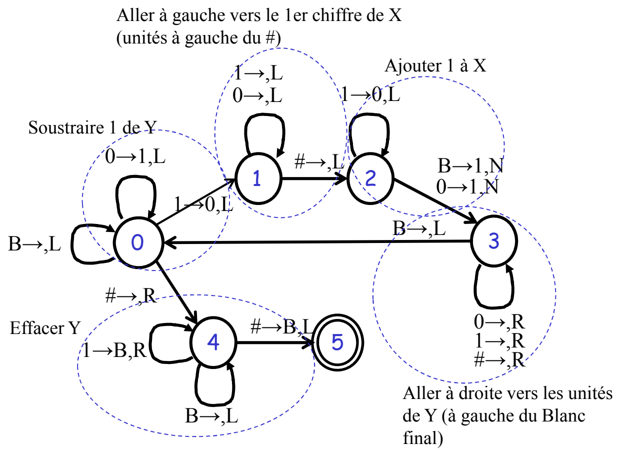
Machine de Turing : Exercice 4

Somme de 2 nombres binaires

Ruban : `BBB1011#110BBBB`

X#Y

Graphe de transition:



Machine de Turing : Exercice 4

Somme de 2 nombres binaires

Trace :

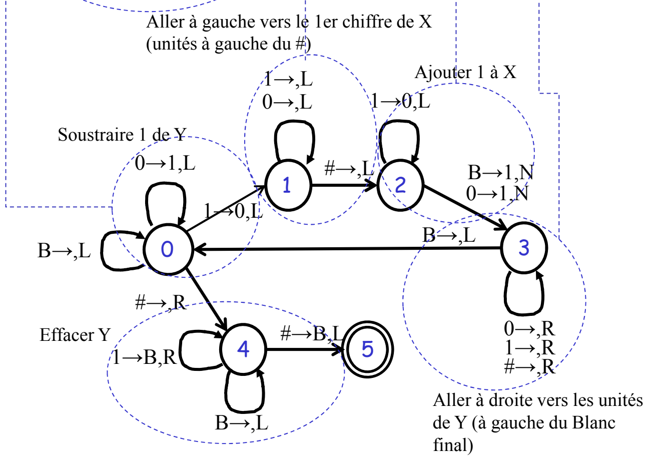
```
1  BBB1011#110BBBB
             0
2  BBB1011#111BBBB
            0
3  BBB1011#101BBBB
           1
4  BBB1011#101BBBB
          1
5  BBB1011#101BBBB
         2
6  BBB1010#101BBBB
        2
7  BBB1000#101BBBB
       2
8  BBB1100#101BBBB
      3
9  BBB1100#101BBBB
       3
10 BBB1100#101BBBB
        3
11 BBB1100#101BBBB
         3
12 BBB1100#101BBBB
          3
13 BBB1100#101BBBB
           3
14 BBB1100#101BBBB
            3
15 BBB1100#101BBBB
             3
```

0

1

2

3



Machine de Turing : Exercice 4

Somme de 2 nombres binaires

Trace :

```
1  BBB1011#110BBBB     16 BBB1100#101BBBB     27 BBB1101#100BBBB
             0                      0                      0
2  BBB1011#111BBBB     17 BBB1100#100BBBB     28 BBB1101#101BBBB
            0                      0                      0          0
3  BBB1011#101BBBB     18 BBB1100#100BBBB     29 BBB1101#111BBBB
           1                      1                      0
-----------------------------------------------------------------------
4  BBB1011#101BBBB     19 BBB1100#100BBBB     30 BBB1101#011BBBB
          1                      1                      1            1
-----------------------------------------------------------------------
5  BBB1011#101BBBB     20 BBB1100#100BBBB     31 BBB1101#011BBBB
         2                      2                      2
6  BBB1010#101BBBB     21 BBB1101#100BBBB     32 BBB1100#011BBBB
        2                      3                      2              2
7  BBB1000#101BBBB                            33 BBB1110#011BBBB
       2                                            3
8  BBB1100#101BBBB
      3
-----------------------------------------------------------------------
9  BBB1100#101BBBB     22 BBB1101#100BBBB     34 BBB1110#011BBBB
       3                      3                      3
10 BBB1100#101BBBB     23 BBB1101#100BBBB     35 BBB1110#011BBBB
        3                      3                      3
11 BBB1100#101BBBB     24 BBB1101#100BBBB     36 BBB1110#011BBBB
         3                      3                      3             3
12 BBB1100#101BBBB     25 BBB1101#100BBBB     37 BBB1110#011BBBB
          3                      3                      3
13 BBB1100#101BBBB     26 BBB1101#100BBBB     38 BBB1110#011BBBB
           3                      3                      3
14 BBB1100#101BBBB                            39 BBB1110#011BBBB
            3                                           3
15 BBB1100#101BBBB
             3
```

Machine de Turing : Exercice 4

Somme de 2 nombres binaires

Trace :

0

```
40 BBB1110#011BBBB
             0
41 BBB1110#010BBBB
            1
51 BBB1111#010BBBB
             0
52 BBB1111#011BBBB
            0
53 BBB1111#001BBBB
           1
69 BB10000#001BBBB
             0
70 BB10000#000BBBB
            1
```

1

```
42 BBB1110#010BBBB
           1
43 BBB1110#010BBBB
          1
54 BBB1111#001BBBB
          1
71 BB10000#000BBBB
           1
72 BB10000#000BBBB
          1
```

2

```
44 BBB1110#010BBBB
         2
45 BBB1111#010BBBB
         3
55 BBB1111#001BBBB
         2
56 BBB1110#001BBBB
        2
57 BBB1100#001BBBB
       2
58 BBB1000#001BBBB
      2
59 BBB0000#001BBBB
     2
60 BB10000#001BBBB
     3
73 BB10000#000BBBB
         2
74 BB10001#000BBBB
         3
```

3

```
46 BBB1111#010BBBB
          3
47 BBB1111#010BBBB
           3
48 BBB1111#010BBBB
            3
49 BBB1111#010BBBB
             3
50 BBB1111#010BBBB
              3
61 BB10000#001BBBB
      3
62 BB10000#001BBBB
       3
63 BB10000#001BBBB
        3
64 BB10000#001BBBB
         3
65 BB10000#001BBBB
          3
66 BB10000#001BBBB
           3
67 BB10000#001BBBB
            3
68 BB10000#001BBBB
             3
69 BB10000#001BBBB
              3
75 BB10001#000BBBB
          3
76 BB10001#000BBBB
           3
77 BB10001#000BBBB
            3
78 BB10001#000BBBB
             3
79 BB10001#000BBBB
              3
```


Machine de Turing : Exercice 4

Somme de 2 nombres binaires

Trace :

0

```
80 BB10001#000BBBB
             0
81 BB10001#001BBBB
            0
82 BB10001#011BBBB
           0
83 BB10001#111BBBB
          0
```

4

```
84 BB10001#111BBBB
           4
85 BB10001#B00BBBB
            4
86 BB10001#BB0BBBB
             4
87 BB10001#BBBBBBB
              4
88 BB10001#BBBBBBB
             4
89 BB10001#BBBBBBB
            4
90 BB10001#BBBBBBB
           4
91 BB10001#BBBBBBB
          4
91 BB10001BBBBBBBB
         5
```

Machine de Turing : Exercice 5

Différence de 2 nombres binaires

X – Y (avec X>Y>=0)
1011 - 110 = 101
11 - 6 = 5

Ruban : `BBB1011#110BBBB`
X#Y

Idée : X - 1 * Y
Y => Y – 1 (décrémenter Y)
X => X - 1 (décrémenter X)
jqa Y = 0
le résultat X - Y est dans X

```
     BBB1011#110BBBB
=>   BBB1011#101BBBB  (Y-1)
     BBB1010#101BBBB  (X-1)
     BBB1010#100BBBB  (Y-1)
     BBB1001#100BBBB  (X-1)
     …
     BBBB101#000BBBB (X-Y)
```

Graphe de transition:

?

Machine de Turing : Exercice 6

Conversion unaire vers binaires

X en unaire (avec X >=0)
$(11111)_1$ => $(100)_2$ = 4

Ruban : `BBB11111BBBB`

Idée : Y=0, Y = Y + 1 * X
X => X – 1 (décrémenter X en unaire)
Y => Y + 1 (incrémenter Y en binaire)
jqa X = 0
le résultat est dans X

```
     BBBBB11111BBB
=>   BBB0B1111BBBB     (Y=0, X-1)
     BBB0B111BBBBB     (X-1)
     BBB1B111BBBBB     (Y+1)
     BBB1B11BBBBBB     (X-1)
     …
     BB11B1BBBBBBB     (Y+1)
     BB11BBBBBBBBB     (X-1)
     B100BBBBBBBBB     (Y+1)
```

Graphe de transition:

?

Machine de Turing : Exercice 7

Conversion binaire vers unaires

X en binaire (avec X >=0)
$(100)_1$ => $(11111)_2$ = 4

Ruban : `BBB100BBBB`

Idée : Y=0, Y = Y + 1 * X
X => X – 1 (décrémenter X en unaire)
Y => Y + 1 (incrémenter Y en binaire)
jqa X = 0
le résultat est dans X

```
      BB100BBBBBBBB
=>    BB100B1BBBBBB     (Y=0)
      BB011B1BBBBBB     (X-1)
      BB011B11BBBBB     (Y+1)
      BB010B111BBBB     (X-1)
        …
      BB000B1111BBB     (X-1)
      BB000B11111BB     (Y+1)
```

Graphe de transition:

?

Machine de Turing : Exercice 8

Multiplication de 2 nombres binaires

U V en binaire (avec U et V >=0)
Résultat U*V:

```
1001 * 1010 = 10010 + 1001000 = 1011010
  9  *   10 =    18 +      72        90
```

Ruban : `BBB1001B1010BBBB`
$V_3V_2V_1V_0$

Idée : $W_0 = U * V_0$ (U si V_0=1, 0 sinon)
$W_{i+1} = W_i + U * V_i * 2^{Vi}$ (U << i si V_i=1, 0 sinon)

```
      BB1001B1010BBBBBBBBBBBBBBBB
=>    BB1001B10YXB10010BBBBBBBBBB
      BB1001BYXXXB10010B1001000BBB
        (X-1)
      BB011B11BBBBB     (Y+1)
      BB010B111BBBB     (X-1)
        …
      BB000B1111BBB     (X-1)
      BB000B11111BB     (Y+1)
```

Graphe de transition:

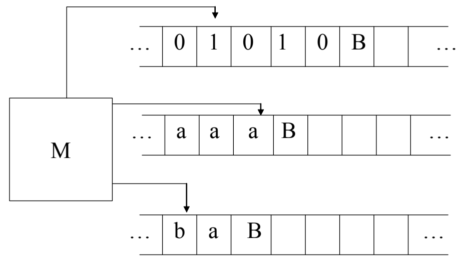
?

Généralisation

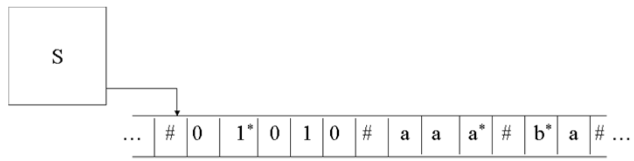
- Plusieurs pistes sur le ruban
- Plusieurs rubans
- Rubans infinis dans les deux directions
- Plusieurs têtes de lecture/écriture
- Multidimentionel

- Tous équivalents au modèle original

Multitape Machine



Equivalent Single Tape Machine:



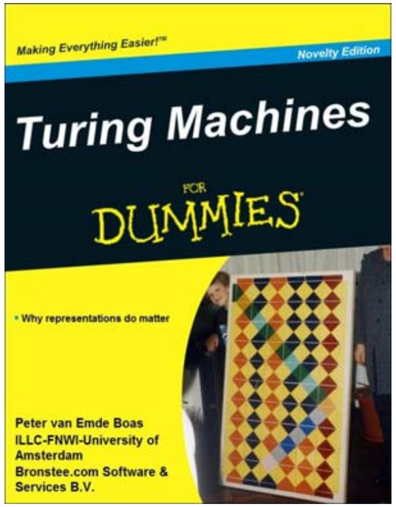
Exemples

- Échanger le contenu de deux variables
- Incrémenter ou décrémenter un compteur
- Convertir un nombre unaires/binaires
- Additionner deux entiers en binaires
- Multiplier deux entiers en binaire
- Comparer deux entiers en binaire
- Trier un tableau

35

Appendix:

Extracted from



36

The teachings of our Master

Our textbooks present Turing Machine programs in the format of quintuples or quadruples.

What format did Turing use himself ?

Some fragments of the 1936 paper

Configuration		*Behaviour*	
m-config.	*symbol*	*operations*	*final m-config.*
b	None	$P0, R$	c
c	None	R	e
e	None	$P1, R$	f
f	None	R	b

Looks like quintuples….

Configuration means state in our terminology

m-config.	*symbol*	*operations*	*final m-config.*
b	None	$P0$	b
	0	$R, R, P1$	b
	1	$R, R, P0$	b

Configuration		*Behaviour*	
m-config.	*symbol*	*operations*	*final m-config.*
b		$Pə, R, Pə, R, P0, R, R, P0, L, L$	o
o	1	R, Px, L, L, L	o
	0		q
q	Any (0 or 1)	R, R	q
	None	$P1, L$	p
p	x	E, R	q
	ə	R	f
	None	L, L	p
f	Any	R, R	f
	None	$P0, L, L$	o

For Turing Composite transitions are allowed

with the m-configuration written below the scanned symbol. The successive complete configurations are separated by colons.

```
  : ə ə 0   0 : ə ə 0   0 : ə ə 0   0 : ə ə 0   0       : ə ə 0   0   1 :
𝔟     𝔬         𝔮           𝔮           𝔮                 𝔭
ə ə 0   0   1 : ə ə 0   0   1 : ə ə 0   0   1 : ə ə 0   0   1 :
    𝔭           𝔭           𝔣               𝔣
ə ə 0   0   1 : ə ə 0   0   1     : ə ə 0   0   1   0 :
            𝔣               𝔣               𝔬
ə ə 0   0   1 x 0 : ....
        𝔢
```

This table could also be written in the form

$$\mathfrak{b} : ə\,ə\,\mathfrak{o}\,0 \quad 0 : ə\,ə\,\mathfrak{q}\,0 \quad 0 : \ldots, \qquad (C)$$

in which a space has been made on the left of the scanned symbol and the m-configuration written in this space. This form is less easy to follow, but we shall make use of it later for theoretical purposes.

This is an example of the Intrinsic Representation

The skeleton tables are to be regarded as nothing but abbreviations: they are not essential. So long as the reader understands how to obtain the complete tables from the skeleton tables, there is no need to give any exact definitions in this connection.

Let us consider an example:

m-config.	*Symbol*	*Behaviour*	*Final m-config.*
$\mathfrak{f}(\mathfrak{C}, \mathfrak{B}, \alpha)$	ə	L	$\mathfrak{f}_1(\mathfrak{C}, \mathfrak{B}, \alpha)$
	not ə	L	$\mathfrak{f}(\mathfrak{C}, \mathfrak{B}, \alpha)$
$\mathfrak{f}_1(\mathfrak{C}, \mathfrak{B}, \alpha)$	α		$\mathfrak{C}$
	not α	R	$\mathfrak{f}_1(\mathfrak{C}, \mathfrak{B}, \alpha)$
	None	R	$\mathfrak{f}_2(\mathfrak{C}, \mathfrak{B}, \alpha)$
$\mathfrak{f}_2(\mathfrak{C}, \mathfrak{B}, \alpha)$	α		$\mathfrak{C}$
	not α	R	$\mathfrak{f}_1(\mathfrak{C}, \mathfrak{B}, \alpha)$
	None	R	$\mathfrak{B}$

From the m-configuration $\mathfrak{f}(\mathfrak{C}, \mathfrak{B}, \alpha)$ the machine finds the symbol of form α which is farthest to the left (the "first α") and the m-configuration then becomes $\mathfrak{C}$. If there is no α then the m-configuration becomes $\mathfrak{B}$.

A Macro language for Turing Machine programs

(In the explanations the symbol "→" is used to signify "the machine goes into the m-configuration. . . .")

$\mathfrak{e}(\mathfrak{C}, \mathfrak{B}, \alpha)$		$\mathfrak{f}\big(\mathfrak{e}_1(\mathfrak{C}, \mathfrak{B}, \alpha), \mathfrak{B}, \alpha\big)$	From $\mathfrak{e}(\mathfrak{C}, \mathfrak{B}, \alpha)$ the first α is erased and $\rightarrow \mathfrak{C}$. If there is no $\alpha \rightarrow \mathfrak{B}$.
$\mathfrak{e}_1(\mathfrak{C}, \mathfrak{B}, \alpha)$	E	$\mathfrak{C}$	
$\mathfrak{e}(\mathfrak{B}, \alpha)$		$\mathfrak{e}\big(\mathfrak{e}(\mathfrak{B}, \alpha), \mathfrak{B}, \alpha\big)$	From $\mathfrak{e}(\mathfrak{B}, \alpha)$ all letters α are erased and $\rightarrow \mathfrak{B}$.

The last example seems somewhat more difficult to interpret than most. Let us suppose that in the list of m-configurations of some machine there appears $\mathfrak{e}(\mathfrak{b}, x)$ $(= \mathfrak{q}$, say). The table is

	$\mathfrak{e}(\mathfrak{b}, x)$	$\mathfrak{e}\big(\mathfrak{e}(\mathfrak{b}, x), \mathfrak{b}, x\big)$
or	$\mathfrak{q}$	$\mathfrak{e}(\mathfrak{q}, \mathfrak{b}, x)$.

Or, in greater detail:

$\mathfrak{q}$		$\mathfrak{e}(\mathfrak{q}, \mathfrak{b}, x)$
$\mathfrak{e}(\mathfrak{q}, \mathfrak{b}, x)$		$\mathfrak{f}\big(\mathfrak{e}_1(\mathfrak{q}, \mathfrak{b}, x), \mathfrak{b}, x\big)$
$\mathfrak{e}_1(\mathfrak{q}, \mathfrak{b}, x)$	E	$\mathfrak{q}$.

In this we could replace $\mathfrak{e}_1(\mathfrak{q}, \mathfrak{b}, x)$ by $\mathfrak{q}'$ and then give the table for $\mathfrak{f}$ (with the right substitutions) and eventually reach a table in which no m-functions appeared.

$\mathfrak{pe}(\mathfrak{C}, \beta)$			$\mathfrak{f}\big(\mathfrak{pe}_1(\mathfrak{C}, \beta), \mathfrak{C}, \mathfrak{ə}\big)$	From $\mathfrak{pe}(\mathfrak{C}, \beta)$ the machine prints β at the end of the sequence of symbols and $\rightarrow \mathfrak{C}$.
$\mathfrak{pe}_1(\mathfrak{C}, \beta)$	Any	R, R	$\mathfrak{pe}_1(\mathfrak{C}, \beta)$	
	None	$P\beta$	$\mathfrak{C}$	
$\mathfrak{l}(\mathfrak{C})$		L	$\mathfrak{C}$	From $\mathfrak{f}'(\mathfrak{C}, \mathfrak{B}, \alpha)$ it does the same as for $\mathfrak{f}(\mathfrak{C}, \mathfrak{B}, \alpha)$ but moves to the left before $\rightarrow \mathfrak{C}$.
$\mathfrak{r}(\mathfrak{C})$		R	$\mathfrak{C}$	
$\mathfrak{f}'(\mathfrak{C}, \mathfrak{B}, \alpha)$			$\mathfrak{f}\big(\mathfrak{l}(\mathfrak{C}), \mathfrak{B}, \alpha\big)$	
$\mathfrak{f}''(\mathfrak{C}, \mathfrak{B}, \alpha)$			$\mathfrak{f}\big(\mathfrak{r}(\mathfrak{C}), \mathfrak{B}, \alpha\big)$	
$\mathfrak{c}(\mathfrak{C}, \mathfrak{B}, \alpha)$			$\mathfrak{f}'\big(\mathfrak{c}_1(\mathfrak{C}), \mathfrak{B}, \alpha\big)$	$\mathfrak{c}(\mathfrak{C}, \mathfrak{B}, \alpha)$. The machine writes at the end the first symbol marked α and $\rightarrow \mathfrak{C}$.
$\mathfrak{c}_1(\mathfrak{C})$	β		$\mathfrak{pe}(\mathfrak{C}, \beta)$	

This Macro Language supports Recursion !

It will be useful to put these tables into a kind of standard form. In the first place let us suppose that the table is given in the same form as the first table, for example, I on p. 233. That is to say, that the entry in the operations column is always of one of the forms $E : E, R : E, L : Pa : Pa, R : Pa, L : R : L :$ or no entry at all. The table can always be put into this form by introducing more m-configurations. Now let us give numbers to the m-configurations, calling them $q_1, \ldots, q_R$, as in §1. The initial m-configuration is always to be called q_1. We also give numbers to the symbols $S_1, \ldots, S_m$ and, in particular, blank $= S_0$, $0 = S_1$, $1 = S_2$. The lines of the table are now of form

m-config.	*Symbol*	*Operations*	*Final m-config.*	
q_i	S_j	PS_k, L	q_m	(N_1)
q_i	S_j	PS_k, R	q_m	(N_2)
q_i	S_j	PS_k	q_m	(N_3)

Lines such as

q_i	S_j	E, R	q_m

are to be written as

q_i	S_j	PS_0, R	q_m

and lines such as

q_i	S_j	R	q_m

to be written as

q_i	S_j	PS_j, R	q_m

In this way we reduce each line of the table to a line of one of the forms (N_1), (N_2), (N_3).

From each line of form (N_1) let us form an expression $q_i S_j S_k L q_m$; from each line of form (N_2) we form an expression $q_i S_j S_k R q_m$; and from each line of form (N_3) we form an expression $q_i S_j S_k N q_m$.

The format of TM programs which today is conventional arises as a simplification introduced for the purpose of constructing the Universal Turing Machine

Turing operates as an Engineer (Programmer) rather than a Mathematician / Logician