Calculabilité

Lionel Vaux

Université d'Aix-Marseille

DIU EIL, semaine 5, promo 2019

Un ordinateur c'est...

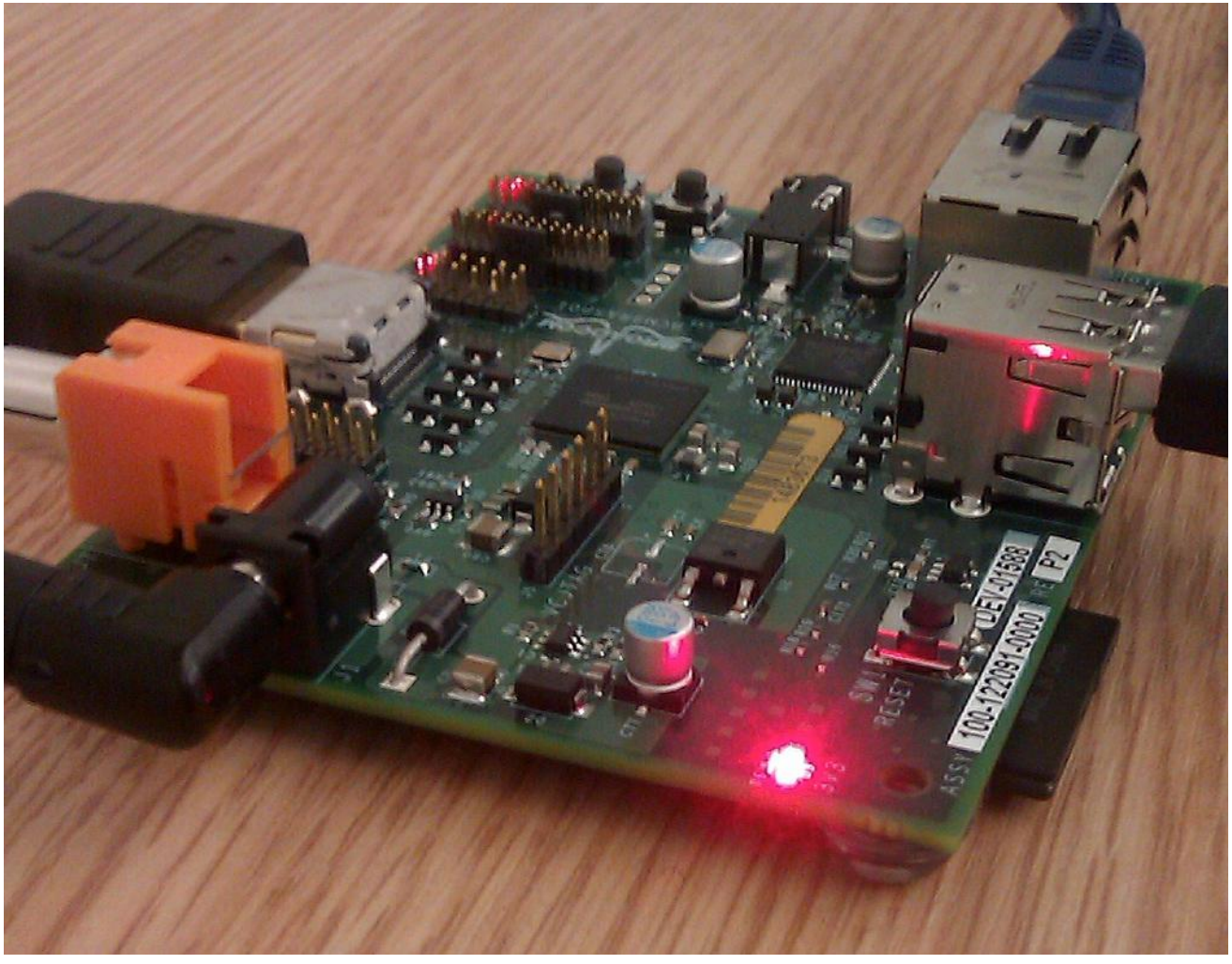
© Kristian Thy, CC-BY-2.0, tiré de Wikimedia Commons

Un ordinateur c'est...

ce qui affiche cette présentation

Un ordinateur c'est aussi...

tout petit (Raspberry Pi) :



© Paul Downey, CC-BY-2.0, tiré de Wikimedia Commons

Un ordinateur c'est aussi...

très gros (IBM Blue Gene/P) :

© Argonne National Laboratory, CC-BY-SA-2.0, tiré de Wikimedia Commons

Un ordinateur c'est aussi...

mon téléphone

et le vôtre aussi

Un ordinateur c'est aussi...

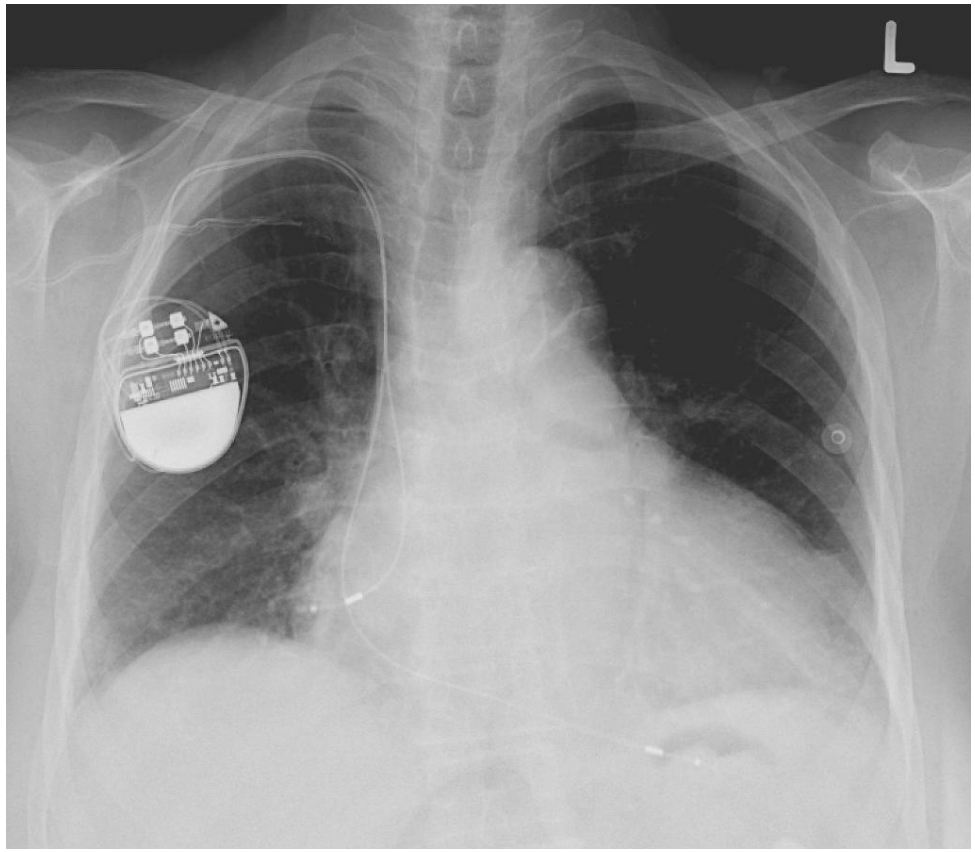
© ChtiTux, CC-BY-SA-2.0, tiré de Wikimedia Commons

Un ordinateur c'est aussi...

© Jon 'ShakataGaNai' Davis, CC-BY-SA-3.0, tiré de Wikimedia Commons

Mais encore. . .

Ceci n'est pas un frigo :

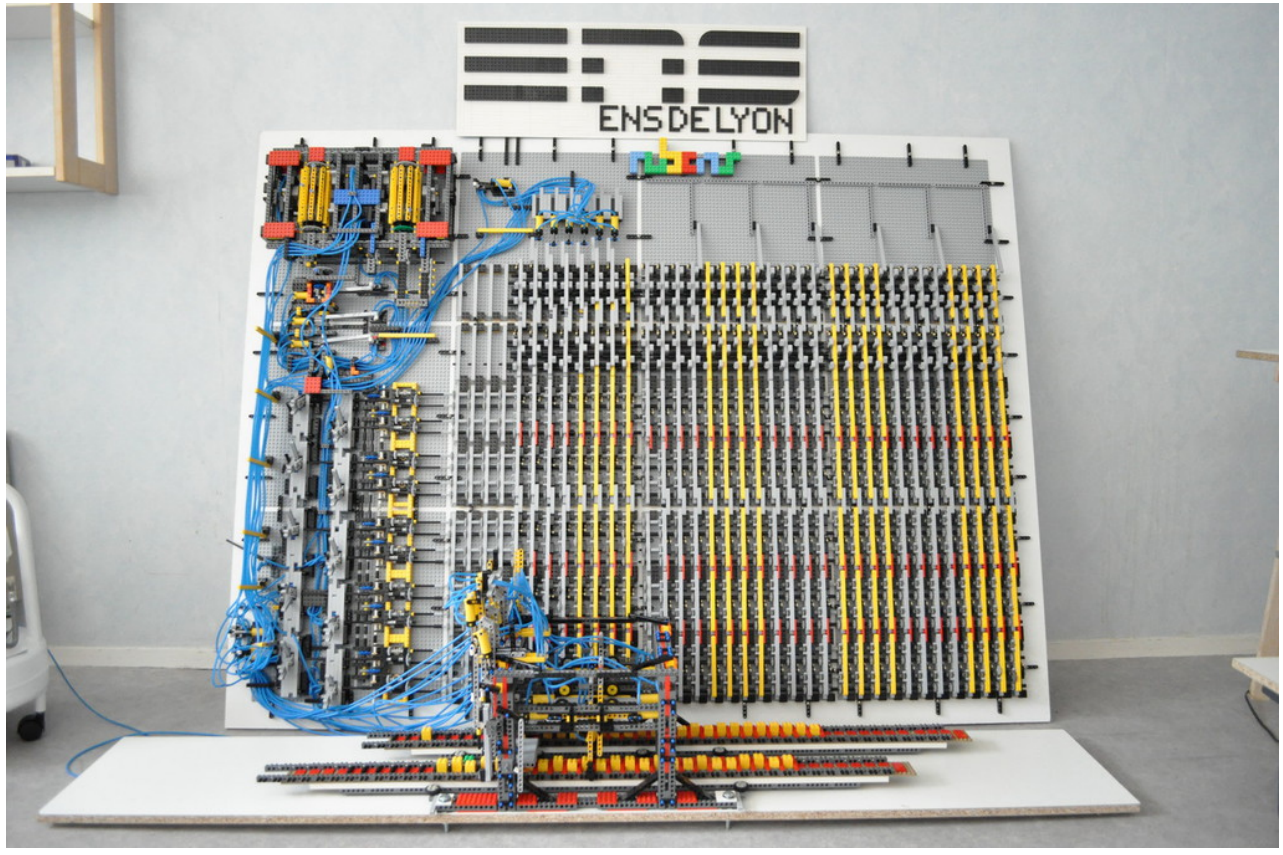


© Lucien Monfils, CC-BY-SA-3.0, tiré de Wikimedia Commons

voir l'histoire de Karen Sandler
`http://www.framablog.org/index.php/post/2012/11/26/un-coeur-gros-comme-ca`
et les travaux de Barnaby Jack sur le piratage de dispositifs médicaux
`https://en.wikipedia.org/wiki/Barnaby_Jack#Pacemakers`

Ou encore. . .

Ceci n'est pas un ordinateur nucléaire :



© Projet Rubens, ÉNS de Lyon, CC-BY

voir le site du projet `http://rubens.ens-lyon.fr/`

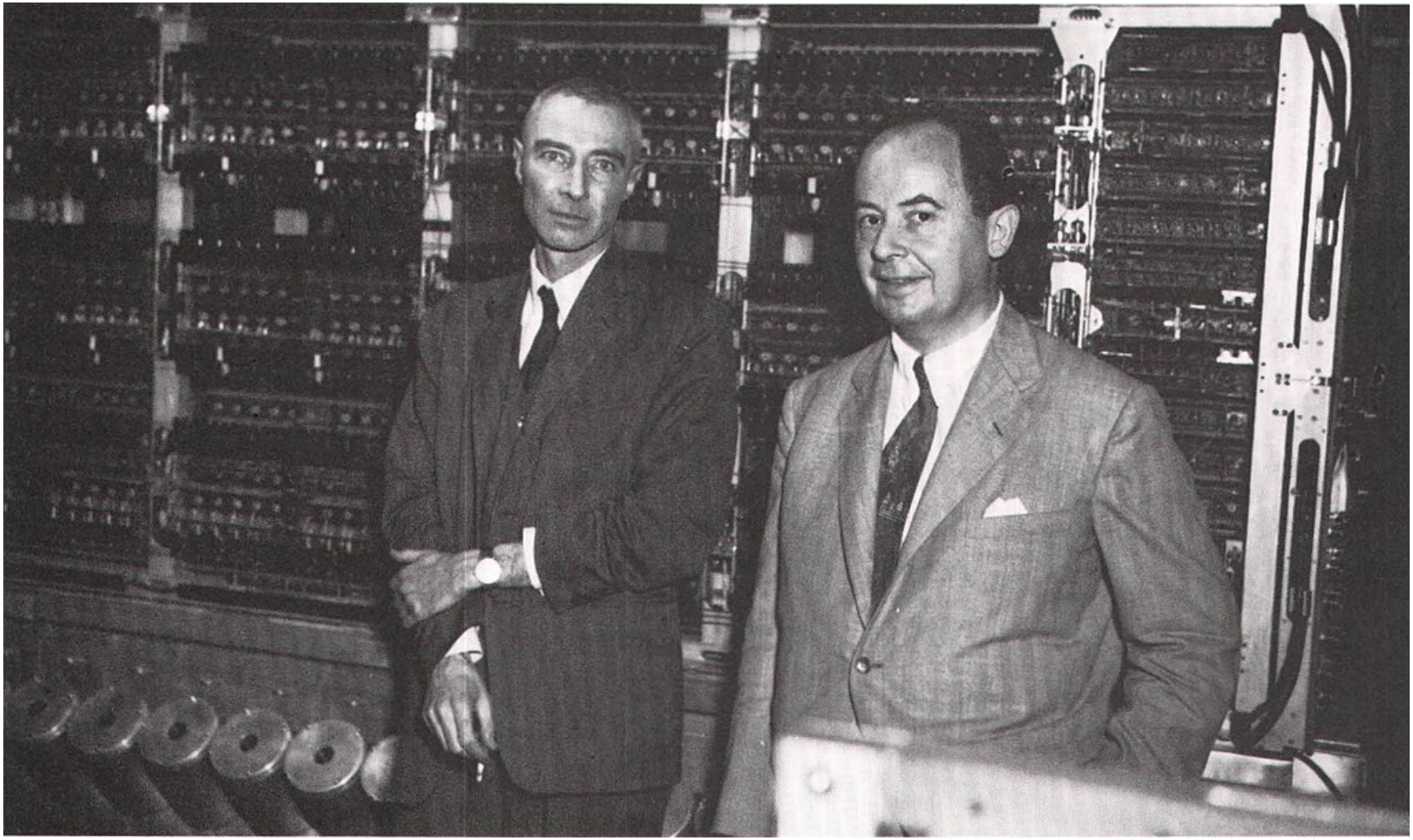
Un ordinateur c'était...

ZX Spectrum

domaine public, tiré de Wikimedia Commons

Le premier ordinateur (?)

EDVAC (1949–1961) : électronique, binaire et programmable.

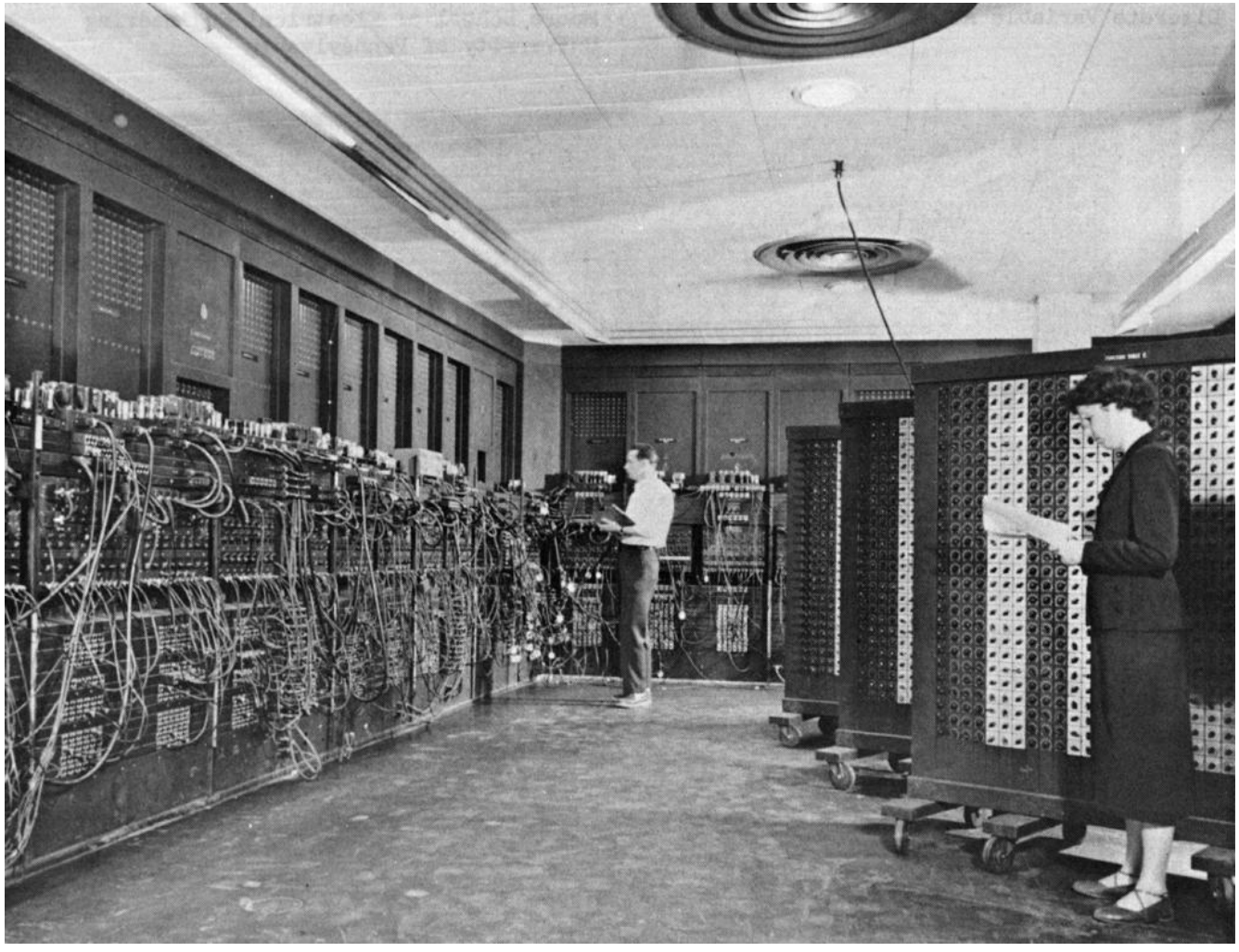


Sur la photo : Robert Oppenheimer et John von Neumann

domaine public, tiré de Wikimedia Commons

Le premier ordinateur (?)

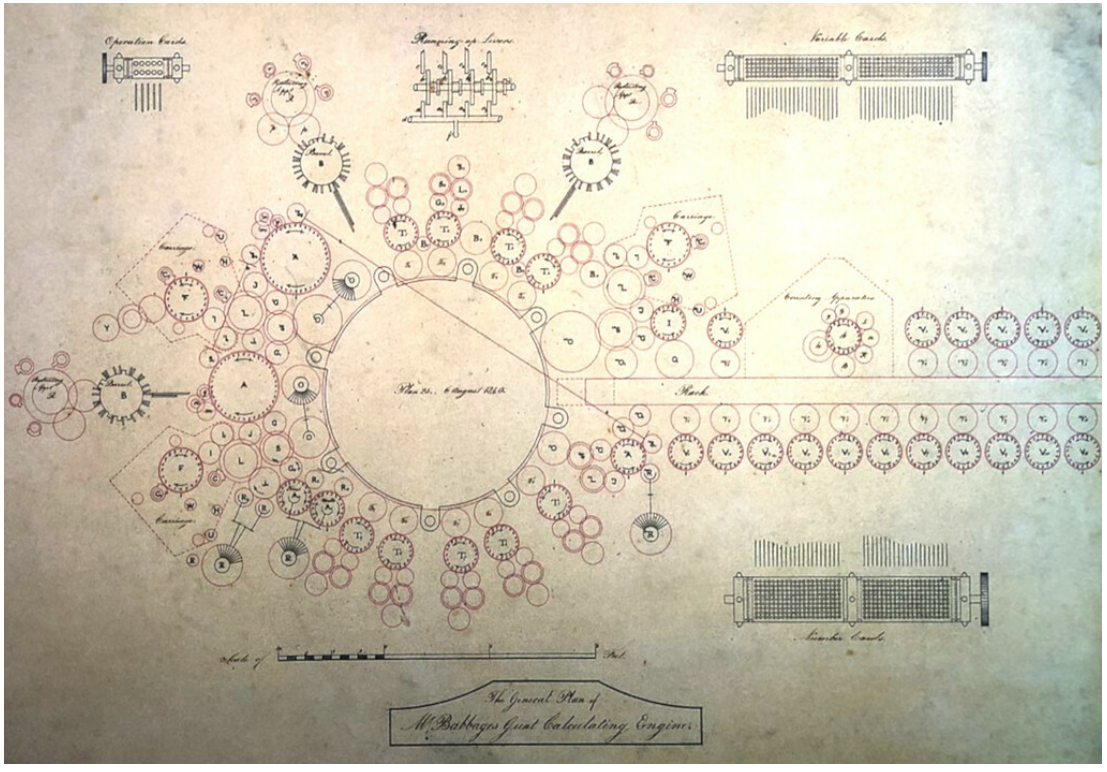
ENIAC (1946–1955) : électronique, décimal et « reconfigurable ».

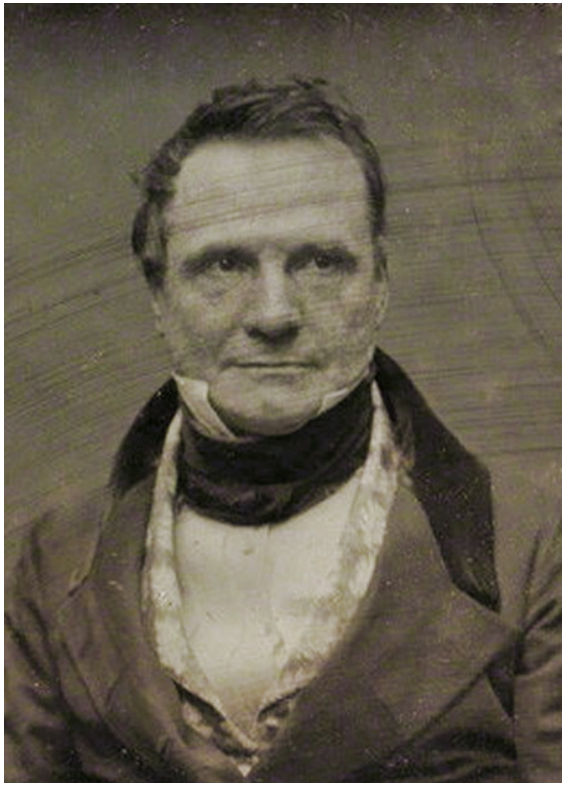


domaine public, tiré de Wikimedia Commons

Le premier ordinateur (?)

La machine analytique de Charles Babbage (1791–1871)





Tentative de définition

Un ordinateur, c'est une machine qu'on peut programmer.

Qu'est-ce que « programmer » ?

Qu'est-ce que « programmer » ?

Vous connaissez

Et qu'est-ce qu'une « machine » ?

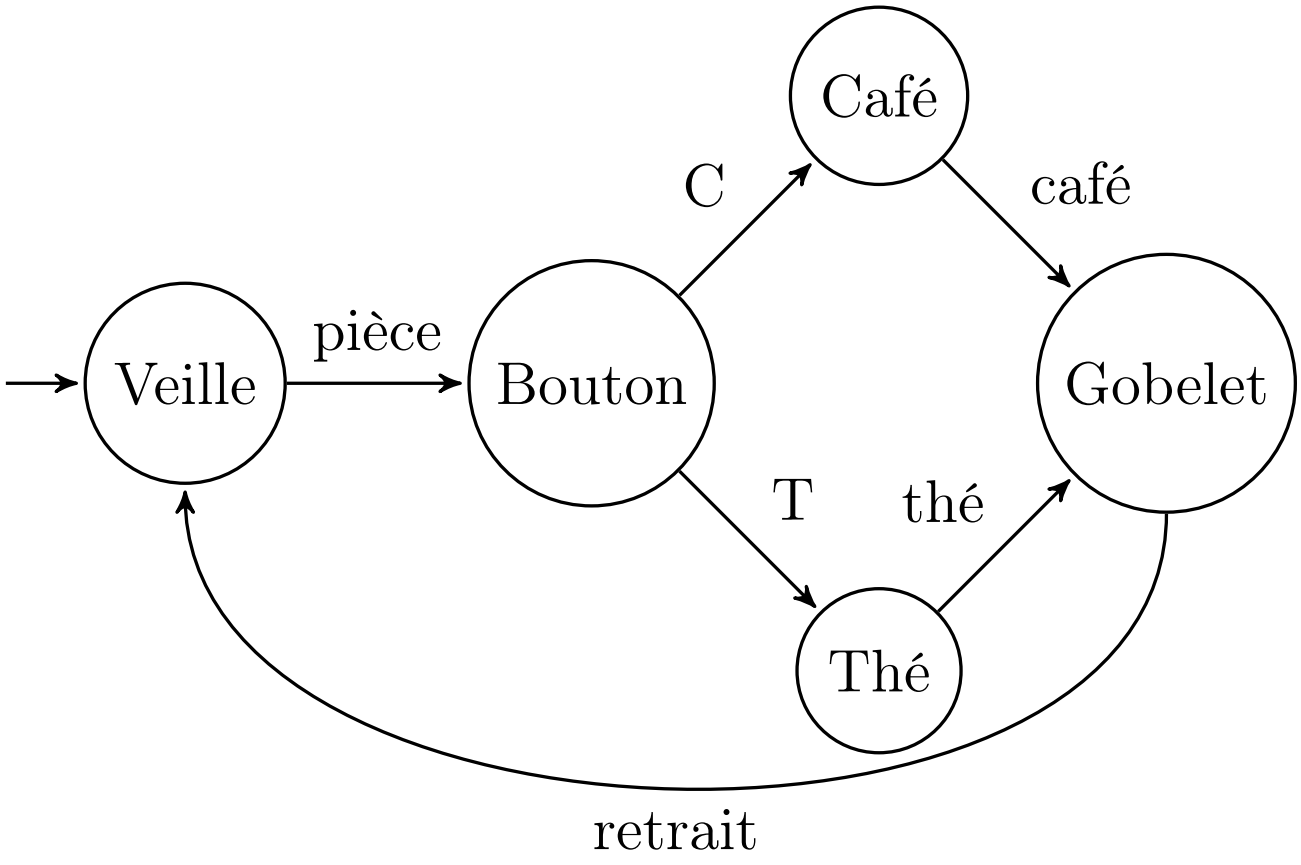
Et qu'est-ce qu'une « machine » ?

Une machine fonctionne de manière automatique.

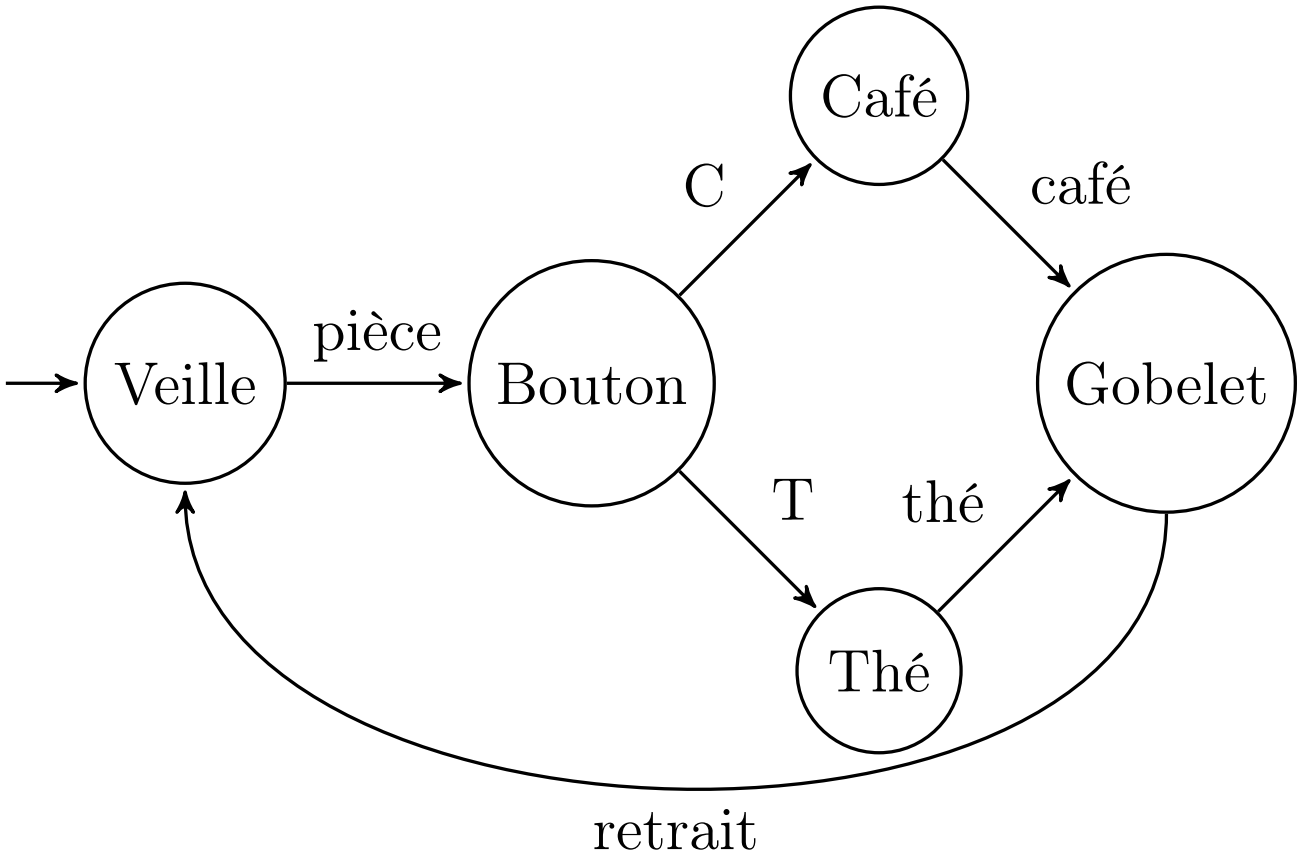
Automates

domaine public, tiré de Wikimedia Commons

Automates



Automates



- un ensemble fini d'états
- des transitions guidées par la donnée à traiter

C'est la notion d'automate fini.

Cf. le cours sur la recherche textuelle.

Que calcule un automate fini ?

- l'information à traiter est la suite des données entrées par l'utilisateur : c'est un mot.

  En théorie des langages formels, on dit « mot »pour n'importe quelle suite de symboles (en programmation, on dit « chaîne de caractères »).

- qu'obtient-on en sortie ?
  - la suite des états traversés par l'automate ?

Que calcule un automate fini ?

- l'information à traiter est la suite des données entrées par l'utilisateur : c'est un mot.

  En théorie des langages formels, on dit « mot »pour n'importe quelle suite de symboles (en programmation, on dit « chaîne de caractères »).

- qu'obtient-on en sortie ?
  - la suite des états traversés par l'automate ?
    ça, c'est plutôt le "comment on calcule"
  - l'état final en fonction de ce qu'on a entré

Que calcule un automate fini ?

- l'information à traiter est la suite des données entrées par l'utilisateur : c'est un mot.

  En théorie des langages formels, on dit « mot »pour n'importe quelle suite de symboles (en programmation, on dit « chaîne de caractères »).

- qu'obtient-on en sortie ?
  - la suite des états traversés par l'automate ?
    ça, c'est plutôt le "comment on calcule"
  - l'état final en fonction de ce qu'on a entré

⇝ on utilise les automates finis pour reconnaître des langages : le langage d'un automate = l'ensemble des mots qui mènent de l'état initial à un état acceptant

Que calcule un automate fini ?

- l'information à traiter est la suite des données entrées par l'utilisateur : c'est un mot.

  En théorie des langages formels, on dit « mot »pour n'importe quelle suite de symboles (en programmation, on dit « chaîne de caractères »).

- qu'obtient-on en sortie ?
  - la suite des états traversés par l'automate ?
    ça, c'est plutôt le "comment on calcule"
  - l'état final en fonction de ce qu'on a entré

$\rightsquigarrow$ on utilise les automates finis pour reconnaître des langages : le langage d'un automate = l'ensemble des mots qui mènent de l'état initial à un état acceptant

Formellement, un automate fini sur les symboles $a, b, c, \ldots \in \Sigma$, c'est :

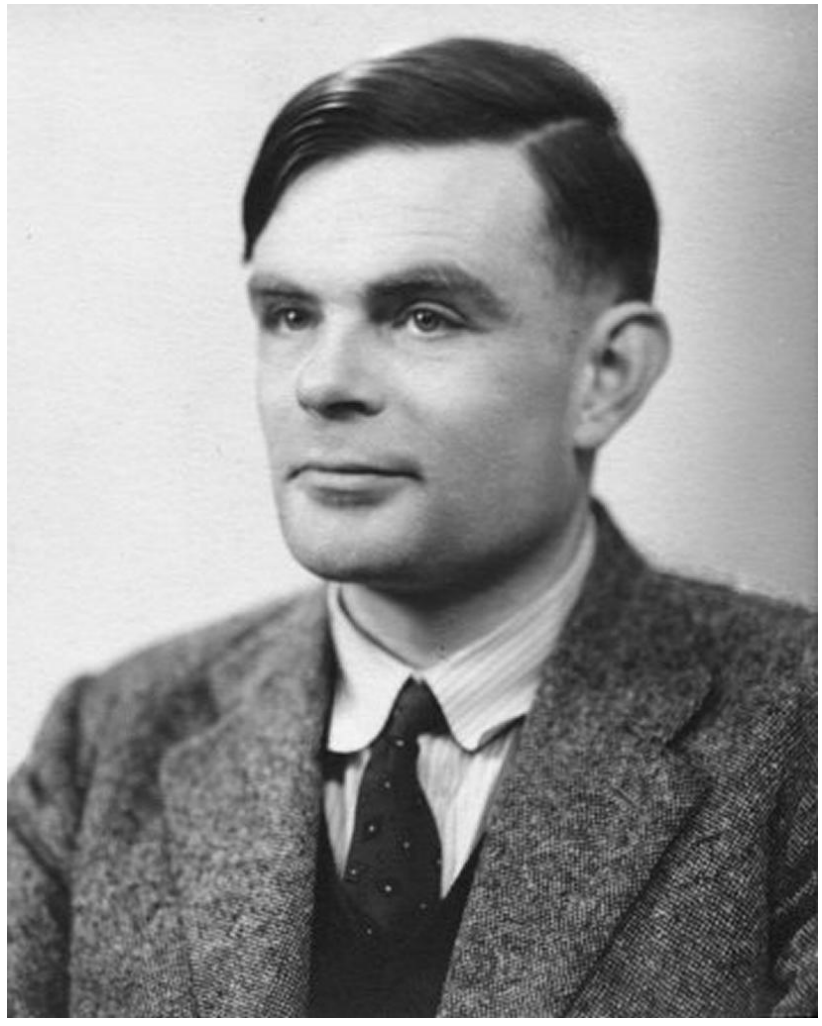
- un ensemble fini d'états E,
- une fonction de transition $\delta : E \times \Sigma \to E$,
- un état initial $I \in E$,
- un ensemble d'états acceptants $F \subset E$.

Exercices : section 1.

Un ordinateur n'est pas une machine à café !

- entrées
- sorties
- opérations
- mémoire
- ...

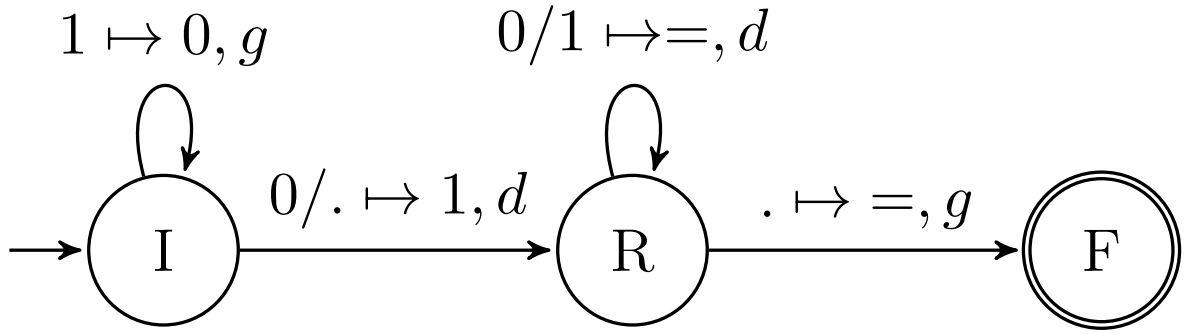
Machines de Turing



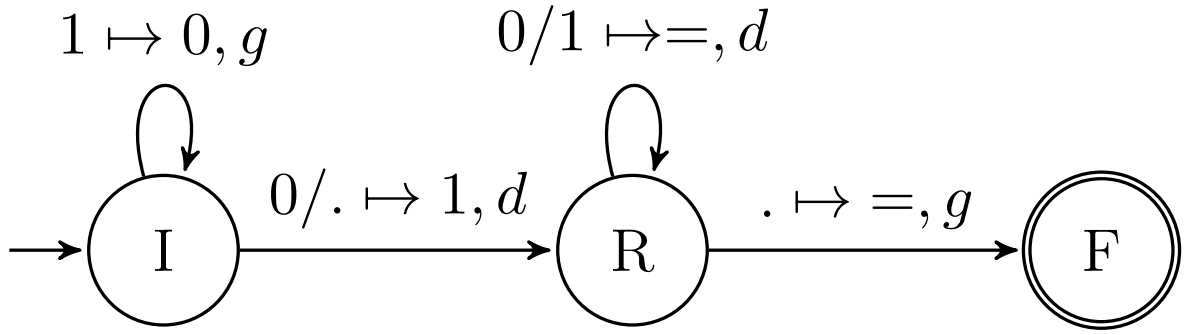
Alan Turing (1912–1954)

domaine public, tiré de Wikimedia Commons

Machines de Turing



Machines de Turing

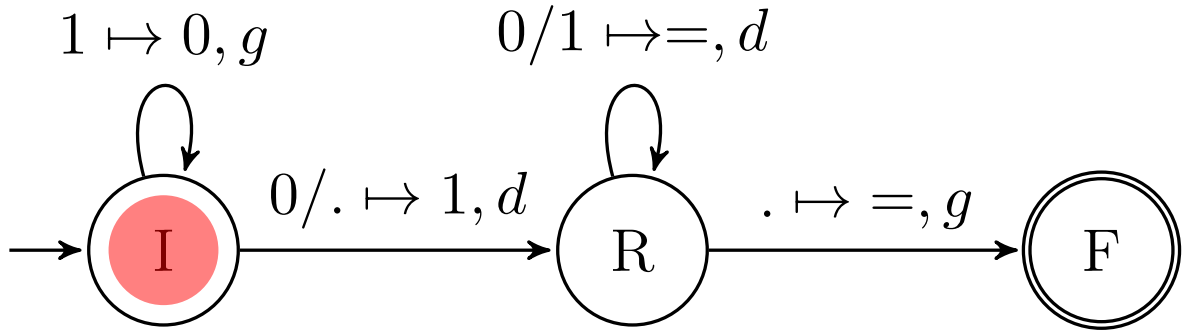


entrée :

$\cdots$	.	1	0	1	1	.	$\cdots$

Mémoire en mode lecture/écriture : on agit sur un ruban sur lequel on lit et on écrit des caractères.

Machines de Turing

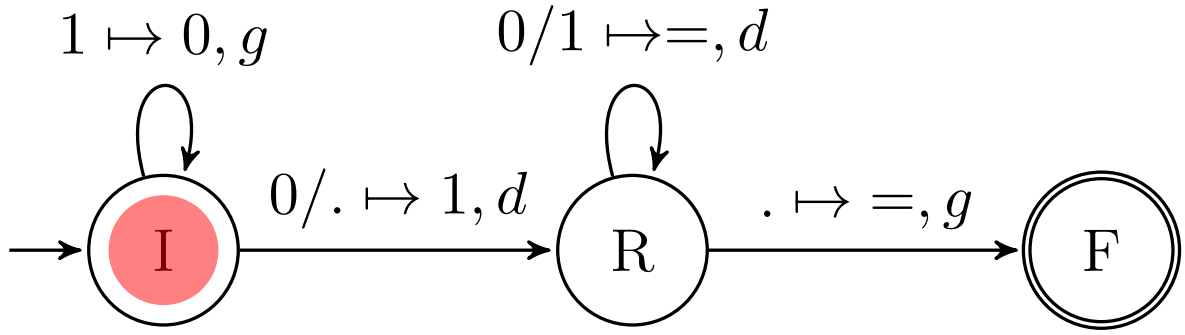


entrée :

$\cdots$	.	1	0	1	1	.	$\cdots$

Mémoire en mode lecture/écriture : on agit sur un ruban sur lequel on lit et on écrit des caractères.

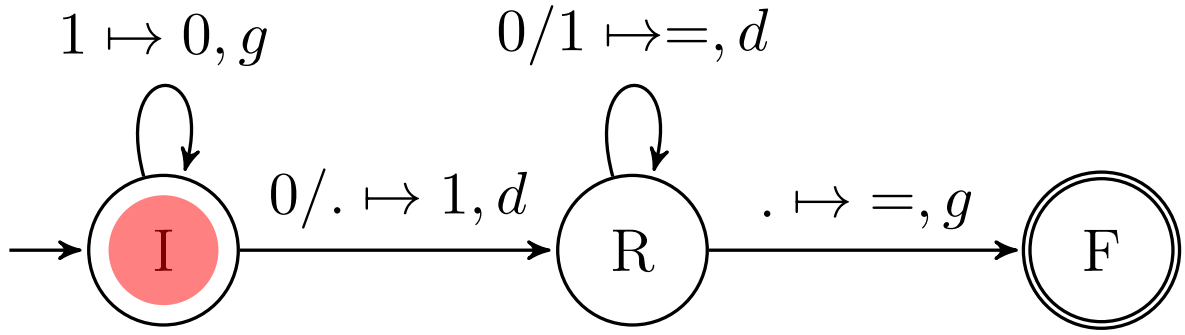
Machines de Turing



entrée :	$\cdots$	.	1	0	1	$\underline{1}$	.	$\cdots$
étape 1 :	$\cdots$	.	1	0	$\underline{1}$	0	.	$\cdots$

Mémoire en mode lecture/écriture : on agit sur un ruban sur lequel on lit et on écrit des caractères.

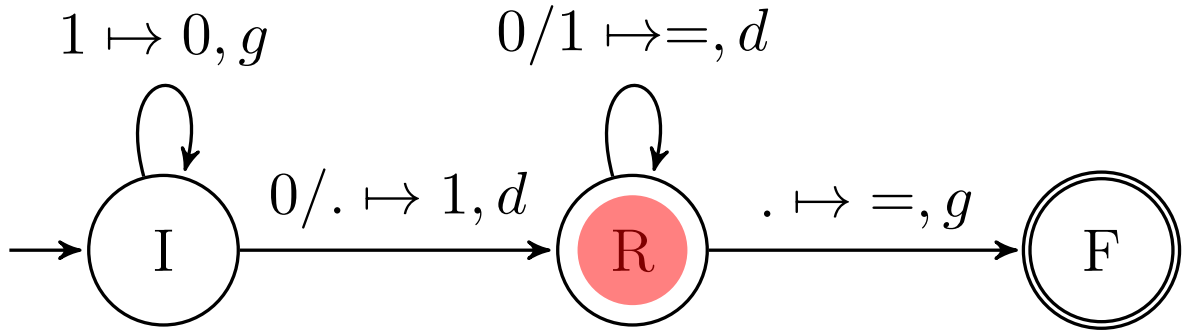
Machines de Turing



entrée :	$\cdots$	.	1	0	1	$\underline{1}$	.	$\cdots$
étape 1 :	$\cdots$	.	1	0	$\underline{1}$	0	.	$\cdots$
étape 2 :	$\cdots$	.	1	$\underline{0}$	0	0	.	$\cdots$

Mémoire en mode lecture/écriture : on agit sur un ruban sur lequel on lit et on écrit des caractères.

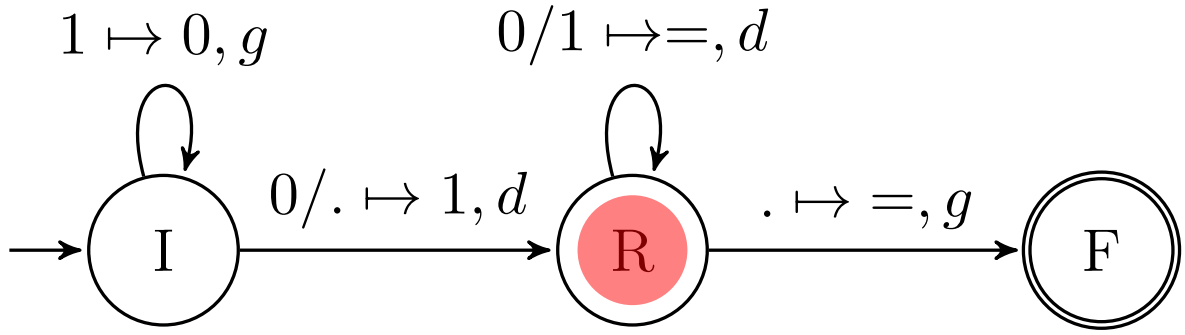
Machines de Turing



entrée :	$\cdots$	.	1	0	1	$\underline{1}$	.	$\cdots$
étape 1 :	$\cdots$	.	1	0	$\underline{1}$	0	.	$\cdots$
étape 2 :	$\cdots$	.	1	$\underline{0}$	0	0	.	$\cdots$
étape 3 :	$\cdots$	.	1	1	$\underline{0}$	0	.	$\cdots$

Mémoire en mode lecture/écriture : on agit sur un ruban sur lequel on lit et on écrit des caractères.

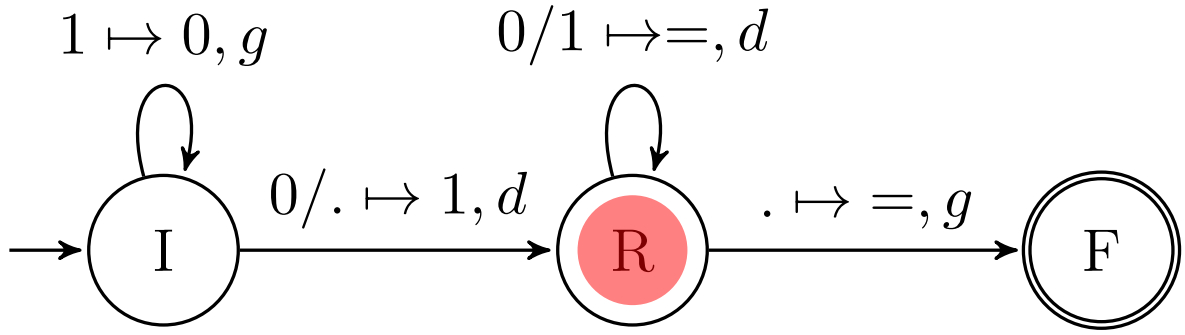
Machines de Turing



entrée :	$\cdots$	.	1	0	1	$\underline{1}$	.	$\cdots$
étape 1 :	$\cdots$	.	1	0	$\underline{1}$	0	.	$\cdots$
étape 2 :	$\cdots$	.	1	$\underline{0}$	0	0	.	$\cdots$
étape 3 :	$\cdots$	.	1	1	$\underline{0}$	0	.	$\cdots$
étape 4 :	$\cdots$	.	1	1	0	$\underline{0}$	.	$\cdots$

Mémoire en mode lecture/écriture : on agit sur un ruban sur lequel on lit et on écrit des caractères.

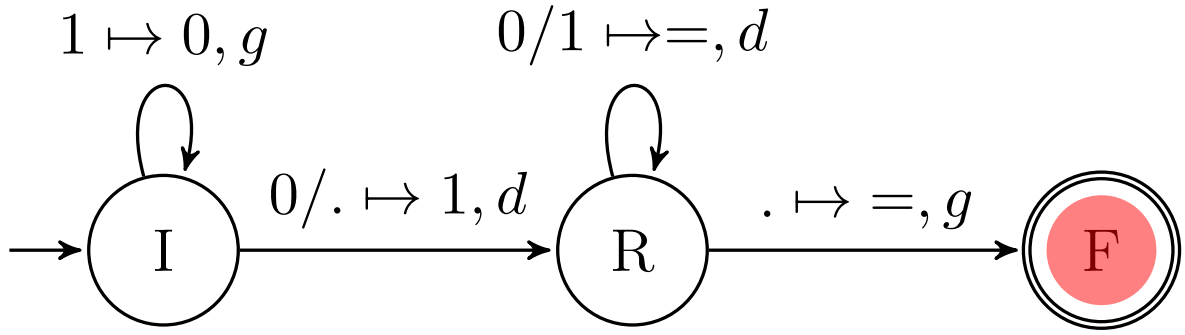
Machines de Turing



entrée :	$\cdots$	.	1	0	1	$\underline{1}$	.	$\cdots$
étape 1 :	$\cdots$	.	1	0	$\underline{1}$	0	.	$\cdots$
étape 2 :	$\cdots$	.	1	$\underline{0}$	0	0	.	$\cdots$
étape 3 :	$\cdots$	.	1	1	$\underline{0}$	0	.	$\cdots$
étape 4 :	$\cdots$	.	1	1	0	$\underline{0}$	.	$\cdots$
étape 5 :	$\cdots$	.	1	1	0	0	$\underline{.}$	$\cdots$

Mémoire en mode lecture/écriture : on agit sur un ruban sur lequel on lit et on écrit des caractères.

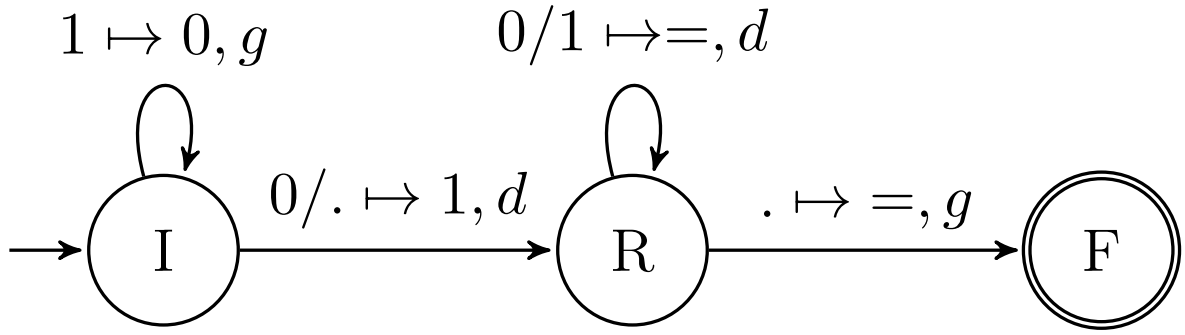
Machines de Turing



entrée :	$\cdots$	.	1	0	1	<u>1</u>	.	$\cdots$
étape 1 :	$\cdots$	.	1	0	<u>1</u>	0	.	$\cdots$
étape 2 :	$\cdots$	.	1	<u>0</u>	0	0	.	$\cdots$
étape 3 :	$\cdots$	.	1	1	<u>0</u>	0	.	$\cdots$
étape 4 :	$\cdots$	.	1	1	0	<u>0</u>	.	$\cdots$
étape 5 :	$\cdots$	.	1	1	0	0	<u>.</u>	$\cdots$
étape 6 :	$\cdots$	.	1	1	0	<u>0</u>	.	$\cdots$

Mémoire en mode lecture/écriture : on agit sur un ruban sur lequel on lit et on écrit des caractères.

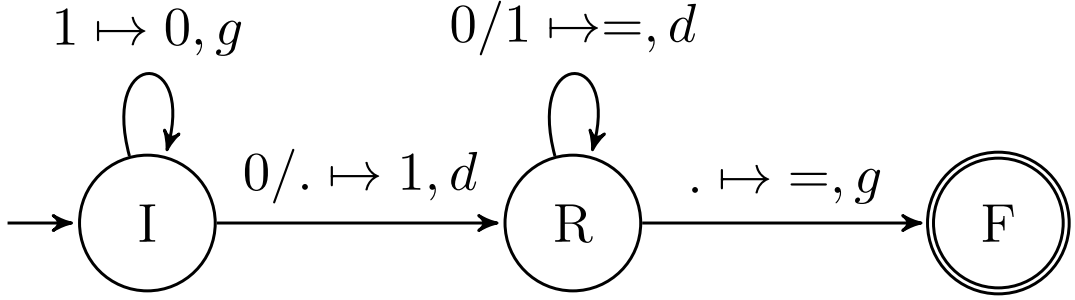
Machines de Turing



entrée :	$\cdots$	.	1	0	1	<u>1</u>	.	$\cdots$
étape 1 :	$\cdots$	.	1	0	<u>1</u>	0	.	$\cdots$
étape 2 :	$\cdots$	.	1	<u>0</u>	0	0	.	$\cdots$
étape 3 :	$\cdots$	.	1	1	<u>0</u>	0	.	$\cdots$
étape 4 :	$\cdots$	.	1	1	0	<u>0</u>	.	$\cdots$
étape 5 :	$\cdots$	.	1	1	0	0	<u>.</u>	$\cdots$
étape 6 :	$\cdots$	.	1	1	0	<u>0</u>	.	$\cdots$

Qu'a-t-on calculé ?

Définitions



Une machine de Turing M est la donnée de :

- un ensemble fini d'états E_M ;
- un état initial $I_M \in E_M$;
- un ensemble d'états finaux $F_M \subset E_M$;
- une fonction de transition

$$\delta_M : (E_M \setminus F_M) \times \{0, 1, .\} \to E_M \times \{0, 1, .\} \times \{g, d\}.$$

$\delta_M(e, a) = (f, b, c)$ signifie que dans l'état e, en lisant la lettre a, on passe dans l'état f en écrivant la lettre b à la place et en suivant la direction c (g pour gauche et d pour droite)

$\cdots$	-3	-2	-1	0	1	2	$\cdots$
$\cdots$	.	1	1	0	0	.	$\cdots$

Un ruban est une fonction $r : \mathbf{Z} \to \{0, 1, .\}$ à support fini.
($r(i) = .$ pour presque tout i)

Calcul

- Une configuration de M est un couple (e, r) formé d'un état et d'un ruban.
- (e, r) est initiale si $e = I_M$ et finale si $e \in F_M$.
- Si (e, r) n'est pas finale, on passe à la configuration suivante $\Delta_M(e, r) = (e', r')$:
  - on calcule la transition : $(e', a', m) = \delta_M(e, r(0))$;
  - on met à jour : $r_0(0) = a'$ et $r_0(i) = r(i)$ si $i \neq 0$;
  - on décale :
    - $r'(i) = r_0(i + 1)$ si $m = g$,
    - $r'(i) = r_0(i - 1)$ sinon.

Calcul

- Une configuration de M est un couple (e, r) formé d'un état et d'un ruban.
- (e, r) est initiale si $e = I_M$ et finale si $e \in F_M$.
- Si (e, r) n'est pas finale, on passe à la configuration suivante $\Delta_M(e, r) = (e', r')$:
  - on calcule la transition : $(e', a', m) = \delta_M(e, r(0))$;
  - on met à jour : $r_0(0) = a'$ et $r_0(i) = r(i)$ si $i \neq 0$;
  - on décale :
    - $r'(i) = r_0(i + 1)$ si $m = g$,
    - $r'(i) = r_0(i - 1)$ sinon.

Éclairant, non ?

Pourquoi seulement des 0 et des 1 ?

On pourrait travailler avec un ensemble de symboles quelconques :

- les chiffres décimaux : 0, 1, ... 9 ;
- tous les caractères disponibles sur un clavier ;
- un seul symbole (trou sur une carte) ;
- ...

Pourquoi seulement des 0 et des 1 ?

On pourrait travailler avec un ensemble de symboles quelconques :

- les chiffres décimaux : 0, 1, ... 9 ;
- tous les caractères disponibles sur un clavier ;
- un seul symbole (trou sur une carte) ;
- ...

Est-ce que ça change quelque chose ?

Pourquoi seulement des 0 et des 1 ?

On pourrait travailler avec un ensemble de symboles quelconques :

- les chiffres décimaux : 0, 1, ... 9 ;
- tous les caractères disponibles sur un clavier ;
- un seul symbole (trou sur une carte) ;
- ...

Est-ce que ça change quelque chose ?

Pas vraiment : on peut tout coder en binaire.

Exemple

$0 \mapsto 0000$, $1 \mapsto 0001$, $2 \mapsto 0010$, $3 \mapsto 0011$, $4 \mapsto 0100$, $5 \mapsto 0101$, $6 \mapsto 0110$, $7 \mapsto 0111$, $8 \mapsto 1000$, $9 \mapsto 1001$.

Pourquoi seulement des 0 et des 1 ?

On pourrait travailler avec un ensemble de symboles quelconques :

- les chiffres décimaux : 0, 1, ... 9 ;
- tous les caractères disponibles sur un clavier ;
- un seul symbole (trou sur une carte) ;
- ...

Est-ce que ça change quelque chose ?

Pas vraiment : on peut tout coder en binaire.

Exemple

$0 \mapsto 0000$, $1 \mapsto 0001$, $2 \mapsto 0010$, $3 \mapsto 0011$, $4 \mapsto 0100$, $5 \mapsto 0101$, $6 \mapsto 0110$, $7 \mapsto 0111$, $8 \mapsto 1000$, $9 \mapsto 1001$.

mais ça va un peu moins vite...

Pourquoi seulement un ruban ?

On pourrait travailler sur plusieurs rubans simultanément.

Pourquoi seulement un ruban ?

On pourrait travailler sur plusieurs rubans simultanément.

Est-ce que ça change quelque chose ?

Pourquoi seulement un ruban ?

On pourrait travailler sur plusieurs rubans simultanément.

Est-ce que ça change quelque chose ?

Pas vraiment : ça revient à changer les symboles.

$\cdots$	.	1	$\underline{0}$	0	.	.	$\cdots$
$\cdots$	.	.	0	1	$\underline{.}$	.	$\cdots$
$\cdots$	.	.	0	$\underline{0}$	0	.	$\cdots$
$\cdots$	.	1	$\underline{0}$	.	0	.	$\cdots$

$\mapsto$

$\cdots$	.	1	$\underline{0}$	0	.	$\underline{.}$	$\cdots$
$\cdots$	.	.	0	1	$\underline{.}$	$\underline{.}$	$\cdots$
$\cdots$	.	.	0	$\underline{0}$	0	$\underline{.}$	$\cdots$
$\cdots$	.	1	$\underline{0}$	.	0	$\underline{.}$	$\cdots$

Pourquoi seulement un ruban ?

On pourrait travailler sur plusieurs rubans simultanément.

Est-ce que ça change quelque chose ?

Pas vraiment : ça revient à changer les symboles.

$\cdots$	.	1	$\underline{0}$	0	.	.	$\cdots$
$\cdots$	.	.	0	1	$\underline{.}$	.	$\cdots$
$\cdots$	.	.	0	$\underline{0}$	0	.	$\cdots$
$\cdots$	.	1	$\underline{0}$	.	0	.	$\cdots$

$\mapsto$

$\cdots$	.	1	$\underline{0}$	0	.	$\underline{.}$	$\cdots$
$\cdots$	.	.	0	1	$\underline{.}$	$\underline{.}$	$\cdots$
$\cdots$	.	.	0	$\underline{0}$	0	$\underline{.}$	$\cdots$
$\cdots$	.	1	$\underline{0}$	.	0	$\underline{.}$	$\cdots$

Mais ça va beaucoup moins vite !
Pour une étape de la machine à k rubans, on doit :

- scanner tous les rubans pour déterminer quel symbole se trouve sous chaque tête ;
- revenir dans l'autre sens pour mettre à jour les symboles et les positions des têtes.

Plus généralement

On peut trouver énormément de variantes des machines de Turing...

Plus généralement

On peut trouver énormément de variantes des machines de Turing... mais on peut chaque fois se ramener au cas binaire sur un ruban.

Quand on parlera de complexité, il faudra être plus prudents...

Que peut-on reconnaître avec une machine de Turing ?

Une machine de Turing manipule des mots : on note Σ^* l'ensemble des mots formés sur l'alphabet (= ensemble fini de symboles) Σ.
On peut faire fonctionner une machine de Turing en mode "accepteur", en sélectionnant des états finaux acceptants :

Définition

On dit que M accepte le mot $w \in \Sigma^*$ si :

- en écrivant le mot w sur le ruban (en laissant les cases blanches partout ailleurs),
- en positionnant la tête sur la première lettre de w,
- en faisant tourner la machine jusqu'à atteindre un état final,
- on obtient un état acceptant.

Que peut-on reconnaître avec une machine de Turing ?

Une machine de Turing manipule des mots : on note Σ^* l'ensemble des mots formés sur l'alphabet (= ensemble fini de symboles) Σ.
On peut faire fonctionner une machine de Turing en mode "accepteur", en sélectionnant des états finaux acceptants :

Définition

On dit que M accepte le mot $w \in \Sigma^*$ si :

- en écrivant le mot w sur le ruban (en laissant les cases blanches partout ailleurs),
- en positionnant la tête sur la première lettre de w,
- en faisant tourner la machine jusqu'à atteindre un état final,
- on obtient un état acceptant.

Mais en général on veut calculer des fonctions.

Que peut-on calculer avec une machine de Turing ?

Définition

On dit que la fonction $f : \Sigma^* \to \Sigma^*$ est calculée par une machine de Turing si :

- en écrivant le mot w sur le ruban (en laissant les cases blanches partout ailleurs),
- en positionnant la tête sur la première lettre de w,
- en faisant tourner la machine jusqu'à atteindre un état final,
- on obtient $f(w)$ (avec la tête sur la première lettre et du blanc partout ailleurs).

Que peut-on calculer avec une machine de Turing ?

Définition

On dit que la fonction $f : \Sigma^* \to \Sigma^*$ est calculée par une machine de Turing si :

- en écrivant le mot w sur le ruban (en laissant les cases blanches partout ailleurs),
- en positionnant la tête sur la première lettre de w,
- en faisant tourner la machine jusqu'à atteindre un état final,
- on obtient $f(w)$ (avec la tête sur la première lettre et du blanc partout ailleurs).

Ça généralise le mode accepteur.

Que peut-on calculer avec une machine de Turing ?

Définition

On dit que la fonction $f : \Sigma^* \to \Sigma^*$ est calculée par une machine de Turing si :

- en écrivant le mot w sur le ruban (en laissant les cases blanches partout ailleurs),
- en positionnant la tête sur la première lettre de w,
- en faisant tourner la machine jusqu'à atteindre un état final,
- on obtient $f(w)$ (avec la tête sur la première lettre et du blanc partout ailleurs).

Ça généralise le mode accepteur.

En général on ne travaille pas sur des mots,
et on doit fixer un codage.

Que peut-on calculer avec une machine de Turing ?

Définition

On dit que la fonction $f : \mathbf{N}^k \to \mathbf{N}$ est calculée par une machine de Turing si :

- en écrivant les nombres $n_1, \ldots, n_k$ en binaire sur le ruban (en les séparant par des blancs par exemple),
- et faisant tourner la machine jusqu'à atteindre un état final,
- on obtient $f(n_1, \ldots, n_k)$ écrit en binaire.

Théorème

Les opérations usuelles sont calculables : somme, multiplication, puissance, etc.

Que peut-on calculer avec une machine de Turing ?

Définition

On dit que la fonction $f : \mathbf{N}^k \to \mathbf{N}$ est calculée par une machine de Turing si :

- en écrivant les nombres $n_1, \ldots, n_k$ en binaire sur le ruban (en les séparant par des blancs par exemple),
- et faisant tourner la machine jusqu'à atteindre un état final,
- on obtient $f(n_1, \ldots, n_k)$ écrit en binaire.

Théorème

Les opérations usuelles sont calculables : somme, multiplication, puissance, etc.

Exercices : section 2.

Que peut-on calculer avec une machine de Turing ?

Définition

On dit que la fonction $f : \mathbf{N}^k \to \mathbf{N}$ est calculée par une machine de Turing si :

- en écrivant les nombres $n_1, \ldots, n_k$ en binaire sur le ruban (en les séparant par des blancs par exemple),
- et faisant tourner la machine jusqu'à atteindre un état final,
- on obtient $f(n_1, \ldots, n_k)$ écrit en binaire.

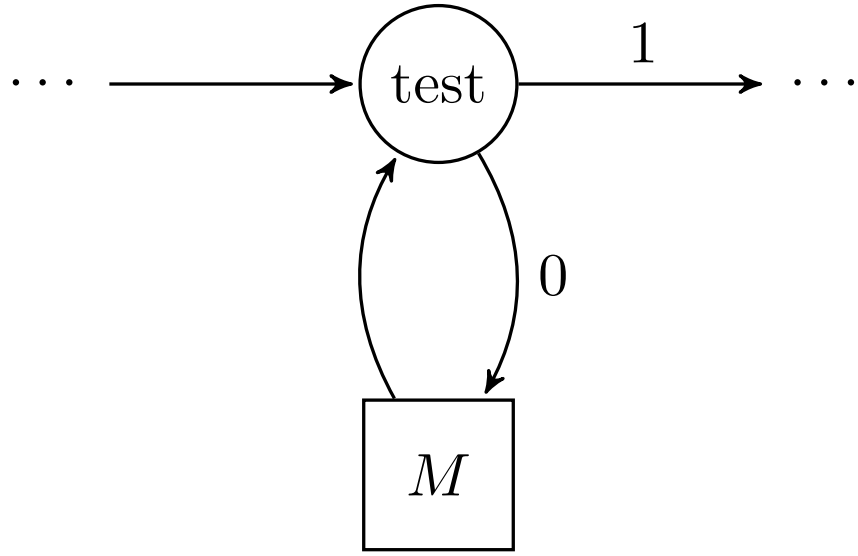
Théorème

Les opérations usuelles sont calculables : somme, multiplication, puissance, etc.

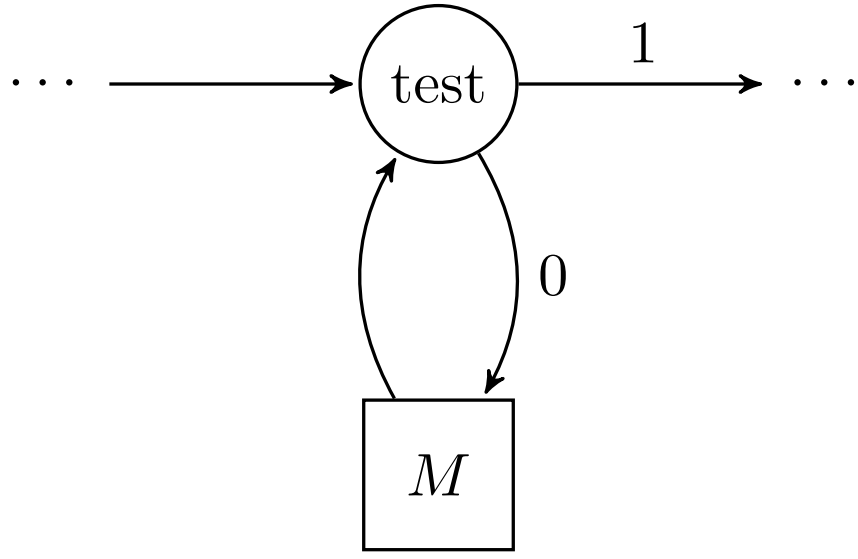
Exercices : section 2.

Mais un ordinateur n'est pas qu'une calculette !

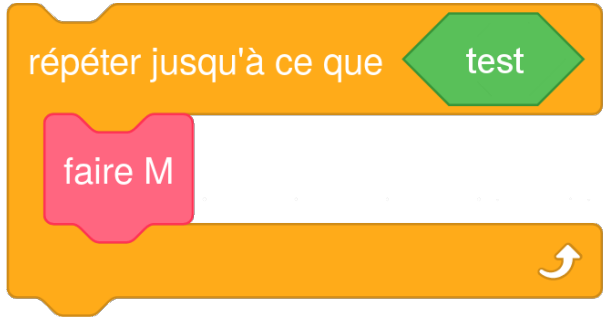
Boucles



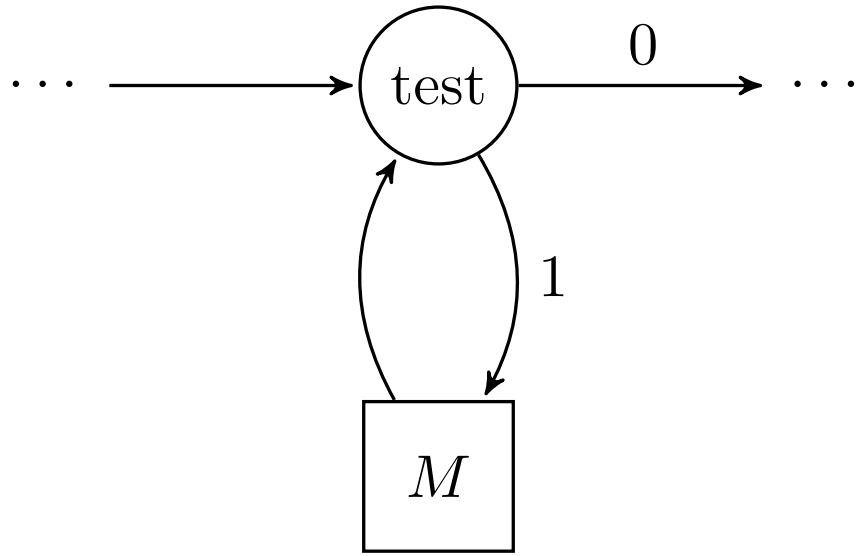
Boucles



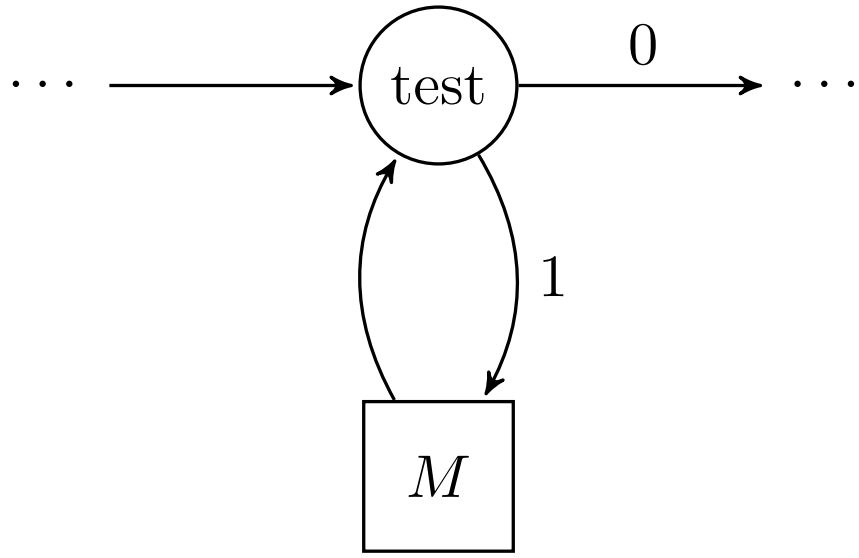
C'est la « boucle » :



Boucles



Boucles

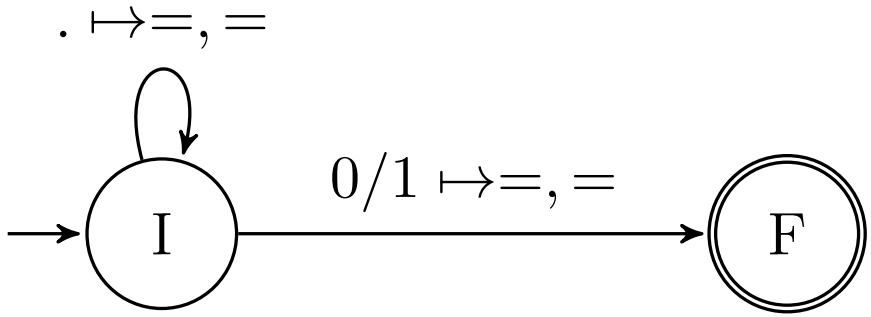


C'est la « boucle » :

```
while test:
        M
```

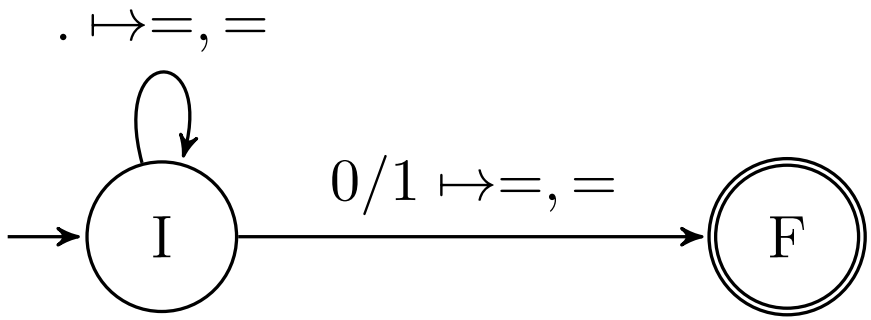
Les bugs

Que calcule la machine suivante ?



Les bugs

Que calcule la machine suivante ?

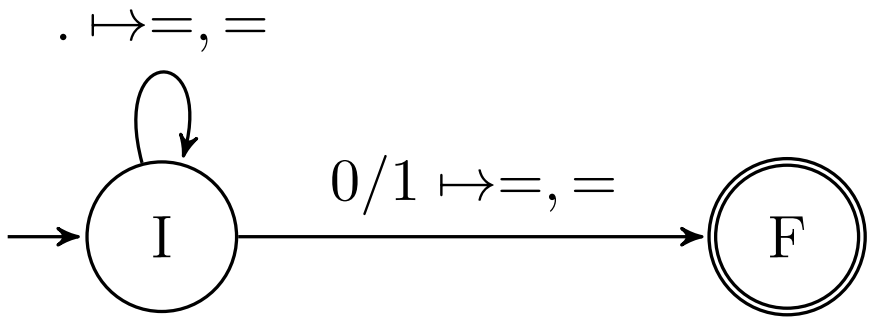


À la louche :

```
def f(x):
        while x == "" :
                pass
        return x
```

Les bugs

Que calcule la machine suivante ?



À la louche :

```
def f(x):
        while x == "" :
                pass
        return x
```

En général, les fonctions qu'on calcule ne sont que partiellement définies !

Codage d'une machine

On peut décrire une machine de Turing sur un ruban en utilisant les symboles :

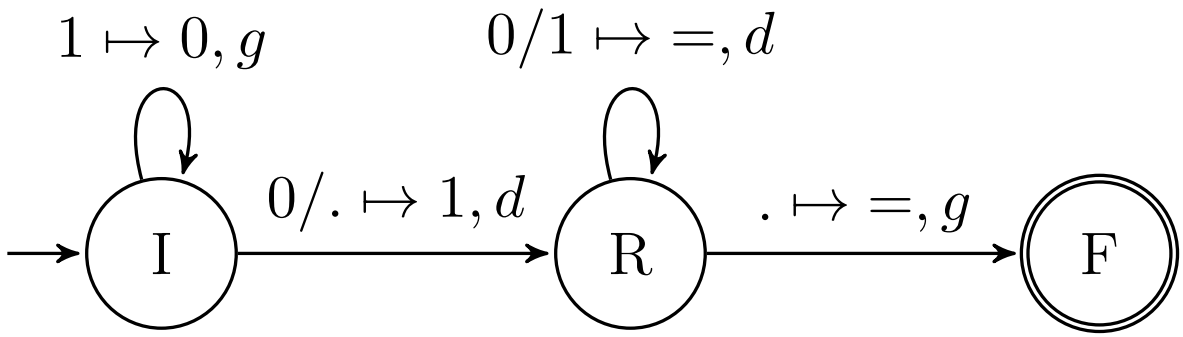
`, 0 1 . g d`

Codage d'une machine

On peut décrire une machine de Turing sur un ruban en utilisant les symboles :

`, 0 1 . g d`

Exemple

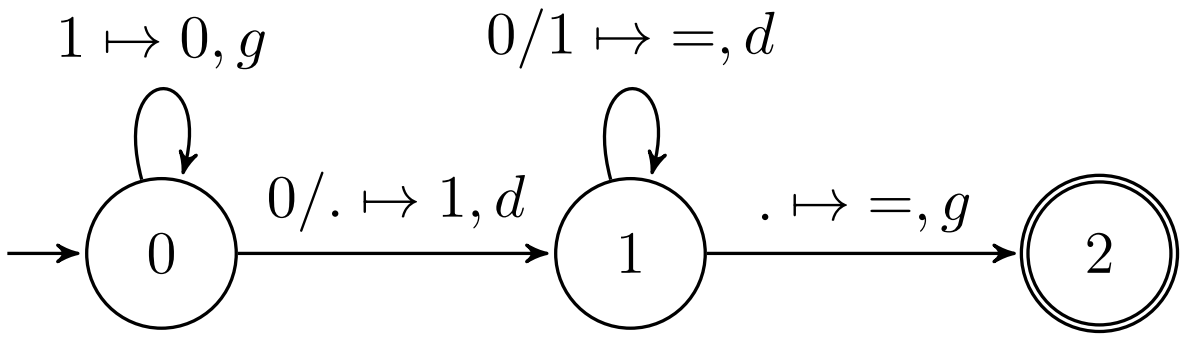


Codage d'une machine

On peut décrire une machine de Turing sur un ruban en utilisant les symboles :

`, 0 1 . g d`

Exemple

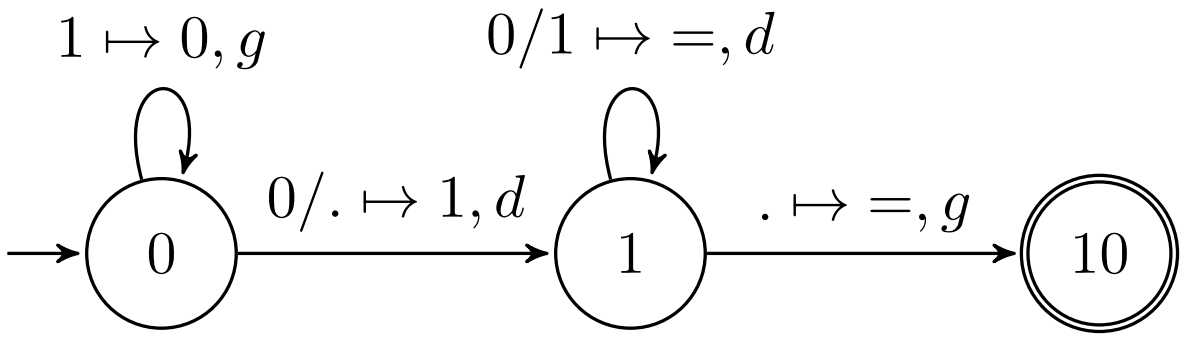


Codage d'une machine

On peut décrire une machine de Turing sur un ruban en utilisant les symboles :

`, 0 1 . g d`

Exemple

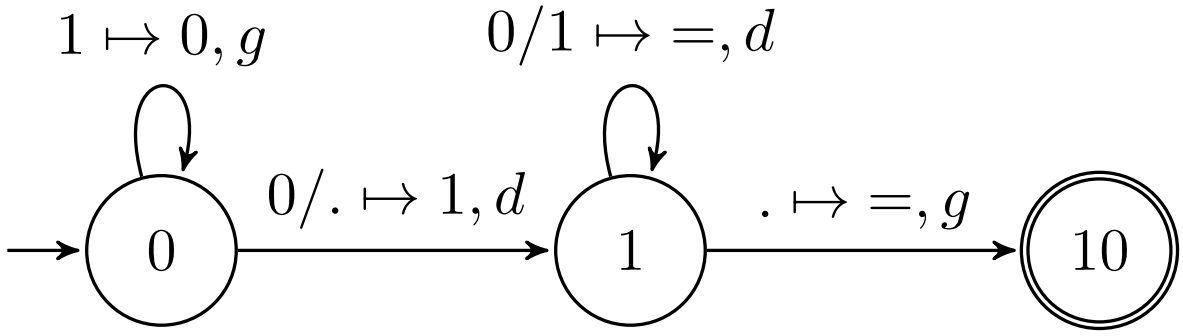


Codage d'une machine

On peut décrire une machine de Turing sur un ruban en utilisant les symboles :

`, 0 1 . g d`

Exemple



`0,.,1,1,d,0,0,1,1,d,0,1,0,0,g,1,.,10,.,g,1,0,1,0,d,1,1,1,1,d,`

c'est la table de transitions :

état courant,symbole lu,état suivant,symbole écrit,déplacement

Machine universelle

Théorème

Il existe une machine de Turing U qui calcule la fonction

(code de la machine M, code du ruban d'entrée) $\mapsto$
résultat de l'exécution de la machine M

Cette machine U simule M en suivant sa description.

Machine universelle

Théorème

Il existe une machine de Turing U qui calcule la fonction

(code de la machine M, code du ruban d'entrée) $\mapsto$
résultat de l'exécution de la machine M

Cette machine U simule M en suivant sa description.

En d'autres mots : U exécute le programme M.

Machine universelle

Théorème

Il existe une machine de Turing U qui calcule la fonction

(code de la machine M, code du ruban d'entrée) $\mapsto$
résultat de l'exécution de la machine M

Cette machine U simule M en suivant sa description.

En d'autres mots : U exécute le programme M.

$$\textit{une machine } M = \textit{un programme pour } U$$

Qu'est-ce qu'un ordinateur ?

Un ordinateur, c'est une machine qu'on peut programmer.

Qu'est-ce qu'un ordinateur ?

Un ordinateur, c'est une machine qu'on peut programmer.

Un ordinateur, c'est une machine de Turing universelle.

Qu'est-ce qu'un ordinateur ?

Un ordinateur, c'est une machine qu'on peut programmer.

Un ordinateur, c'est une machine de Turing universelle.

ou quelque chose qui y ressemble

D'autres modèles de calcul

- les fonctions récursives (Kurt Gödel)
- le λ-calcul (Alonzo Church)
- machines RAM (modèle de nos ordinateurs, Von Neumann)
- machines à compteurs (Marvin Minsky)

D'autres modèles de calcul

- les fonctions récursives (Kurt Gödel)
- le λ-calcul (Alonzo Church)
- machines RAM (modèle de nos ordinateurs, Von Neumann)
- machines à compteurs (Marvin Minsky)
- les automates cellulaires (ex. : jeu de la vie de Conway)

D'autres modèles de calcul

- les fonctions récursives (Kurt Gödel)
- le λ-calcul (Alonzo Church)
- machines RAM (modèle de nos ordinateurs, Von Neumann)
- machines à compteurs (Marvin Minsky)
- les automates cellulaires (ex. : jeu de la vie de Conway)
- la redstone dans Minecraft
- ...

D'autres modèles de calcul

- les fonctions récursives (Kurt Gödel)
- le λ-calcul (Alonzo Church)
- machines RAM (modèle de nos ordinateurs, Von Neumann)
- machines à compteurs (Marvin Minsky)
- les automates cellulaires (ex. : jeu de la vie de Conway)
- la redstone dans Minecraft
- ...
- tous les langages de programmation

D'autres modèles de calcul

- les fonctions récursives (Kurt Gödel)
- le λ-calcul (Alonzo Church)
- machines RAM (modèle de nos ordinateurs, Von Neumann)
- machines à compteurs (Marvin Minsky)
- les automates cellulaires (ex. : jeu de la vie de Conway)
- la redstone dans Minecraft
- ...
- tous les langages de programmation

un langage de programmation =
la description d'un programme universel

C'est toujours la même chose

Ces modèles décrivent la même notion de calcul !

C'est toujours la même chose

Ces modèles décrivent la même notion de calcul !

Exercices : sections 3 et 4.

C'est toujours la même chose

Ces modèles décrivent la même notion de calcul !

Exercices : sections 3 et 4.

Dans chacun de ces modèles il y a un programme universel
et un programme universel peut en simuler un autre. . .

Thèse de Church

« C'est la seule bonne notion de calcul. »

Alonzo Church (1903–1995)

© Princeton University, tiré de Wikimedia Commons

L'incalculable

- Il y a une infinité de fonctions $f : \mathbf{N} \to \mathbf{N}$
- Il y a une infinité de langages $L \subset \Sigma^*$
- Il y a une infinité de machines de Turing

L'incalculable

- Il y a une infinité non dénombrable de fonctions $f : \mathbf{N} \to \mathbf{N}$
- Il y a une infinité non dénombrable de langages $L \subset \Sigma^*$
- Il y a une infinité dénombrable de machines de Turing

L'incalculable

- Il y a une infinité non dénombrable de fonctions $f : \mathbf{N} \to \mathbf{N}$
- Il y a une infinité non dénombrable de langages $L \subset \Sigma^*$
- Il y a une infinité dénombrable de machines de Turing
- Donc il y a infiniment peu de fonctions calculables et de problèmes ($w \in L$?) décidables

L'incalculable

- Il y a une infinité non dénombrable de fonctions $f : \mathbf{N} \to \mathbf{N}$
- Il y a une infinité non dénombrable de langages $L \subset \Sigma^*$
- Il y a une infinité dénombrable de machines de Turing
- Donc il y a infiniment peu de fonctions calculables et de problèmes ($w \in L$?) décidables
- Mais la vraie question est :

$$\text{Y a-t-il des } \begin{cases} \text{fonctions non calculables} \\ \text{problèmes non décidables} \end{cases} \text{ intéressants ?}$$

Le problème de l'arrêt

Le problème de l'arrêt pour un modèle de calcul est la fonction :
$A : (\texttt{P}, \texttt{E}) \mapsto$

$$\begin{cases} 1 & \text{si } \texttt{P} \text{ est un programme dont le calcul sur l'entrée } \texttt{E} \text{ termine} \\ 0 & \text{sinon} \end{cases}$$

Par exemple, en Python, avec :

```
def P(n):
        s = 0
        i = 0
        while i<n:
                s = i+s
        return s
```

Que vaut $A(\texttt{P}, 0)$? Que vaut $A(\texttt{P}, 42)$?

Bug ?

Théorème

Le problème de l'arrêt n'est pas calculable.

Supposons qu'on ait un programme `A` pour le calculer.
On définit un nouveau programme `P` comme suit :

```
def P(f):
        if A(f,f):
                while True:
                        pass
        else:
                return 0
```

Alors que calcule `P(P)` ?

Bug ?

Théorème

Le problème de l'arrêt n'est pas calculable.

Supposons qu'on ait un programme `A` pour le calculer.
On définit un nouveau programme `P` comme suit :

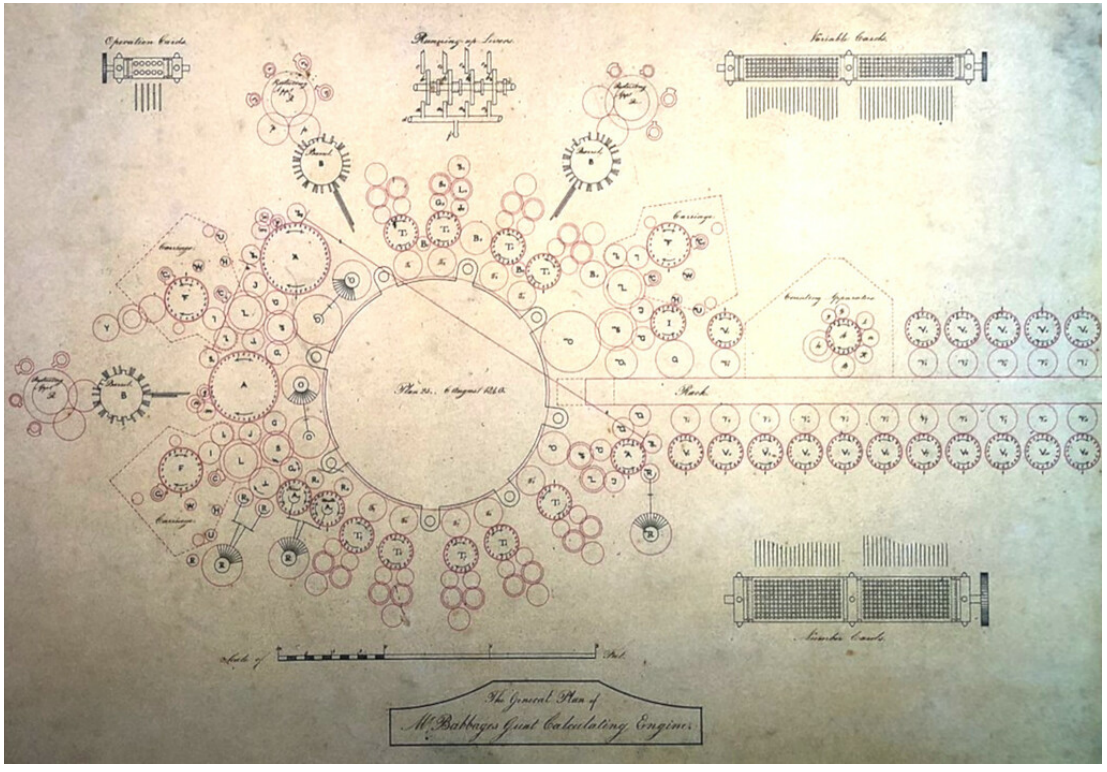
```
def P(f):
        if A(f,f):
                while True:
                        pass
        else:
                return 0
```

Alors que calcule `P(P)` ?

Exercices : section 5.

Le premier ordinateur (?)

La machine analytique de Charles Babbage (1791–1871)



Le premier concept (jamais réalisé) d'une machine universelle

gauche : © ArnoldReinhold, CC-BY-4.0, tiré de Wikimedia Commons

droite : domaine public, tiré de Wikimedia Commons

La programmation avant les ordinateurs

années 1940 premiers calculateurs universels en fonctionnement

La programmation avant les ordinateurs

années 1940 premiers calculateurs universels en fonctionnement

années 1930 travaux de Church, Gödel, Turing, *etc.*

La programmation avant les ordinateurs

années 1940	premiers calculateurs universels en fonctionnement
années 1930	travaux de Church, Gödel, Turing, *etc.*
1843	premier programme informatique : la note G

La note G d'Ada Lovelace (1815–1852)

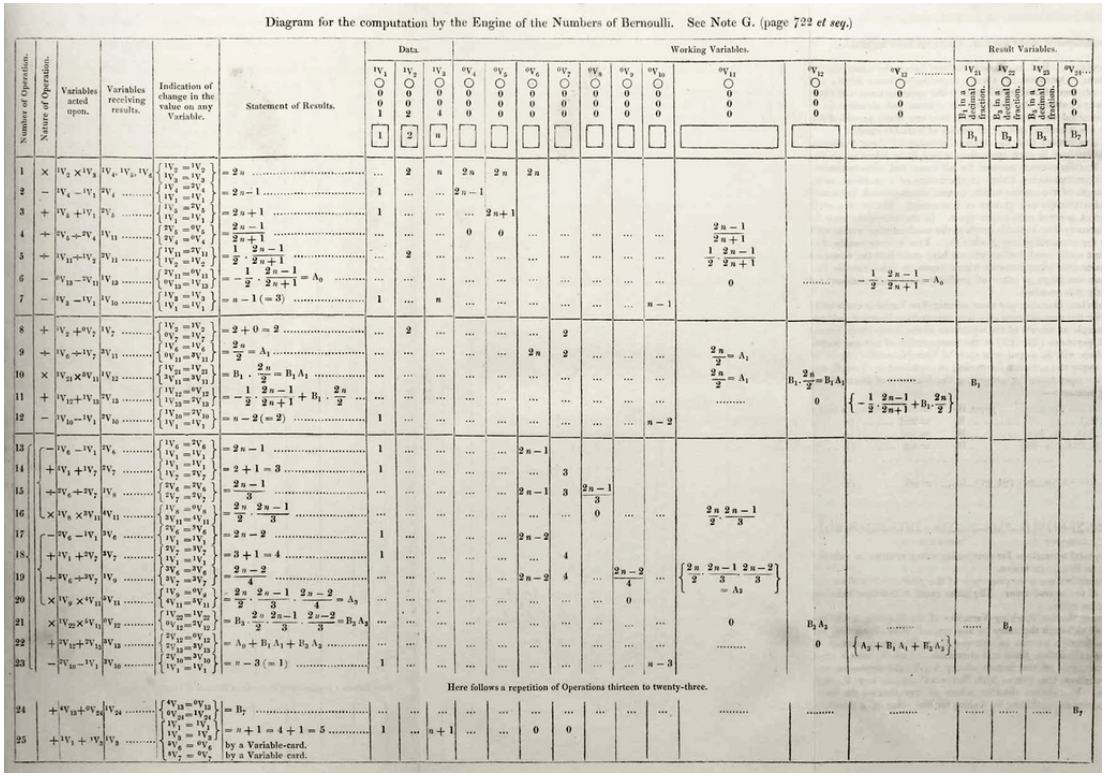


Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 *et seq.*)

Number of Operation.	Nature of Operation.	Variables acted upon.	Variables receiving results.	Indication of change in the value on any Variable.	Statement of Results.	Data.			Working Variables.								Result Variables.			
						$^1V_1$	$^1V_2$	$^1V_3$	$^0V_4$	$^0V_5$	$^0V_6$	$^0V_7$	$^0V_8$	$^0V_9$	$^0V_{10}$	$^0V_{11}$	$^0V_{12}$	$^0V_{13}$	$^1V_{21}$	$^1V_{22}$	$^1V_{23}$	$^0V_{24}$
						1	2	n											B_1	B_2	B_3	B_7
1	×	$^1V_2 \times ^1V_3$	$^1V_4, ^1V_5, ^1V_6$	$^1V_2 = ^1V_2$ $^1V_3 = ^1V_3$	$= 2n$	...	2	n	$2n$	$2n$	$2n$											
2	−	$^1V_4 - ^1V_1$	$^2V_4$	$^1V_4 = ^2V_4$ $^1V_1 = ^1V_1$	$= 2n-1$	1	...	...	$2n-1$													
3	+	$^1V_5 + ^1V_1$	$^2V_5$	$^1V_5 = ^2V_5$ $^1V_1 = ^1V_1$	$= 2n+1$	1	...	...	...	$2n+1$												
4	÷	$^2V_5 \div ^2V_4$	$^1V_{11}$	$^2V_5 = ^0V_5$ $^2V_4 = ^0V_4$	$= \frac{2n-1}{2n+1}$	...	...	...	0	0	...	...	...	...	...	$\frac{2n-1}{2n+1}$						
5	÷	$^1V_{11} \div ^1V_2$	$^2V_{11}$	$^1V_{11} = ^2V_{11}$ $^1V_2 = ^1V_2$	$= \frac{1}{2} \cdot \frac{2n-1}{2n+1}$	...	2	...	...	...	...	...	...	...	...	$\frac{1}{2} \cdot \frac{2n-1}{2n+1}$						
6	−	$^2V_{13} - ^2V_{11}$	$^1V_{13}$	$^2V_{13} = ^0V_{13}$ $^2V_{11} = ^1V_{13}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$	...	...	...	...	...	...	...	...	...	...	0	.....	$-\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$				
7	−	$^1V_3 - ^1V_1$	$^1V_{10}$	$^1V_3 = ^1V_3$ $^1V_1 = ^1V_1$	$= n-1\ (=3)$	1	...	n	...	...	...	...	...	...	$n-1$							
8	+	$^1V_2 + ^0V_7$	$^1V_7$	$^1V_2 = ^1V_2$ $^0V_7 = ^1V_7$	$= 2+0 = 2$	...	2	...	...	...	...	2										
9	÷	$^1V_6 \div ^1V_7$	$^2V_{11}$	$^1V_6 = ^1V_6$ $^0V_{11} = ^2V_{11}$	$= \frac{2n}{2} = A_1$	...	...	...	...	...	$2n$	2	...	...	...	$\frac{2n}{2} = A_1$						
10	×	$^2V_{23} \times ^2V_{11}$	$^1V_{12}$	$^1V_{23} = ^0V_{23}$ $^2V_{11} = ^1V_{11}$	$= B_1 \cdot \frac{2n}{2} = B_1 A_1$	...	...	...	...	...	...	...	...	...	...	$\frac{2n}{2} = A_1$	$B_1 \cdot \frac{2n}{2} = B_1 A_1$	.....	B_1			
11	+	$^1V_{12} + ^1V_{13}$	$^2V_{13}$	$^1V_{12} = ^0V_{12}$ $^1V_{13} = ^0V_{13}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}$	...	...	...	...	...	...	...	...	...	...	.....	0	$\left\{-\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}\right\}$				
12	−	$^1V_{10} - ^1V_1$	$^2V_{10}$	$^1V_{10} = ^2V_{10}$ $^1V_1 = ^1V_1$	$= n-2\ (=2)$	1	...	...	...	...	...	...	...	...	$n-2$							
13	−	$^1V_6 - ^1V_1$	$^2V_6$	$^1V_6 = ^2V_6$ $^1V_1 = ^1V_1$	$= 2n-1$	1	...	...	...	...	$2n-1$											
14	+	$^1V_3 + ^1V_7$	$^2V_7$	$^1V_3 = ^1V_3$ $^1V_7 = ^2V_7$	$= 2+1 = 3$	1	...	...	...	...	...	3										
15	÷	$^2V_6 \div ^2V_7$	$^1V_8$	$^2V_6 = ^0V_6$ $^2V_7 = ^2V_7$	$= \frac{2n-1}{3}$	...	...	...	...	...	$2n-1$	3	$\frac{2n-1}{3}$									
16	×	$^1V_8 \times ^2V_{11}$	$^2V_{11}$	$^1V_8 = ^0V_8$ $^2V_{11} = ^1V_{11}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3}$	...	...	...	...	...	...	...	0	...	...	$\frac{2n}{2} \cdot \frac{2n-1}{3}$						
17	−	$^2V_6 - ^1V_1$	$^2V_6$	$^2V_6 = ^2V_6$ $^1V_1 = ^1V_1$	$= 2n-2$	1	...	...	...	...	$2n-2$											
18	+	$^1V_3 + ^2V_7$	$^1V_7$	$^1V_3 = ^1V_3$ $^2V_7 = ^1V_7$	$= 3+1 = 4$	1	...	...	...	...	...	4										
19	÷	$^2V_6 \div ^2V_7$	$^2V_9$	$^2V_6 = ^2V_6$ $^2V_7 = ^2V_7$	$= \frac{2n-2}{4}$	...	...	...	...	...	$2n-2$	4	...	$\frac{2n-2}{4}$	...	$\left\{\frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = A_3\right\}$						
20	×	$^2V_9 \times ^0V_{11}$	$^2V_{11}$	$^2V_9 = ^0V_9$ $^0V_{11} = ^2V_{11}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = A_3$	...	...	...	...	...	...	...	...	0								
21	×	$^1V_{22} \times ^0V_{11}$	$^2V_{12}$	$^1V_{22} = ^1V_{22}$ $^0V_{12} = ^2V_{12}$	$= B_3 \cdot \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = B_3 A_3$	...	...	...	...	...	...	...	...	...	...	0	$B_3 A_3$	.....	.....	B_3		
22	+	$^2V_{12} + ^2V_{13}$	$^2V_{13}$	$^2V_{12} = ^0V_{12}$ $^2V_{13} = ^2V_{13}$	$= A_3 + B_1 A_1 + B_2 A_2$	...	...	...	...	...	...	...	...	...	...	.....	0	$\{A_3 + B_1 A_1 + B_2 A_2\}$				
23	−	$^1V_{10} - ^1V_1$	$^2V_{10}$	$^1V_{10} = ^2V_{10}$ $^1V_1 = ^1V_1$	$= n-3\ (=1)$	1	...	...	...	...	...	...	...	...	$n-3$							

Here follows a repetition of Operations thirteen to twenty-three.

24	+	$^4V_{13} + ^0V_2$	$^1V_{24}$	$^4V_{13} = ^0V_{13}$ $^0V_2 = ^1V_2$	$= B_7$	...	...	...	...	...	...	...	...	...	...	.....	.....	.....	.....	.....	.....	B_7
25	+	$^1V_1 + ^1V_3$	$^1V_3$	$^1V_1 = ^1V_1$ $^1V_3 = ^1V_3$ $^0V_2 = ^0V_2$ $^4V_7 = ^0V_7$ by a Variable-card. by a Variable card.	$= n+1 = 4+1 = 5$	1	...	$n+1$	...	...	0	0										

Le tout premier programme informatique

gauche : domaine public, tiré de Wikimedia Commons

droite : © The RedBurn, CC-BY-SA-4.0, tiré de Wikimedia Commons

Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 *et seq.*)

Number of Operation.	Nature of Operation.	Variables acted upon.	Variables receiving results.	Indication of change in the value on any Variable.	Statement of Results.	Data. $^1V_1$ 1 [1]	Data. $^1V_2$ 2 [2]	Data. $^1V_3$ 4 [n]	Working Variables. $^0V_4$ [ ]	$^0V_5$ [ ]	$^0V_6$ [ ]	$^0V_7$ [ ]	$^0V_8$ [ ]	$^0V_9$ [ ]	$^0V_{10}$ [ ]	$^0V_{11}$	$^0V_{12}$	$^0V_{13}$	Result Variables. $^1V_{21}$ B_1	$^1V_{22}$ B_3	$^1V_{23}$ B_5	$^0V_{24}$... B_7
1	×	$^1V_2 \times {}^1V_3$	$^1V_4, {}^1V_5, {}^1V_6$	$\left\{\begin{matrix} {}^1V_2 = {}^1V_2 \\ {}^1V_3 = {}^1V_3 \end{matrix}\right\}$	$= 2n$	...	2	n	$2n$	$2n$	$2n$											
2	−	$^1V_4 - {}^1V_1$	$^2V_4$	$\left\{\begin{matrix} {}^1V_4 = {}^2V_4 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n-1$	1	...	...	$2n-1$													
3	+	$^1V_5 + {}^1V_1$	$^2V_5$	$\left\{\begin{matrix} {}^1V_5 = {}^2V_5 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n+1$	1	...	...	...	$2n+1$												
4	÷	$^2V_5 \div {}^2V_4$	$^1V_{11}$	$\left\{\begin{matrix} {}^2V_5 = {}^0V_5 \\ {}^2V_4 = {}^0V_4 \end{matrix}\right\}$	$= \frac{2n-1}{2n+1}$	...	...	...	0	0	...	...	...	...	...	$\frac{2n-1}{2n+1}$						
5	÷	$^1V_{11} \div {}^1V_2$	$^2V_{11}$	$\left\{\begin{matrix} {}^1V_{11} = {}^2V_{11} \\ {}^1V_2 = {}^1V_2 \end{matrix}\right\}$	$= \frac{1}{2} \cdot \frac{2n-1}{2n+1}$	...	2	...	...	...	...	...	...	...	...	$\frac{1}{2} \cdot \frac{2n-1}{2n+1}$						
6	−	$^0V_{13} - {}^2V_{11}$	$^1V_{13}$	$\left\{\begin{matrix} {}^2V_{11} = {}^0V_{11} \\ {}^0V_{13} = {}^1V_{13} \end{matrix}\right\}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$	...	...	...	...	...	...	...	...	...	...	0	.....	$-\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$				
7	−	$^1V_3 - {}^1V_1$	$^1V_{10}$	$\left\{\begin{matrix} {}^1V_3 = {}^1V_3 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= n-1\ (=3)$	1	...	n	...	...	...	...	...	...	$n-1$							
8	+	$^1V_2 + {}^0V_7$	$^1V_7$	$\left\{\begin{matrix} {}^1V_2 = {}^1V_2 \\ {}^0V_7 = {}^1V_7 \end{matrix}\right\}$	$= 2+0=2$	...	2	...	...	...	...	2										
9	÷	$^1V_6 \div {}^1V_7$	$^3V_{11}$	$\left\{\begin{matrix} {}^1V_6 = {}^1V_6 \\ {}^0V_{11} = {}^3V_{11} \end{matrix}\right\}$	$= \frac{2n}{2} = A_1$	...	...	...	...	...	$2n$	2	...	...	...	$\frac{2n}{2} = A_1$						
10	×	$^1V_{21} \times {}^3V_{11}$	$^1V_{12}$	$\left\{\begin{matrix} {}^1V_{21} = {}^1V_{21} \\ {}^3V_{11} = {}^3V_{11} \end{matrix}\right\}$	$= B_1 \cdot \frac{2n}{2} = B_1 A_1$	...	...	...	...	...	...	...	...	...	...	$\frac{2n}{2} = A_1$	$B_1 \cdot \frac{2n}{2} = B_1 A_1$	.....	B_1			
11	+	$^1V_{12} + {}^1V_{13}$	$^2V_{13}$	$\left\{\begin{matrix} {}^1V_{12} = {}^0V_{12} \\ {}^1V_{13} = {}^2V_{13} \end{matrix}\right\}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}$ ...	...	...	...	...	...	...	...	...	...	...	.....	0	$\left\{-\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}\right\}$				
12	−	$^1V_{10} - {}^1V_1$	$^2V_{10}$	$\left\{\begin{matrix} {}^1V_{10} = {}^2V_{10} \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= n-2\ (=2)$	1	...	...	...	...	...	...	...	...	$n-2$							
13	−	$^1V_6 - {}^1V_1$	$^2V_6$	$\left\{\begin{matrix} {}^1V_6 = {}^2V_6 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n-1$	1	...	...	...	...	$2n-1$											
14	+	$^1V_1 + {}^1V_7$	$^2V_7$	$\left\{\begin{matrix} {}^1V_1 = {}^1V_1 \\ {}^1V_7 = {}^2V_7 \end{matrix}\right\}$	$= 2+1=3$	1	...	...	...	...	...	3										
15	÷	$^2V_6 \div {}^2V_7$	$^1V_8$	$\left\{\begin{matrix} {}^2V_6 = {}^2V_6 \\ {}^2V_7 = {}^2V_7 \end{matrix}\right\}$	$= \frac{2n-1}{3}$	...	...	...	...	...	$2n-1$	3	$\frac{2n-1}{3}$									
16	×	$^1V_8 \times {}^3V_{11}$	$^4V_{11}$	$\left\{\begin{matrix} {}^1V_8 = {}^0V_8 \\ {}^3V_{11} = {}^4V_{11} \end{matrix}\right\}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3}$	...	...	...	...	...	...	...	0	...	...	$\frac{2n}{2} \cdot \frac{2n-1}{3}$						
17	−	$^2V_6 - {}^1V_1$	$^3V_6$	$\left\{\begin{matrix} {}^2V_6 = {}^3V_6 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n-2$	1	...	...	...	...	$2n-2$											
18	+	$^1V_1 + {}^2V_7$	$^3V_7$	$\left\{\begin{matrix} {}^2V_7 = {}^3V_7 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 3+1=4$	1	...	...	...	...	...	4										
19	÷	$^3V_6 \div {}^3V_7$	$^1V_9$	$\left\{\begin{matrix} {}^3V_6 = {}^3V_6 \\ {}^3V_7 = {}^3V_7 \end{matrix}\right\}$	$= \frac{2n-2}{4}$	...	...	...	...	...	$2n-2$	4	...	$\frac{2n-2}{4}$	...	$\left\{\frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{3}\right\} = A_3$						
20	×	$^1V_9 \times {}^4V_{11}$	$^5V_{11}$	$\left\{\begin{matrix} {}^1V_9 = {}^0V_9 \\ {}^4V_{11} = {}^5V_{11} \end{matrix}\right\}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = A_3$	...	...	...	...	...	...	...	...	0								
21	×	$^1V_{22} \times {}^5V_{11}$	$^0V_{12}$	$\left\{\begin{matrix} {}^1V_{22} = {}^1V_{22} \\ {}^0V_{12} = {}^2V_{12} \end{matrix}\right\}$	$= B_3 \cdot \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{3} = B_3 A_3$	...	...	...	...	...	...	...	...	...	...	0	$B_3 A_3$	.....	.....	B_3		
22	+	$^2V_{12} + {}^2V_{13}$	$^3V_{13}$	$\left\{\begin{matrix} {}^2V_{12} = {}^0V_{12} \\ {}^2V_{13} = {}^3V_{13} \end{matrix}\right\}$	$= A_0 + B_1 A_1 + B_3 A_3$	...	...	...	...	...	...	...	...	...	...	.....	0	$\left\{A_3 + B_1 A_1 + B_3 A_3\right\}$				
23	−	$^2V_{10} - {}^1V_1$	$^3V_{10}$	$\left\{\begin{matrix} {}^2V_{10} = {}^3V_{10} \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= n-3\ (=1)$	1	...	...	...	...	...	...	...	...	$n-3$							

Here follows a repetition of Operations thirteen to twenty-three.

24	+	$^4V_{13} + {}^0V_{24}$	$^1V_{24}$	$\left\{\begin{matrix} {}^4V_{13} = {}^0V_{13} \\ {}^0V_{24} = {}^1V_{24} \end{matrix}\right\}$	$= B_7$	...	...	...	...	...	...	...	...	...	...	.....	.....	.....	.....	.....	.....	B_7
25	+	$^1V_1 + {}^1V_3$	$^1V_3$	$\left\{\begin{matrix} {}^1V_1 = {}^1V_1 \\ {}^1V_3 = {}^1V_3 \\ {}^5V_6 = {}^0V_6 \\ {}^5V_7 = {}^0V_7 \end{matrix}\right\}$	$= n+1 = 4+1=5$ ..... by a Variable-card. by a Variable card.	1	...	$n+1$	...	...	0	0										