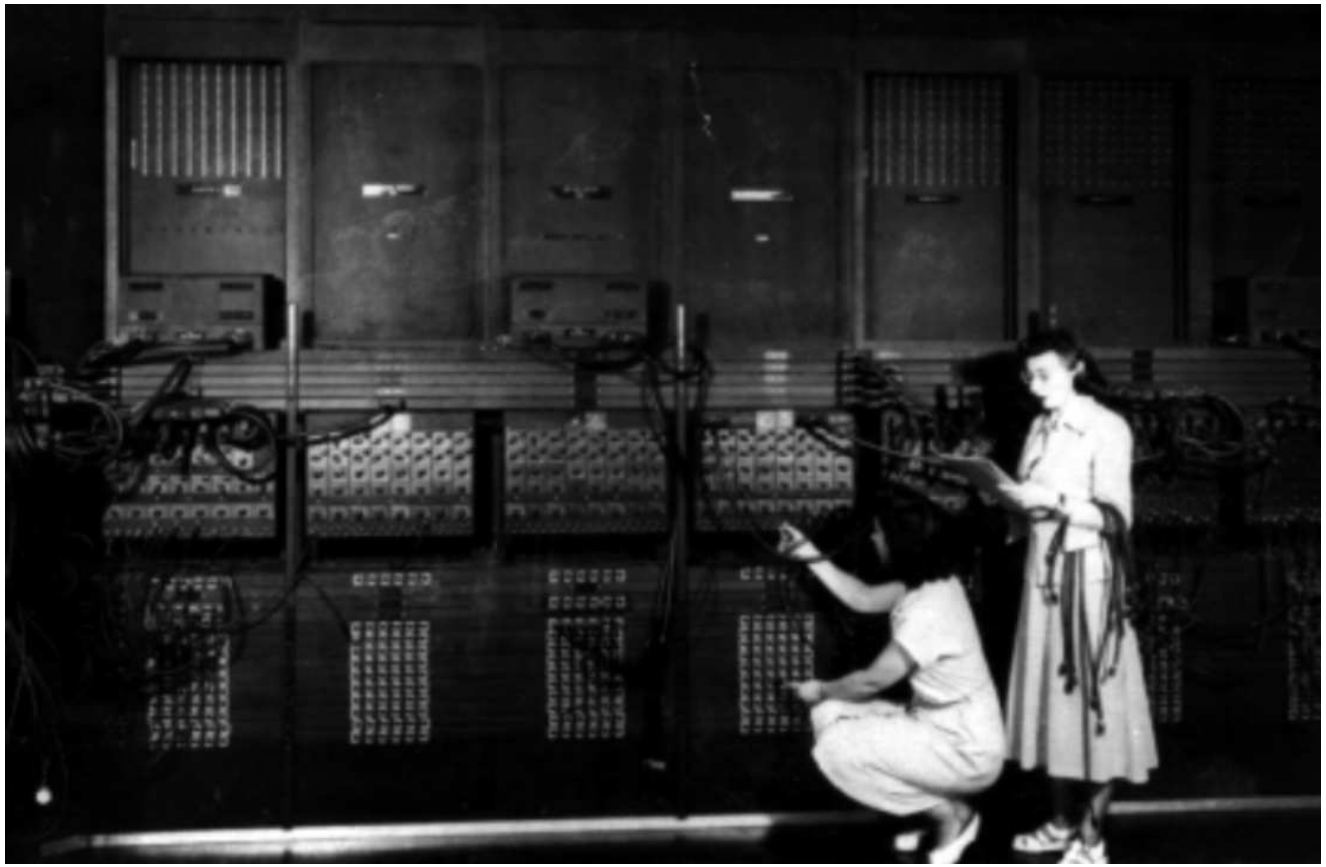
Code source sans code

Le cas de l'ENIAC.



Liesbeth De Mol

CNRS, Université de Lille 3, liesbeth.demol@univ-lille3.fr

Introduction

Objet: Expliquer l'ENIAC original (un premier ordinateur) comme machine programmable et inspiration de la programmation moderne (mais sans code)

Motivations pédagogiques, philosophiques et politiques Revisiter des questions fondamentals d'un point de vue historique et non-statique

- Qu'est-ce qu'est la programmation?
- Qu"est-ce qu'un programme?
- Qu'est ce que code source?
- Qu'est-ce qu'un ordinateur?
- Qu'est-ce qu'est l'informatique
- etc

$\Rightarrow$ Attaquer des malcompréhensions et misréprésentations

$\Rightarrow$ See the wood for the trees again

$\Rightarrow$ Identifier des questions fondamentales de la discipline

Sommaire

- Une introduction dans l'ENIAC
- La notion de programme et de programmation "von Neumannien"
- La notion de programme et de programmation chez Curry
- Discussion

Une introduction dans l'ENIAC

Introduire la machine...

Contexte historique général (1)

Motivation: calcul des tables d'artillerie à BRL et Penn University (problèmes de précisions et de vitesse):

"*You finally get errors in the final result of the order of 1 percent or worse. The ballistics work that we were doing required errors of something like 10 times better than that. So what was being done at the time was that things were being run on the analyzer and then they were being smoothed by hand calculations.*"
(Eckert, 1977)

"*In spite of the extensive arrangements the laboratory made [...] the backlog continued to grow. At one point, more than 100 female students were engaged to carry out firing table calculations. It was to relieve this bottleneck that John W. Mauchly and J. Presper Eckert [...] proposed the construction of the ENIAC*"
(Polachek, 1997)

Introduire la machine...

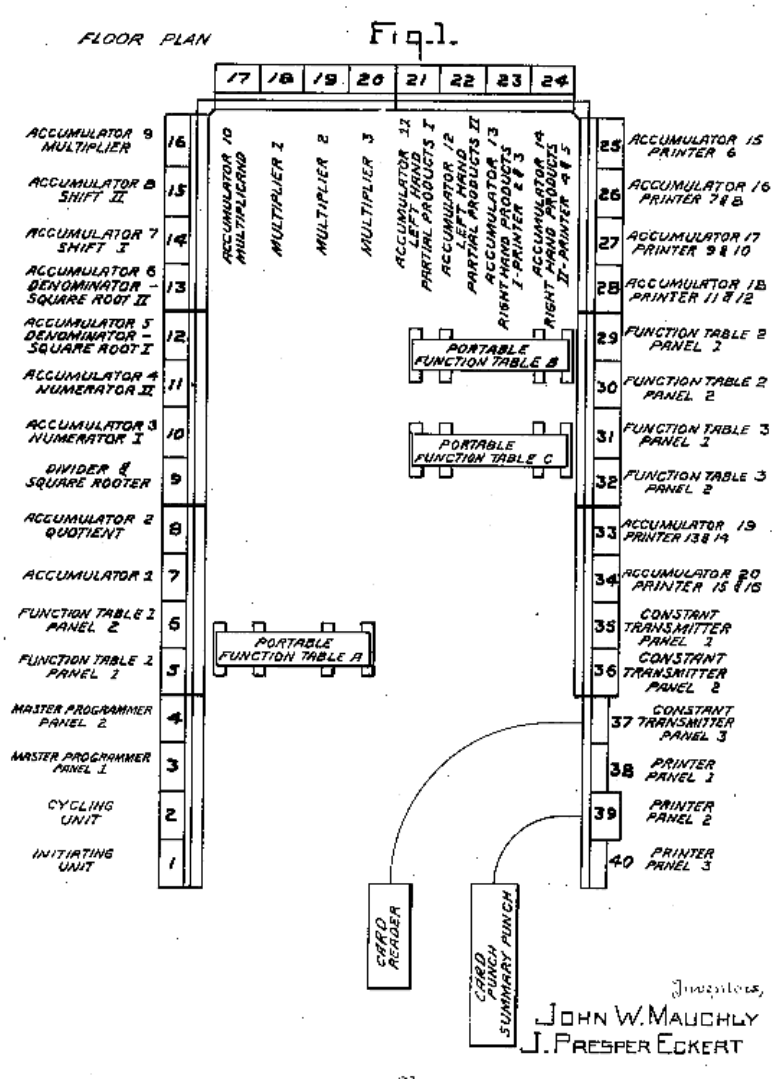
Contexte historique général (2)

En 1941, Mauchly rencontre Presper J. Eckert. Eckert "*was willing and agreeable to talk about the possibility of electronic computers [...] Nobody else really wanted to give it a second thought*" [Mauchly, 1970].

$\Rightarrow$ Proposition formelle au Navy Ordnance pour la construction d'une machine de calcul *électronique*; commencement en 1943.

- Présentation au publique le 15 Février 1946 de l'ENIAC, The Electronic(!) Numerical Integrator And Computer – machine parallèle (!!)
- 18.000 tubes à vide; 1.500 relays et 40 panneaux pour 30 unitès
- Grand vitesse: 5000 addition par seconde!

Introduire la machine: architecture-programmation



Introduire la machine...

Les unités

- Element de départ
- Element cyclique
- 20 accumulateurs (pour l'addition)
- unités pour la multiplication, la divison et des racines carrées
- le "transmetteur de constants" et 3 tables de fonction (les unités de mémoire électronique)
- le "master programmer" (controle des boucles et conditionel)
- un lecteurs de cartes perforées et imprimeur

Introduire la machine...

Principes généraux

1. Deux types de circuits:
   - **circuits numériques:** transmission des signales électroniques qui réprésent des chiffres (0 à 9)
   - **circuits de programmes:** controle de communication entre les différent unités; réglage de “l’ordre” d’un programme, ces différents pas
2. Chaque unité prend un certain nombre de **temps d’addition** et émet un pouls quand finit, alertant les autres unités impliqués – permet la synchronisation
3. **Programmation locale** de chaque unité – panneaux individuels pour chaque unité
4. **Contrôle central** “Master programmer” comme moyen de séquencage et contrôle des boucles

$\Rightarrow$ Un “programme” c’est la structure des connections entre les différents unités par les cables numériques et de programmes et la “programmation” des unités par des commutateurs

Introduire la machine...

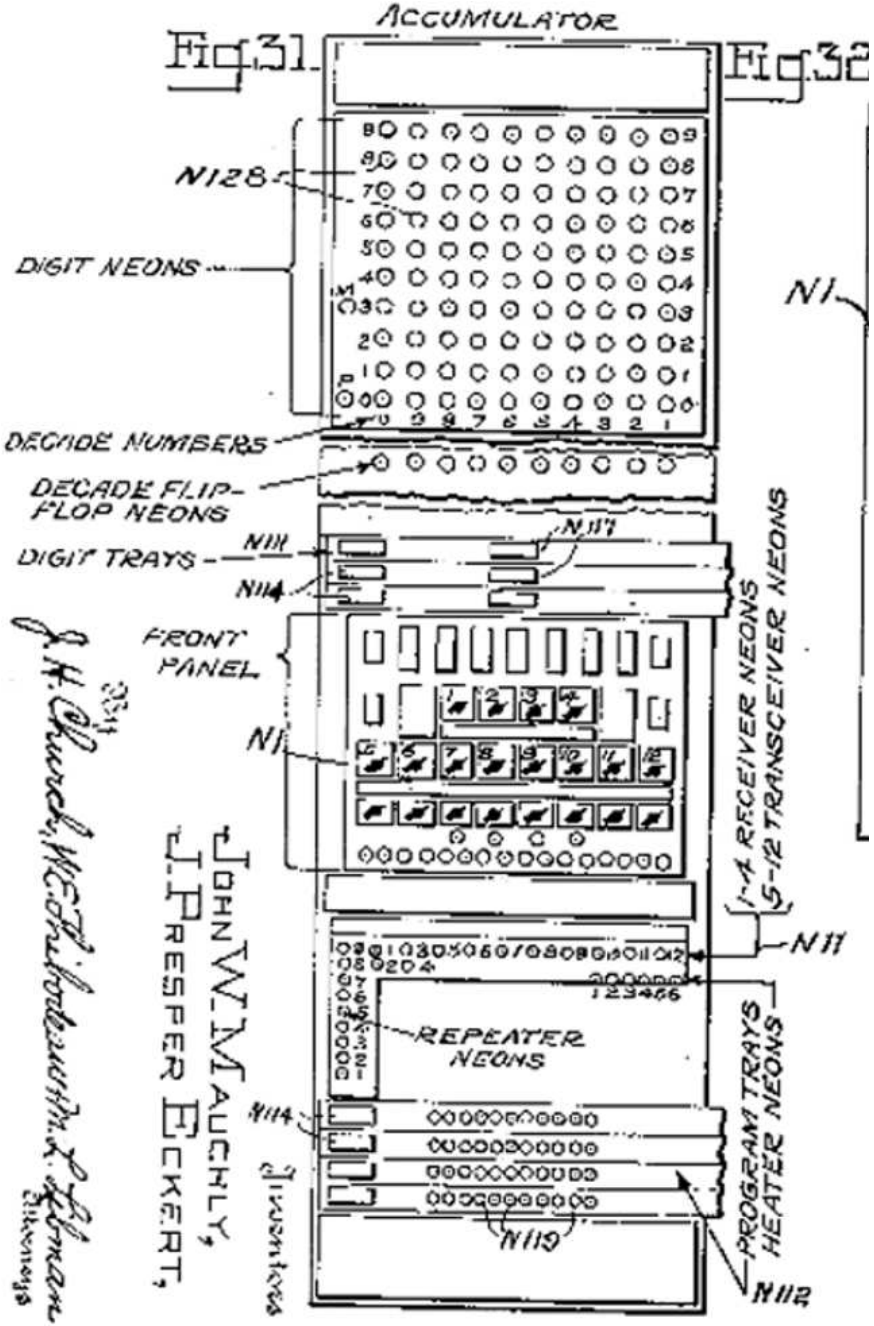
L'accumulateur, unité d'artihmétique La partie numérique

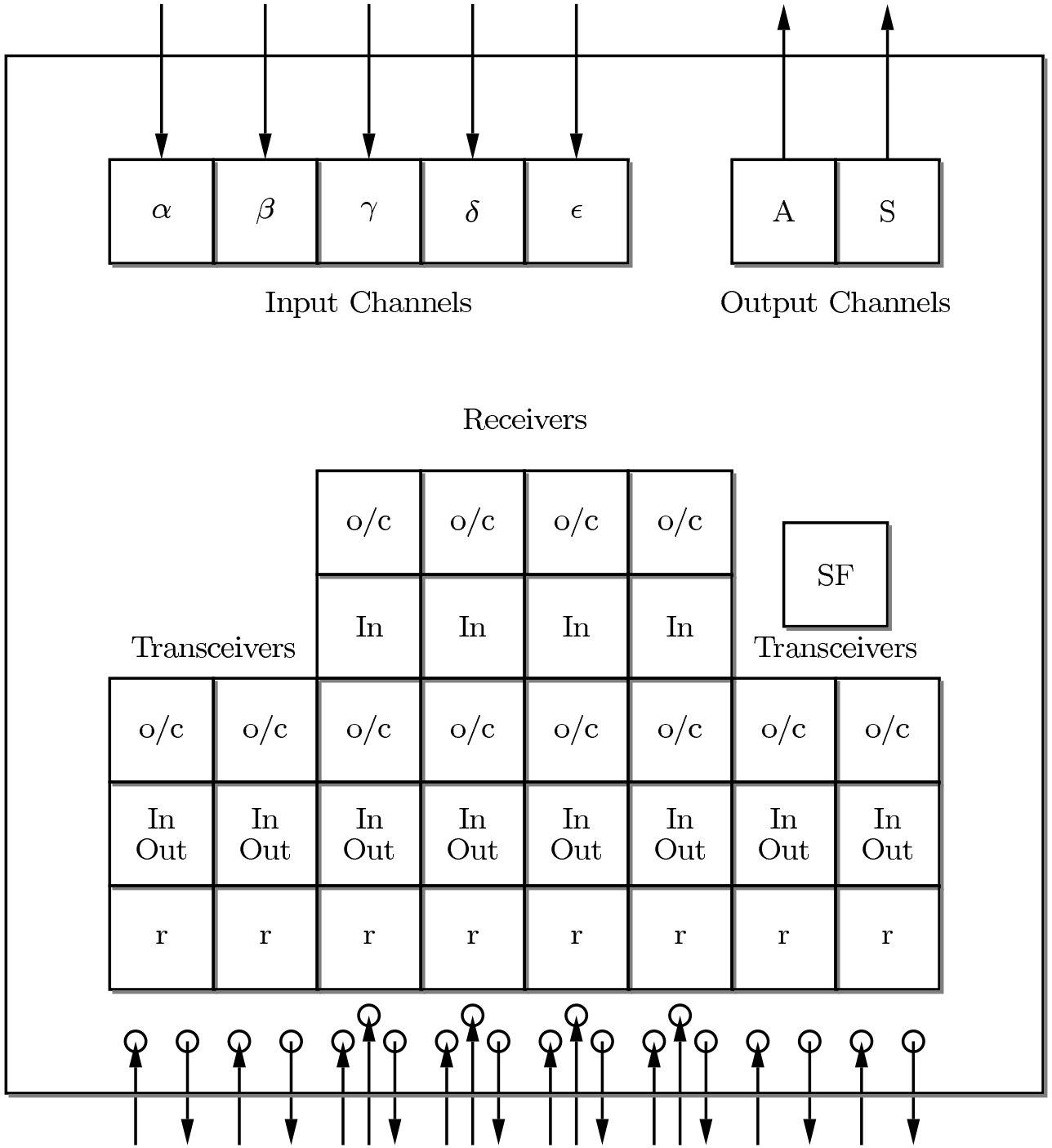
- Chaque peut stocker un nombre décimal signé de 10 chiffres dans 10 "decade ring counters" + PM counter (pour le signe)
- 5 canals d'entrée et 2 de sortie (A et S) pour la transmission d'un nombre n (par A) ou son complément (10^{10}- n)

Introduire la machine...

L'accumulateur, unité d'artihmétique La partie de programmation

* 12 controle de programme; 4 récepteurs et 8 émetteurs-récepteurs
* Emetteur-récepteur: input and output program pulse terminal; clear-correct switch (remettre à zero) et commutateur pour la sélection des canaux d'entrée et de sortie (de α à ϵ, A, S, AS ou O) et commutateur de répétition (recoit ou transmet de 0 à 9 fois)
* Récepteur: pas de "program pulse output terminal" et pas de commutateur de répétition

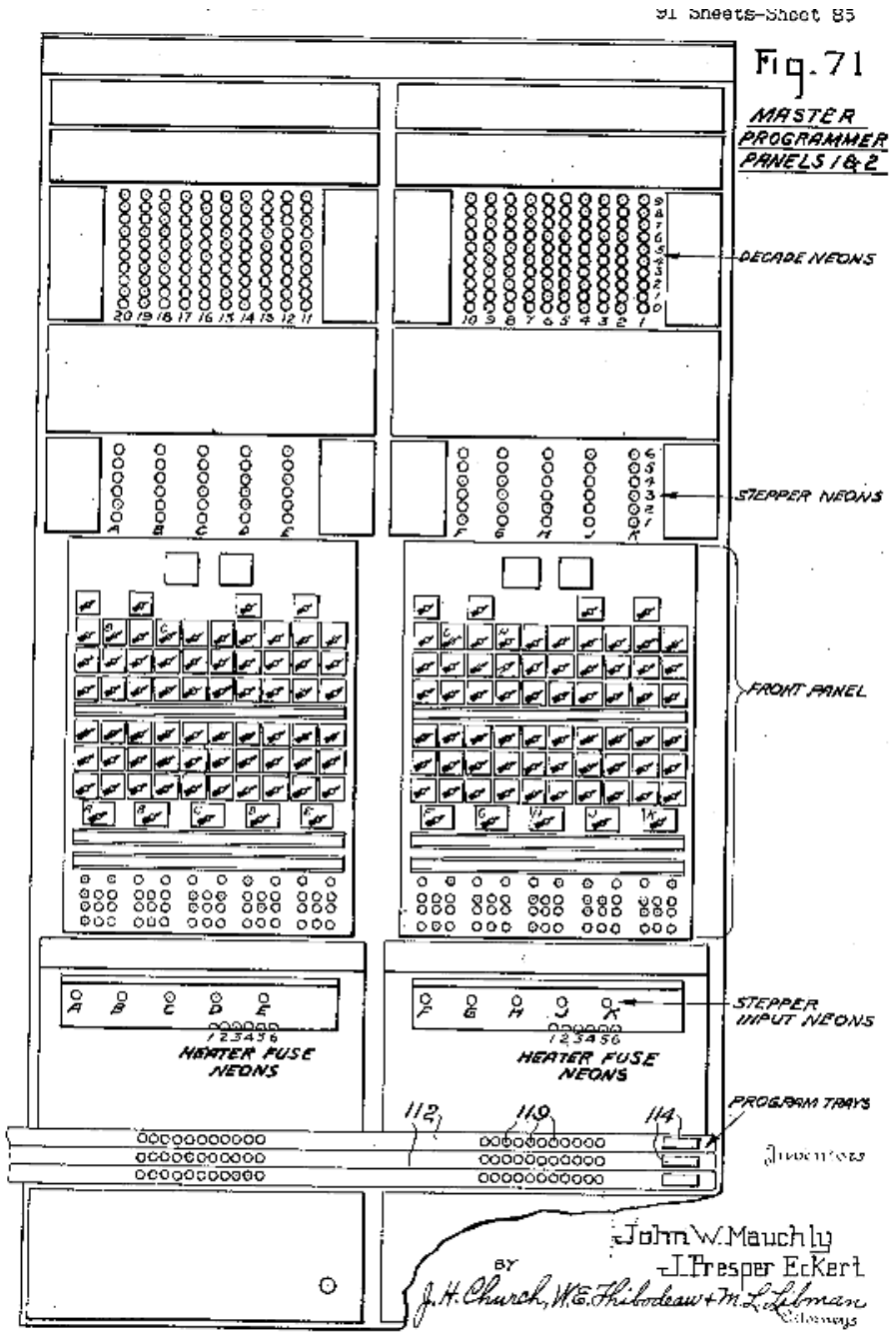

ACCUMULATOR
Fig.31.
Fig.32
N128
DIGIT NEONS
N1
DECADE NUMBERS
DECADE FLIP-FLOP NEONS
DIGIT TRAYS
N111
N114
N117
FRONT PANEL
N1
1-4 RECEIVER NEONS
5-12 TRANSCEIVER NEONS
N11
REPEATER NEONS
N114
N119
PROGRAM TRAYS
HEATER NEONS
N112
JOHN W. MAUCHLY,
J. PRESPER ECKERT,


α
β
γ
δ
ϵ
A
S
Input Channels
Output Channels
Receivers
o/c
In
SF
Transceivers
Transceivers
In
Out
r

Introduire la machine...

"Master programmer": unité de contrôle

- 10 unité indépendant, chaque avec un compteur de 6 étapes ("stepper counter")
- 3 terminals d'entrée pour chaque steppeur (stepper input, direct input et clear input)
- 6 terminals de sortie pour chaque étape d'un steppeur; avec chaque étape on associe un nombre d_s par fixer un commutateur ("decade switches")


91 Sheets-Sheet 85
Fig. 71
MASTER PROGRAMMER PANELS 1 & 2
DECADE NEONS
STEPPER NEONS
FRONT PANEL
STEPPER INPUT NEONS
HEATER FUSE NEONS
HEATER FUSE NEONS
PROGRAM TRAYS
112
119
114
John W. Mauchly
J. Presper Eckert
BY
J. H. Church, W. G. Thibodeau + M. L. Lehman

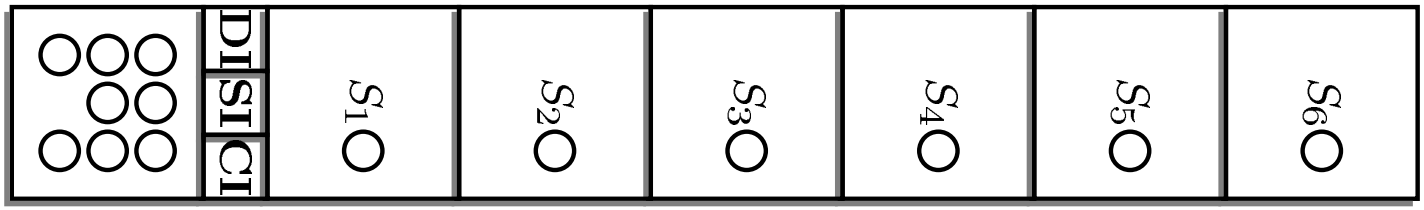
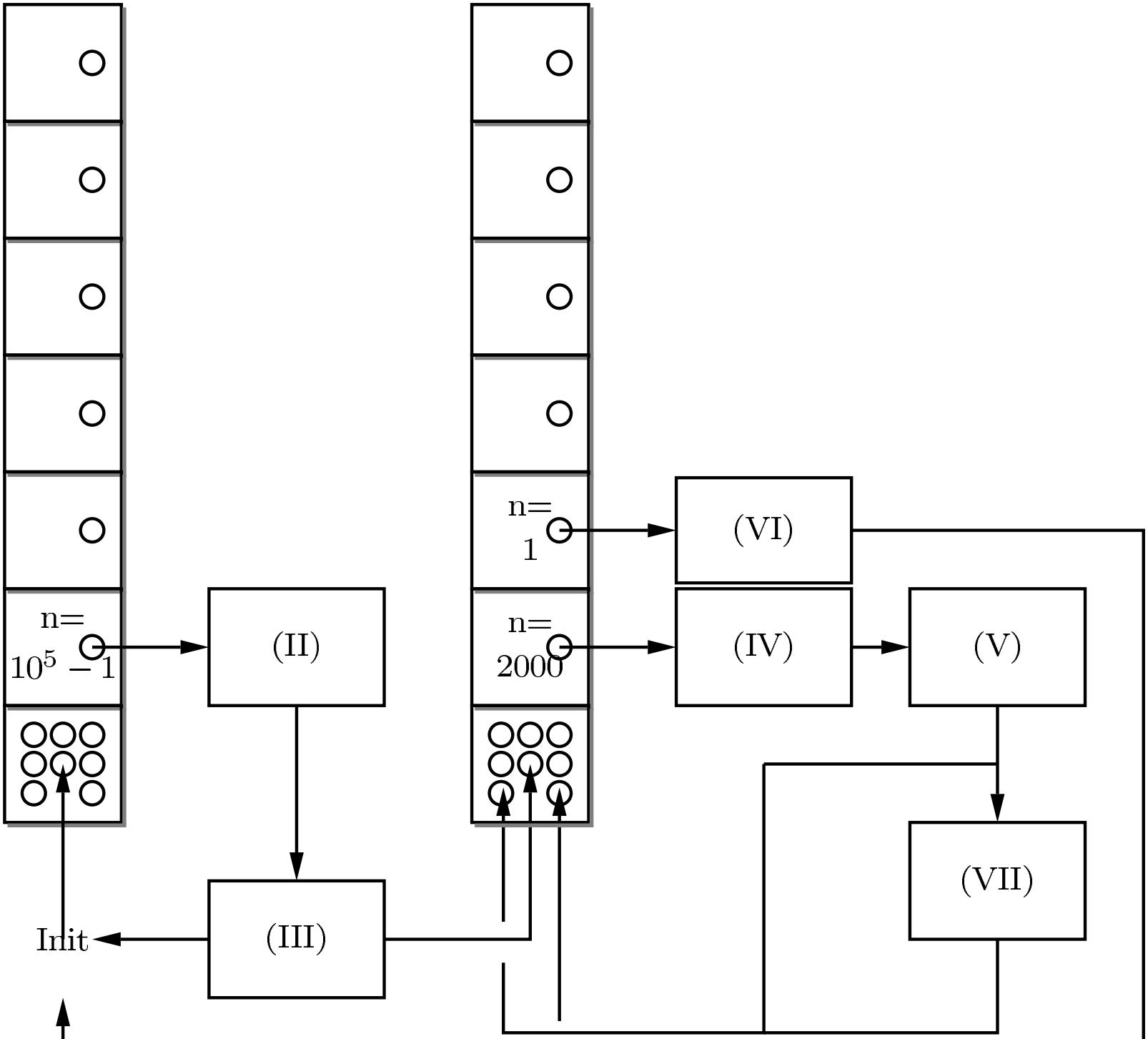


Figure 1: A Schematic (Reduced) Representation of a stepper counter of the Master Programmer.


n=
$10^5 - 1$
n=
1
n=
2000
(II)
(III)
(IV)
(V)
(VI)
(VII)
Init

Introduire la machine...

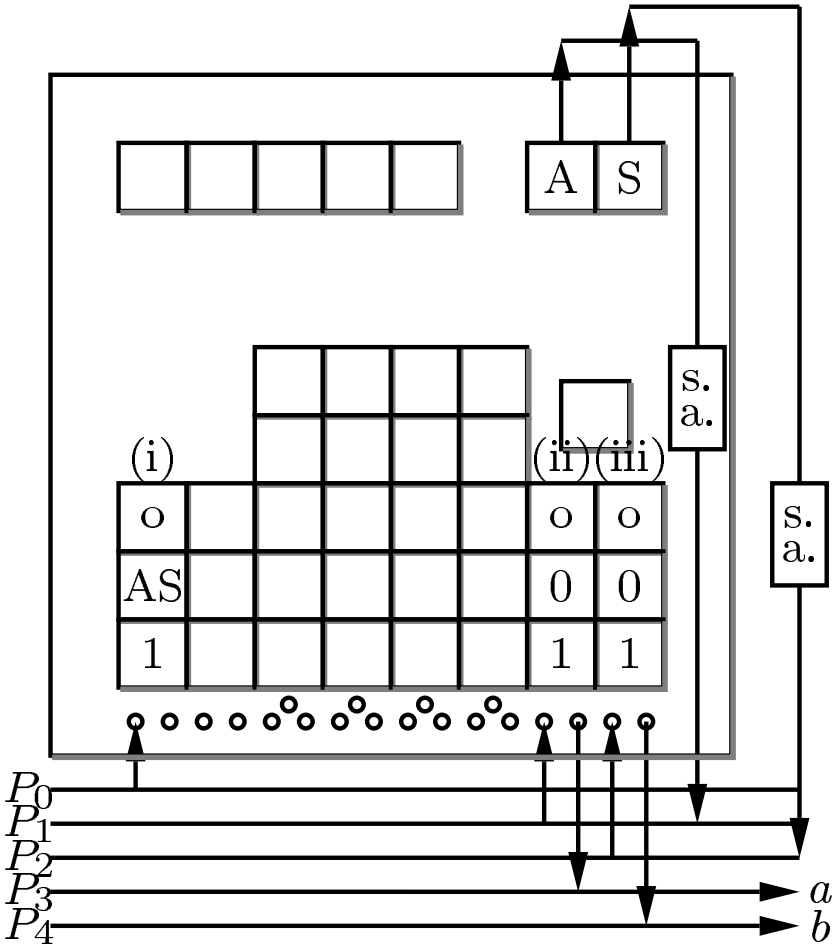
Une technique de programmation: le conditionel...

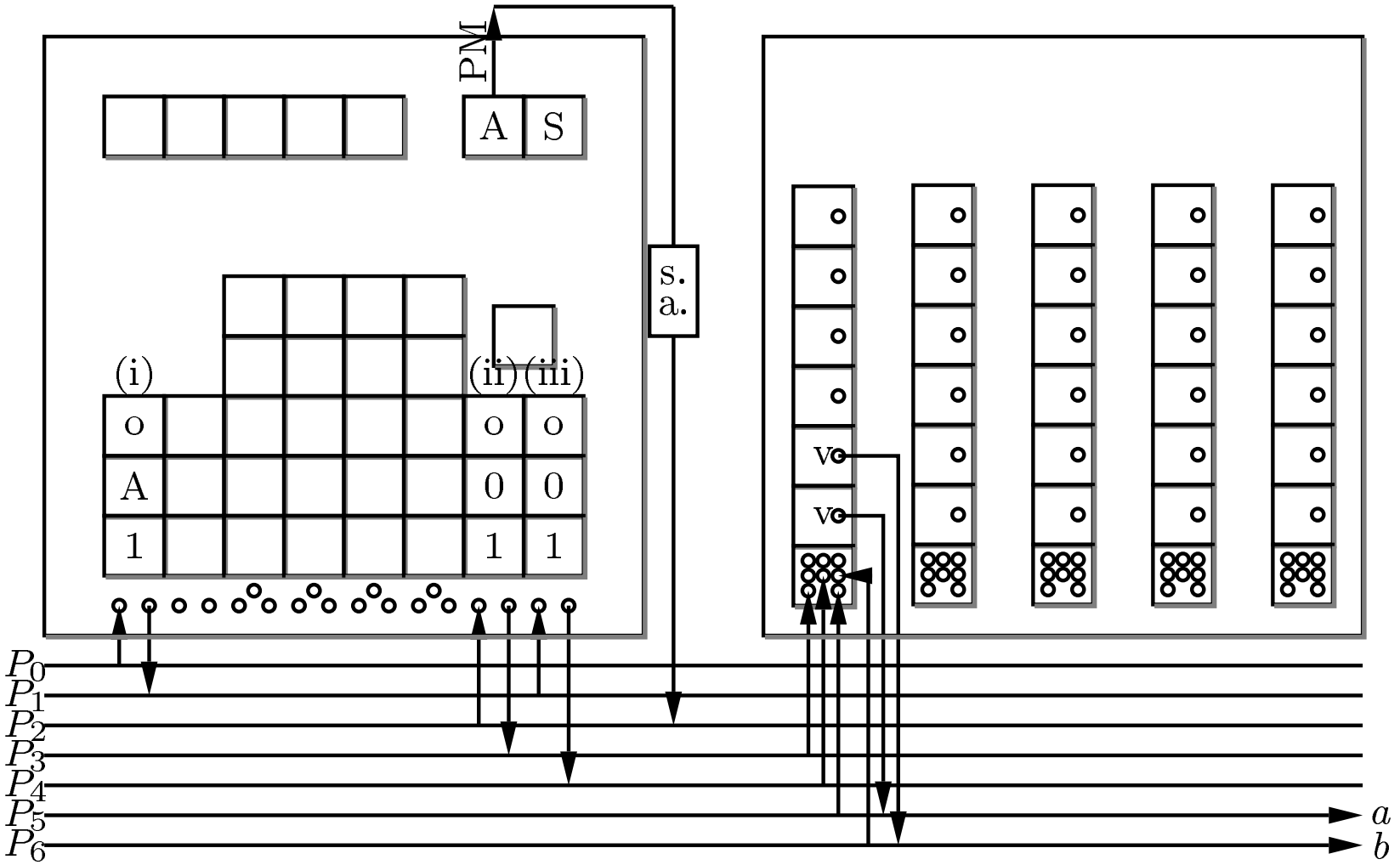
- L'exploitation des réprésenations physiques des signes (signe P c'est 9 pouls, signe M c'est 0 pouls) et des chiffres (0 n'a aucun pouls)
- Adapteur spécial pour la transformations des pouls de chiffre aux pouls de programme vers le "program pulse input terminal" d'un contrôl "dummy"

Deux méthodes pricipaux

- 'IF' avec deux canaux de sortie
- 'IF' avec un canal de sortie et un stepeur du "master programmer"

$\Rightarrow$ Un exemple de l'interchangéabilité entre nombres et programmes dans l'ENIAC (programmation au niveau/avec des "data")


A
S
s.
a.
(i)
(ii)
(iii)
o
o
o
AS
0
0
1
1
1
s.
a.
P_0
P_1
P_2
P_3
P_4
a
b


PM
A
S
s.
a.
(i)
(ii)
(iii)
o
A
1
0
0
1
1
vo
vo
P_0
P_1
P_2
P_3
P_4
P_5
P_6
a
b

Introduire la machine...

Une évaluation.

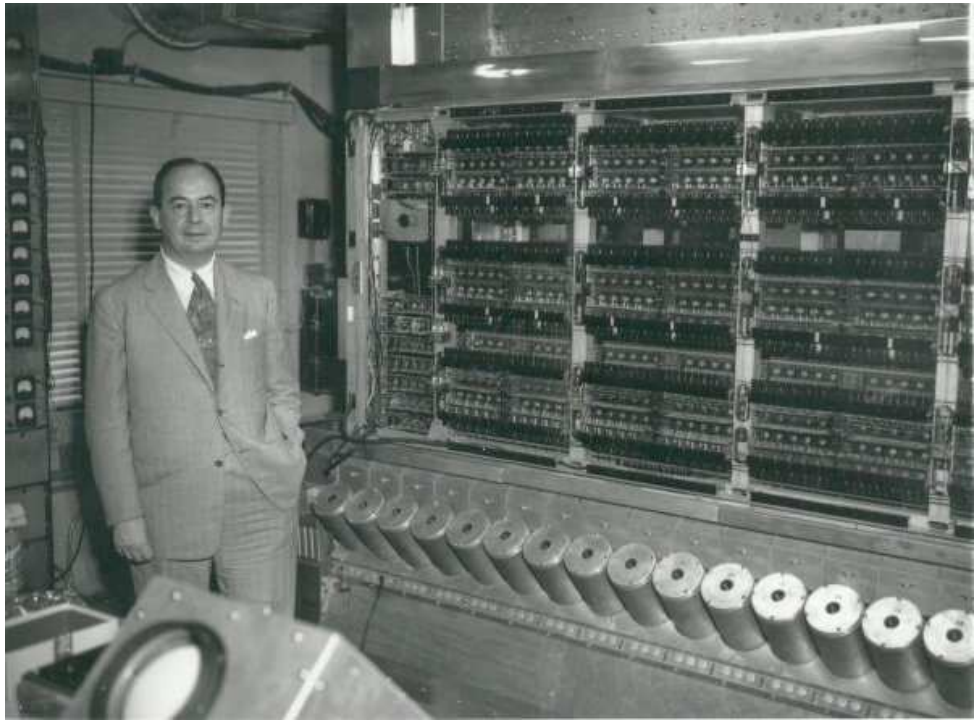
+ Machine extrèmement **rapide** en contraste avec les autres machines de ces temps: la vitesse était impressionante pour tous.

+ Machine programmable et “general-purpose”

- Méthode local de programmation – pas de “code”; programmer c’est câblé: “The ENIAC was a son-of-a-bitch to program” (Adèle Goldstine)

$\Rightarrow$ Requis une reflexion profonde sur “programmation”....

La notion de programme et de programmation “von Neumannien”



La notion de programme et de programmation "von Neumannien"

Le défi: Contrôler la machine à grand vitesse

- **Contrôle interiorisé – Le programme enrégistré comme un problème d'ingénierie**: "It is evident that **in order to be efficient**, you will have to treat the problem of logical instructions on the same level on which you treat other memory problems. In a complicated process, you must not only remember some intermediate result [...] but you must also remember what it is you are supposed to do." (von Neumann, 1946)
- **Contrôle du "flux"** "[C]ontemplate the prospect of locking twenty people for two years during which they would be steadily performing computations. And you must give them such explicit instructions at the time of incarceration that at the end of two years you could return and obtain the correct result for your lengthy problem! This dramatizes the necessity for high planning, foresight, and consideration of the logical nature of computation. **This integration of logic in the problem is a consequence of the high speed.** " (Von Neumann, 1948)

$\Rightarrow$ Les réponses von Neumannien:

- Câblage permanent de l'ENIAC comme machine sérielle qui utilise un code, le "order code" – commencement de programme-comme-texte
- EDVAC design – machine qui est construit autour d'un "order code"
- **The planning and coding of problems**

La notion de programme et de programmation "von Neumannien"

Les diagrammes de flux

- "It is [...] advisable to plan **first** the course of the process and the relationship of its successive stages to their changing codes, and to **extract from this the original coded sequence as a secondary operation** [...] We [...] propose to begin the planning of a codes sequence by laying out a schematic of the course of C through that sequence, i.e. through the required region of the selectron memory. This schematic is the *flow diagram* of C."

$\Rightarrow$ **Séparation des opérations logiques et arithmétiques visuel et structurel – boîtes alternatives et d'opération** "[Arithmetical operations and transfers of numbers] represent the proper mathematical (as distinguished from the logical] activities of the machine, and they should be shown as such. For this reason we will denote each area in which a coherent group of such operations takes place by a special box."

$\Rightarrow$ **"Changements de diagramme" (des sousroutines)**: "While C moves along the flow diagram , the contents of the alternative boxes [...] and of the operation boxes [...] will keep changing [...] This whole modus procendi assumes, however, that hte flow diagram itself [...] remains unchanged. This, however, is not unqualifiedly true.[...] $\Rightarrow$ the variable remote connection: situation de "jump" avec changement d'addresse

$\Rightarrow$ **..."info additonel"** substitution boxes (changement d'état); table de stockage; assertion boxes

$\Rightarrow$ Diagramme comme moyen de differentier entre la structure logique (le dynamique d'un programme) et les opérations arithmétique (l'aspect statique d'un programme)

La notion de programme et de programmation "von Neumannien"

Les étapes de codage – division du travail

Etape 1 L'étape mathématique de préparation: analyse, algothmization et estime de précision "It should be noted that the first step has **nothing to do with computing or with machines**. [T]he second step has, at least, nothing to do with mechanization: it would be equally necessary if the problems were to be computed "by hand"."

Etape 2 Préparation du diagramme de flux: l'étape 'dynamique' ou 'macroscopique', c.-à-d. le contrôle logique "the drawing of the flow diagram present little difficulty. Every mathematician, or every **moderately mathematically trained** person should be able to do this in a routine manner. [Errors or ommissions] should not be frequent, and will in most cases signalize themselves by some inner maladjustment of the diagram, which becomes obvious before the diagram is completed."

Etape 3 "[C]onsists of the individual coding of every operation box, alternative box and variable remote connection. This is the *static* or *microscopic* stage of coding [...] We feel certain that a **moderate amount of experience** with this stage of coding suffices to remove from it all difficulties, and to make it a perfectly routine operation"

Stage 4 Le numration des positions de stockage et des ordres: très proche à la machine: "In all the coding that we are going to do, references to specific memory locations will occur. These locations are $2^{12} = 4096$ in number, and we always enumerate them with the help of a 12 binary digit number or aggregate"

La notion de programme et de programmation “von Neumannien”

Order Code

Table 1: Table of basic orders [**?**]

Nr	Symb.	Description
1	x	Clear A and add number located at x into it
2.	x-	Clear A and subtract number at position x into it
3.	x M	Clear A and add the absolute value of the number located at x into it
4.	x -M	Clear A and subtract the absolute value of the number at position x into it
5.	x h	Add number located at x into the A
6.	x h-	Subtract number at position x into the A
7.	x hM	Add the absolute value of the number located at x into the A
8.	x h-M	Subtract the absolute value of the number at position x into the A
9.	xR	Clear R and add number located at position x into it
10.	A	Clear the A and shift the number held in the register into it

Continued on next page

Table 1 – continued from previous page		
Nr	**Symbol**	**Description**
11.	x X	Clear the A and multiply the number located at position x by the number in the register
12.	x ÷	Clear register and divide the number in the A by the number located at x leaving the remainder in A and the quotient in R
13.	x C	Shift the control to the left of the order pair[a] 1 at position x
14.	x C’	Shift the control to the right of the order pair at position x
15.	x Cc	If the number in A ≥ 0 shift the control as in x C
16.	x Cc	If the number in A ≥ 0 shift the control as in x C’
17.	x S	Transfer the number in A to position x

Continued on next page

Table 1 – continued from previous page		
Nr	**Symbol**	**Description**
18.	x Sp	Replace the left-hand 12 digits of the left-hand order located at position x by the 12 digits 9 to 20 in A
19.	x Sp’	Replace the left-hand 12 digits of the right-hand order located at position x by the 12 digits 29 to 40 in A
20.	R	Replace the content $\epsilon_0\epsilon_1\epsilon_2\ldots\epsilon_{39}$ of A by $\epsilon_0\epsilon_0\epsilon_1\epsilon_2\ldots\epsilon_{39}$
21.	L	Replace the content $\epsilon_0\epsilon_1\epsilon_2\ldots\epsilon_{39}$ and $\eta_0\eta_1\eta_2\ldots\eta_{39}$ of A and R by $\epsilon_0\epsilon_2\epsilon_3\ldots\epsilon_{39}0$ and $\eta_1\eta_2\eta_3\ldots\eta_{39}\epsilon_1$

[a] In volume 1, orders are ordered in pairs in the memory because orders are “less than half as long as a forty binary digit number, and hence the orders are stored in the Selectron memory in pairs.”

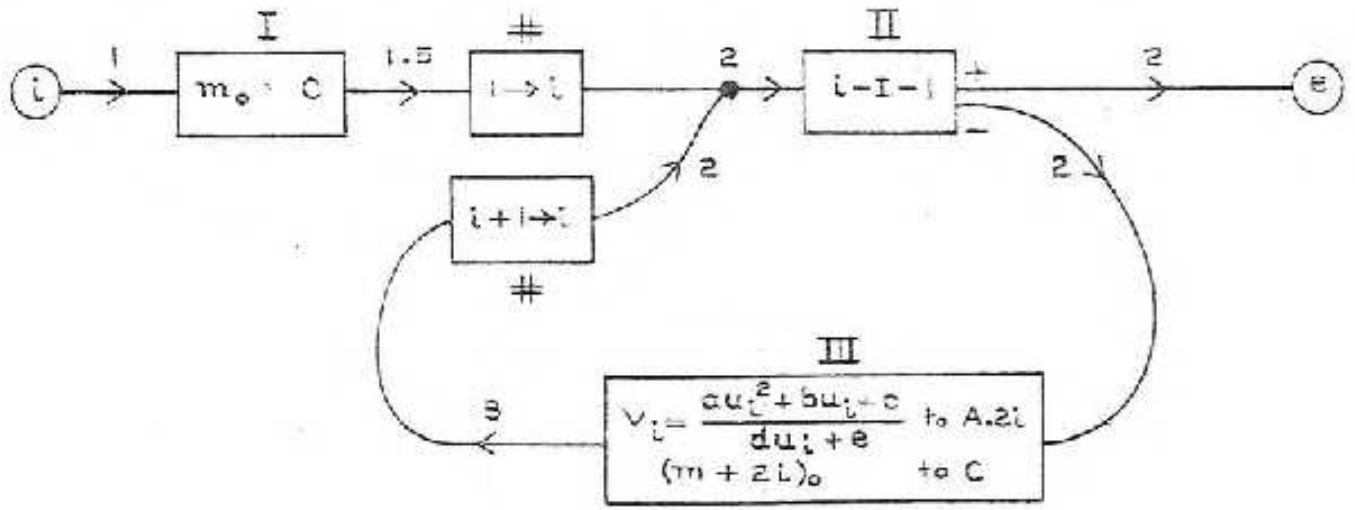
La notion de programme et de programmation “von Neumannien”

Premier exemple qui requis un diagramme – conditionel et boucle

Le problème: pour $u1, ..., u_i$ calcule $v_1, ..., v_I$ avec $v_j = \frac{a^2 u_j + b u_j + c}{d u_j + e}$ et chaque pair u_j, v_j stock dans deux positions avoisinant $m + 2j - 2, m + 2j - 1$ et m et I stocqué dans certains positions de mémoire.

Structuration du programme en trois étapes (I à III) et structuration du mémoire en trois partie, A, B, C (A pour u_i et v_i; B pour les constants, I ($(m + 2I)_0$) et m_0; C pour le variable d’induction i ($m + 2i$)

Calcul direct des positions, p.e. le “variable d’induction i” est en fait le calcul de $m + 2i - 2$, la position de u_i.



a, b, c, d, e
m, I
i

	1	1.5	2	3
A.2i'	—	—	— for $i' \geq i$ $v_{i'}$ for $i' < i$	— for $i' > i$ $v_{i'}$ for $i' \leq i$
C	—	m_o	$(m+2i-2)_o$	$(m+2i)_o$

A.2i'−1	$u_{i'}$
B.1	a
2	b
3	c
4	d
5	e
6	m_o
7	$(m+2I)_o$

La notion de programme et de programmation “von Neumannien”

Premier exemple qui requis un diagramme – conditionel et boucle

1,1	B.6			Ac	m_o
2	C	S		C	m_o

The coding of II is also quite simple, we must only observe that $(m + 2i-2)_o$ - $(m + 2I)_o$ has the same sign as i-I-1. The code follows:

11,1	C			Ac	$(m + 2i-2)_o$
2	B.7	h-		Ac	$(m + 2i-2)_o$ $(m + 2I)_o$
3	e	Cc			

La notion de programme et de programmation “von Neumannien”

Premier exemple qui requis un diagramme – conditionel et boucle

111,1	C		Ac	$(m + 2i-2)_0$
2	111,3	Sp	111,3	$(m + 2i-2)_0$
3	-			
[	m + 2i-2	]	Ac	u_i
4	s.1	S	s.1	u_i
5	C		Ac	$(m + 2i-2)_0$
6	B.8	h	Ac	$(m + 2i-1)_0$
7	111,22	Sp	111,22	m + 2i-1 S
8	B.8	h	Ac	$(m + 2i)_0$
9	C	S	C	$(m + 2i)_0$
10	B.4	R	R	d
11	s.1	x	Ac	du_i
12	B.5	h	Ac	$du_i + e$
13	s.2	S	s.2	$du_i + e$
14	B.1	R	R	a
15	s.1	x	Ac	au_i
16	B.2	h	Ac	$au_i + b$
17	s.3	S	s.3	$au_i + b$
18	s.3	R	R	$au_i + b$
19	s.1	x	Ac	$au_i^2 + bu_i$
20	B.3	h	Ac	$au_i^2 + bu_i + c$
21	s.2	÷	R	$v_i = \frac{au_i^2 + bu_i + c}{du_i + e}$

La notion de programme et de programmation "von Neumannien"
Diagrammes de flux comme "langage" de programmation?

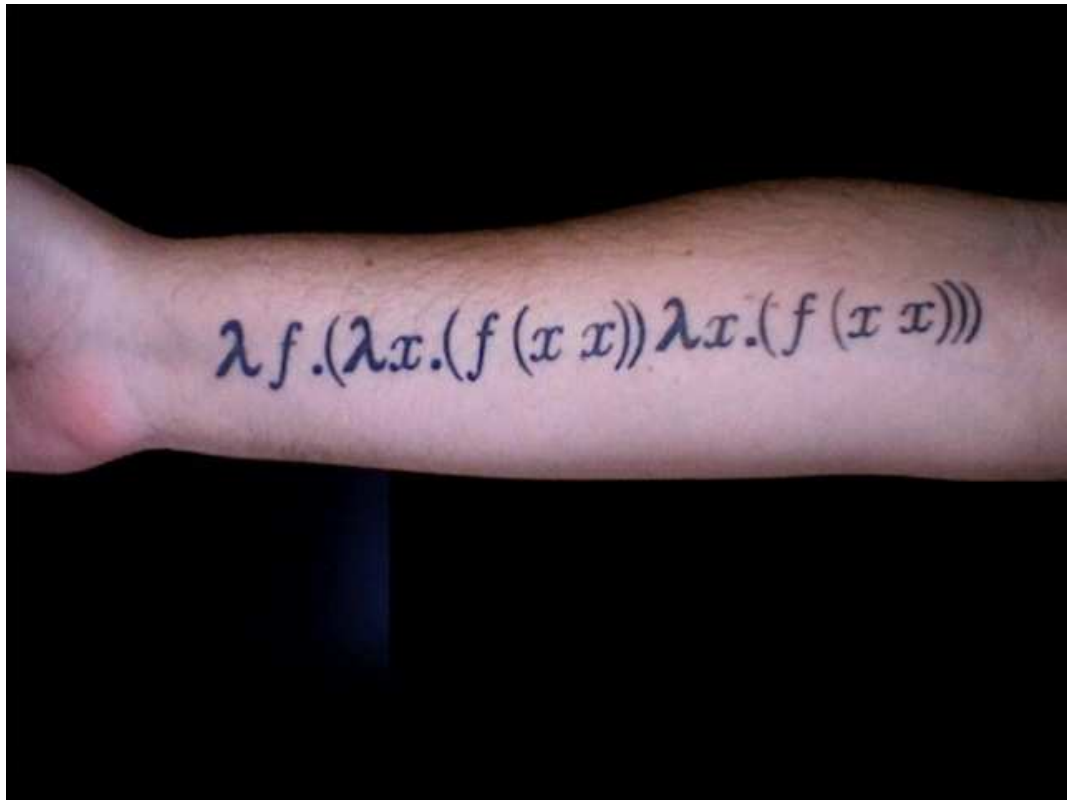
- Ambition, c'est le contrôle du flux, après il nous reste le codage non-problématique
- Division de travail refleter dans division entre "programme-comme-diagramme" – l'aspect dynamique et macro/"planning" – et programme-comme-code – l'aspect statique et micro/"coding"

⇒ Mise au point des boucles itératifs et les conditionels comme contrôle du flux et non pas un système de sousprogrammes:"Any further adjustments which are necessitated by the relationships of the subroutines to each other [...]are made **by typing directly into the memory** [...] References to another subroutine [...] are likely to be rare and irregularly distributed. They are therefore less well suited to automatic treatment [...] than to **ad hoc, manual treatment**, by direct typing into the machine" (VN&G, 1946-48)

⇒ **Sousestimation du problème** "**the problem of coding routines need not and should not be a dominant difficulty** [In] fact we have made a careful analysis of this question and we have concluded from it that the problem of coding can be dealt with in a very satisfactory way." (von Neumann and Goldstine, 1946-48)

⇒ Division arificielle qui bloque des avances fondamentaux dans la théorie de programmation qui requis de connecter ces deux niveaux: Un "formalisme" qui est extrèment dure à utiliser et nécessité de spécifier plein de choses pour le codeur au niveau macro – il ne peut pas les séparer!

La notion de programme et de programmation chez Curry



La notion de programme et de programmation chez Curry

Programmer l'ENIAC sans code (1)

- En collaboration avec Willa Wyatt, une des programmeuses de l'ENIAC, Curry écrivait un rapport "A study of inverse interpolation of the Eniac" (1946, déclassé en 1999)


- "The problem of inverse interpolation [...] is important in the calculation of firing tables. Suppose the trajectory calculations have given us the coordinates (x, y) of the projectile as functions of t (time) and ϕ (angle of departure). For the tables we want t and ϕ as functions of x and y; indeed we wish to determine ϕ so as to hit a target whose position (x, y) is known, and t is needed for the fuze (mèche) setting or other purposes. [...] In this report the problem of inverse interpolation is studied with reference to the programming on the ENIAC as a problem in its own right."

La notion de programme et de programmation chez Curry

Programmer l'ENIAC sans code (2) – ré-utilisé les diagrammes de câblage

- **Logique des étapes** "The stages can be programmed as independent units, with a uniform notation as to program lines, and then put together; and since each stage uses only a relatively small amount of the equipment the programming can be done on sheets of paper of ordinary size."
- **Adaptabilité** *"[The] basic scheme was not designed specifically for a particular problem, but as a basis from which modifications could be made for various such problems."*
- Origine d'une théorie de composition des programmes (sous-programmes)
  - "**The problem of program composition was a major consideration in a study of inverse interpolation on the ENIAC** [...]; for although that study was made under stress and was directed primarily towards finding at least one practical method of programming a specific problem, yet an effort was made to **construct the program by piecing together subprograms in such a way that modifications could be introduced by changing these subprograms.**" (Curry, 1950)
  - "This problem is almost ideal for the study of programming; because, **although it is simple enough to be examined in detail by hand methods; yet it is complex enough to contain a variety of kinds of program compositions**." (Curry 1952)

La notion de programme et de programmation chez Curry

Goulot d'étranglement de la programmation et la programmation automatique

- **Le goulot d'étranglement de la programmation**:"In the present state of development of automatic digital computing machinery, **a principal bottleneck is the planning of the computation** [...] Ways of shortening this process, and of systemizing it, so that a part of the work can be done by the technically trained personnel **or by machines**, are much desired. The present report is an attack on this problem from the standpoint of **composition of computing schedules**"

- **La crique sur VN comme trop "hands-on"** "But one comment seems to be in order in regard to [GvN's] arrangement. The scheme allows certain data to be inserted directly into the machine by means of a typewriter-like device. Such an arrangement is very desirable for trouble-shooting and computations of a tentative sort, but for final computations of major importance **it would seem preferable to proceed entirely from a program or programs recorded in permanent form, not subject to erasure, such that the computation can be repeated automatically** [...] on the basis of the record."

- "It is said that during the war an error in one of the firing tables was caused by using the wrong lead screw in the differential analyser. **Such an error would have been impossible if the calculation had been completely programmed**." (Curry, 1950)

La notion de programme et de programmation chez Curry

Une notation pour la composition des programmes

- Composition de X et Y: $X \to Y$
- Affectation: $\{\eta : \lambda\}$ (stockage de η dans λ)
- Conditionel: $X \to Y \& Z$
- Teste de Prédicat: $\{\Phi\}$

Composition des programmes, quelques exemples

- Un programme arithméthique:

$$\{x : A\} \to \{x + 1 : A\} \to \{A : w\} \to \{y : A\} \to \{A + 1 : A\} \to \{A : R\}$$
$$\to \{Rw : A\} \to \{A : \lambda\}$$

- Un boucle: $\{A : i\} \to \{A - m : A\} \to \{A < 0\} \to (\{i : A\} \to \{A + 1 : A\} \to \{Kc\}) \,\&\, O_z$
  "In order to carry out iterative calculations, it is necessary to have programs which double back on themselves, repeating certain parts as prescribed a certain number of times, or until certain conditions have been fulfilled"
- Programme complexe: $X \to (Y \to (Z \to O_1 \& X \& Z))$ (la composition de trois programme X, Y et Z avec trois sortie, y inclus deux boucles

$\Rightarrow$ *Une notation qui reflète la "formation" des programmes automatisable*

La notion de programme et de programmation chez Curry

Deux étapes: analyse et synthèse: Etape 1 (1)

Table 2: Table des programmes élementaires de Curry

Number for i =				**Symbol**	**effects**			**GvN for $i =$**			
0	**1**	**2**	**3**		A	R	X	**0**	**1**	**2**	**3**
1				$\{0 : A\}$	0	-	-	a			
2	3			$\{\pi_i(1) : A\}$	$\pi_i(1)$	-	-	a	a		
	4	5	6	$\{\pi_i(A) : A\}$	$\pi_i(A)$	-	-		a	a	a
7	8	9	10	$\pi_i(R) : A\}$	$\pi_i(R)$	R	-	A	a	a	a
11	12	13	14	$\{\pi_i(x) : A\}$	$\pi_i(x)$	-	x	x	x-	xM	x-M
15				$\{d(*)\} : A\}$	$d(*)$		x	a			
16	17			$\{A + \pi_i(1) : A\}$	$A + \pi_i(1)$	-	-	a	a		
18	19	20	21	$\{A + \pi_i(R) : A\}$	$A + \pi_i(R)$	R	-	a	a	a	a
22	23	24	25	$\{A + \pi_i(x) : A\}$	$A + \pi_i(x)$	-	x	h	-h	Mh	-Mh
26				$\{A + d(*) : A\}$	$A + d(*)$	-	x	a			
27				$\{r\}$	$r(A)$		-	R			
28				$\{l\}$	$l(A)$		-	L			
29				$\{xR : A\}$	b	b	x	X			

Table 2 – continued from previous page											
Number for i =				**Symbol**	**effects**			**GvN for $i =$**			
0	**1**	**2**	**3**		A	R	X	**0**	**1**	**2**	**3**
30				$\{A : R\}$	A	A	-	a			
31				$\{x : R\}$	A	-	A	xR			
32				$\{A/x : R\}$	A	A	x	$\div$			
33				$\{A : x\}$	A	-	A	S			
34				$\{A : d(*)\}$	A	-	c	Sp			
35				$\{A : e(*)\}$	A	-	c	a			
36				$\{K\}$	-	-	-	C			
37				$\{A < 0\}$	-	-	-	Cc			
38				stop	-	-	-	a			

La notion de programme et de programmation chez Curry

Deux étapes: analyse et synthèse: Etape 1 (2)

Etape 1 L'analyse des programmes complexes en programmes élementaires (basic programs) (cfr VN's stage 2 and 3).

Utilisation des réductions aux "commandes élémentaires" (basic orders)

$$
\begin{aligned}
(\alpha_1) \quad & \{\zeta : \lambda\} = \{\zeta : A\} \to \{A : \lambda\} \\
(\alpha_2) \quad & \{\zeta : \lambda\} = \{\zeta : R\} \to \{R : \lambda\} \\
(\beta_1) \quad & \{\phi(\xi) : A\} = \{\xi : A\} \to \{\phi(A) : A\} \\
(\beta_2) \quad & \{\phi(\xi) : R\} = \{\xi : R\} \to \{\phi(R) : R\}
\end{aligned}
$$

Aussi pour des programmes plus complexes, p.e., conditionels:

$$
\begin{aligned}
\{\xi < 0\} = \quad & \{\xi : A\} \to \{A < 0\} \\
\{\xi > 0\} = \quad & \{-\xi < 0\} \\
\{\xi < \eta\} = \quad & \{\xi - \eta < 0\} \\
\{\xi > \eta 0\} = \quad & \{\eta - \xi < 0\} \\
\{\Phi_1 \wedge \Phi_2\} = \quad & \{\Phi_1\} \to (\{\Phi_2\} \to O_1 \,\&\, O_2) \,\&\, O_2 \\
\{\Phi_1 \vee \Phi_2\} = \quad & \{\Phi_1\} \to O_1 \,\&\, (\{\Phi_2\} \to O_1 \,\&\, O_2) \\
\{\sim \Phi\} = \quad & \{\Phi\} \to (\{Kc\} \to O_2) \,\&\, O_1
\end{aligned}
$$

La notion de programme et de programmation chez Curry

Deux étapes: analyse et synthèse: Etape 1 (3)

$$\zeta = (x+1)(y+1)$$

$$\xi_1 = x+1 \quad \eta_1 = y+1 \quad \zeta = \xi_1\eta_1$$

$$\{\zeta : \lambda\} = \{\zeta : A\} \to \{A : \lambda\}$$

$$\{\zeta : A\} = \{\xi_1 : A\} \to \{A\eta_1 : A\}$$

$$\{\xi_1 : A\} = \{x : A\} \to \{x+1 : A\}$$

$$\{A\eta_1 : A\} = \{A : w\} \to \{w\eta_1 : A\}$$

$$\{w\eta_1 : A\} = \{y : A\} \to \{A+1 : A\} \to \{A : R\} \to \{Rw : A\}$$

La combinaison de ces compositions donne:

$$\{\zeta : \lambda\} = \{x : A\} \to \{x+1 : A\} \to \{A : w\} \to \{y : A\} \to \{A+1 : A\} \to \{A : R\}$$
$$\to \{Rw : A\} \to \{A : \lambda\}$$

La notion de programme et de programmation chez Curry

Deux étapes: analyse et synthèse: Etape 2

Etape 2 La composition actuelle; cfr étape 4 de VN:

From top to bottom: The $T_1(X)$ transformation; the $T_2(Y)$ transformation; and finally the substitution $[\frac{\Theta T_1}{[T_2](Y)}](X)$ that substitutes Y in X at position m.

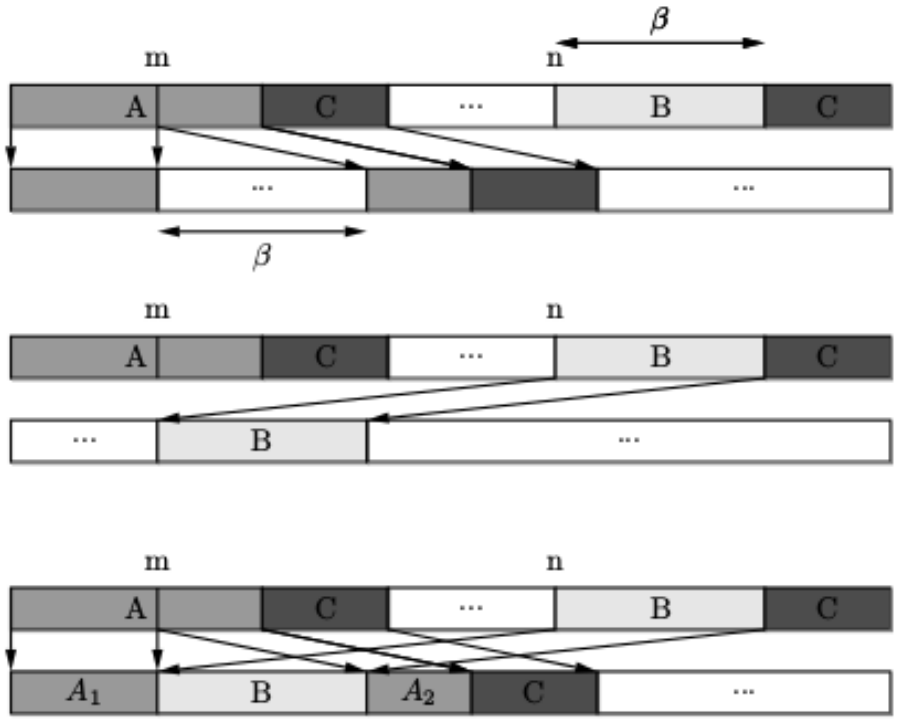


Figure 1: From top to bottom: The $T_1(X)$ transformation; the $T_2(Y)$ transformation; and finally the substitution $[\frac{\Theta T_1}{[T_2](Y)}](X)$ that substitutes Y in X at position m.

La notion de programme et de programmation chez Curry

Méchanization et simplification

> "Von Neumann and Goldstine have pointed out that [we] should not use the technique of program composition to make the simpler sorts of programs, – these would be programmed directly –, but only to avoid repetitions in forming programs of some complexity. Nevertheless, there are three reasons for pushing the technique clear back to formation of the simplest possible programs from the basic programs, viz.: (1) Experience in logic and in mathematics shows that an **insight into principles** is often best obtained by a consideration of cases too simple for practical use [...] (2) It is quite possible that the technique of program composition can **completely replace the elaborate methods** of Goldstine and von Neumann [...] (3) The technique of program composition **can be mechanized**; if it should prove desirable to set up programs [...] by machinery, presumably this may be done by analyzing them clear down to the basic programs" (Curry, 1952)

$\Rightarrow$ Motivé par une idée fondamental de la logique mais appliqué et adapté aux machines...

La notion de programme et de programmation chez Curry

La notation de composition comme langage de programmation?

- Définition de programme chez Curry: "An assignment of $n+1$ words to the first $n+1$ locations will be called a program." $X = M_0 M_1 ... M_n$
- Ambition, c'est de rendre plus facile le codage en créant une notation de programme qui permet aussi l'automatisation
- La notation: reflète la théorie et la méthode de la composition des programmes; elle connecte (et ne divise pas) les étapes 2 et 3: "The present theory develops, in fact, a notation for program construction which is **more compact** than the "flow charts" of [von Neumann and Goldstine]. Flowcharts will be used [...] primarily **as an expository device**. By means of this notation a composite program can be exhibited as a function of its components in such a way that **the actual formation of the composite program can be carried out by a suitable machine**"

$\Rightarrow$ Notation ("langage") comme médiateur entre les différents étapes de programmation propre (c'est la notation qui developpe l'analyse en programmes élémentaires)

$\Rightarrow$ Méthode plus général et plus indépendant de GVN

Discussion

- Origin de “code”/”programme” en trois étapes:
  - la machine programmable et à grand vitesse sans code inspire
  - contrôle de la vitesse: une approche de division de travail refleté dans une division entre programme-comme-diagramme et programme-comme-code
  - “programmer”: une approche d’automatisation avec une notation qui reflète le structure d’un programme comme composition de sous-programmes: le “code source” c’est ce que doit être analyser et synthétiser
- La “nouveleté radicale” de l’ordinateur, ce n’est pas le programme enrégistré mais sa vitesse – elle demande une réflexion profonde sur la programmation dont le programme enrégistré est qu’un étape
- Programme-sans-texte, oui; programme-sans-machine, non: le programme-comme-texte contient ainsi toujours les traces d’un histoire d’une médiation entre humain et machine
- Le vrais défi: “computing science is and will always be concerned with the interplay between mechanized and human symbol manipulation, usually referred to as ”computing” and ”programming” respectively.” (Dijkstra, 1988) C’est Curry qui l’a compris et non pas VN.