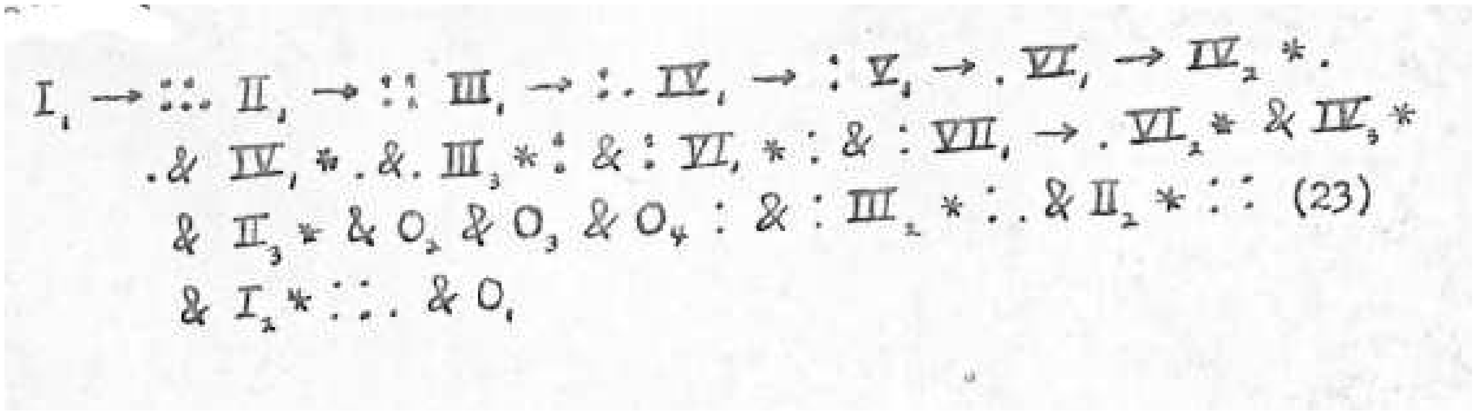
Curry's study of inverse interpolation on the ENIAC
From a concrete problem to the problem of program composition


$$\begin{aligned}
&I_1 \rightarrow \therefore.\ II_1 \rightarrow ::\ III_1 \rightarrow :.\ IV_1 \rightarrow :\ V_1 \rightarrow .\ VI_1 \rightarrow IV_2 *. \\
&.\&\ IV_1 * .\&.\ III_3 * :\ \&:\ VI_1 * : \&:\ VII_1 \rightarrow .\ VI_2 * \&\ IV_3 * \\
&\&\ II_3 * \&\ O_2\ \&\ O_3\ \&\ O_4\ :\ \&:\ III_2 * :.\ \&\ II_2 * ::\quad (23) \\
&\&\ I_2 * \therefore.\ \&\ O_1
\end{aligned}$$

M. Bullynck[1] and L. De Mol[2]

[1] Paris 8, maarten.bullynck@kuttaka.org

[2]Universiteit Gent, elizabeth.demol@ugent.be

Introduction

- How a logician got involved with computers...
- 1946: "A study of inverse interpolation of the Eniac"
- 1949: "On the composition of programs for automatic computing"
- 1950: "A program composition technique as applied to inverse interpolation"
- Discussion

How a logician got involved with computers...

How a logician got involved with computers...

Highlights in Curry's career before 1945

- 1924: Starts PhD on differential equations and switches to PhD in logic
- 1926–1927: Reads Russell and Whitehead's *Principia Mathematica* and starts developing his theory of combinators
- 1927–1928: Discovers Schönfinkel's paper "Über die Bausteine der mathematischen Logik" (1924)
- 1929: PhD *Grundlagen der kombinatorischen Logik*, supervised by Hilbert (actually Bernays)
- September 1929: Appointed at the State College, Pennsylvania
- 1930ies: Publications on combinatory logic
- During World War II, research in applied mathematics (Heaviside operational calculus)
- 1942–1944: work at Frankford Arsenal and Applied Physics Laboratories
  - 1944: moves to Aberdeen Proving Ground → ENIAC

How a logician got involved with computers...
Computability, combinators, λ and Curry

- Proposal different models of computability, a confluence of ideas in 1936: Church, Turing and Post
- Before 1936: development of Church's set of postulates and ultimately λ-calculus
- The power of $\lambda \sim$ combinators: Curry knows about theory of **computability**
- "[I]t is evident that one can formalize in various ways and that some of these ways constitute a more profound analysis than others. Although from some points of view one way of formalization is as good as any other, yet a certain interest attaches to the problem of simplification [...] In fact we are concerned with constructing systems of an extremely rudimentary character, which analyze processes ordinarily taken for granted." [Curry, 1942]

How a logician got involved with computers...

- The Ballistic Research Laboratories (Aberdeen Proving Ground) had "assembled a 'Computations Committee' to prepare for utilizing the machine after its completion", and the ENIAC was extensively test-run during its first months.
- The members:
  - Leland B. Cunningham (an astronomer)
  - **Haskell B. Curry (a logician)**
  - Derrick H. Lehmer (a number theorist)

How a logician got involved with computers...

Three reports

- 1946: "A study of inverse interpolation of the Eniac"
- 1949: On the composition of programs for automatic computing
- 1950: A program composition technique as applied to inverse interpolation
- (1954: "The logic of program composition", presented at 2e Colloque International de Logique Mathématique, Paris, 25-30 août 1952)

1946: "A study of inverse interpolation of the Eniac"

1946:"A study of inverse interpolation of the Eniac"

- In collaboration with Willa Wyatt one of ENIAC's programmers

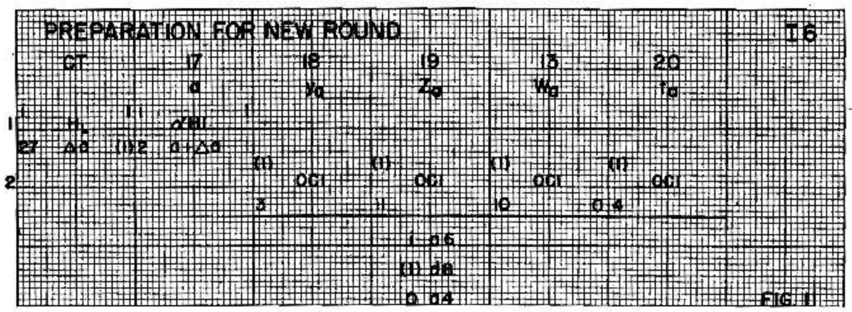
- declassified in 1999
- Focus on theoretical aspects of programming besides details of the wiring

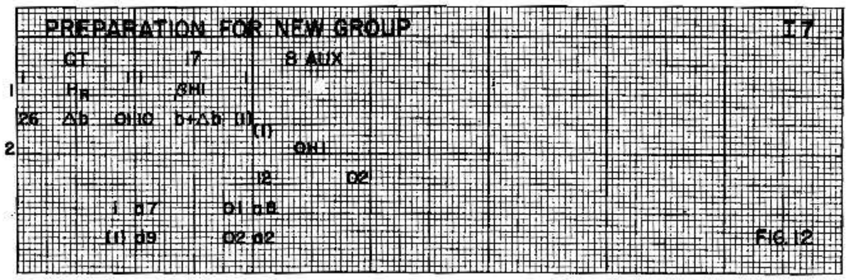
1946:"A study of inverse interpolation of the Eniac"
The problem of Inverse Interpolation

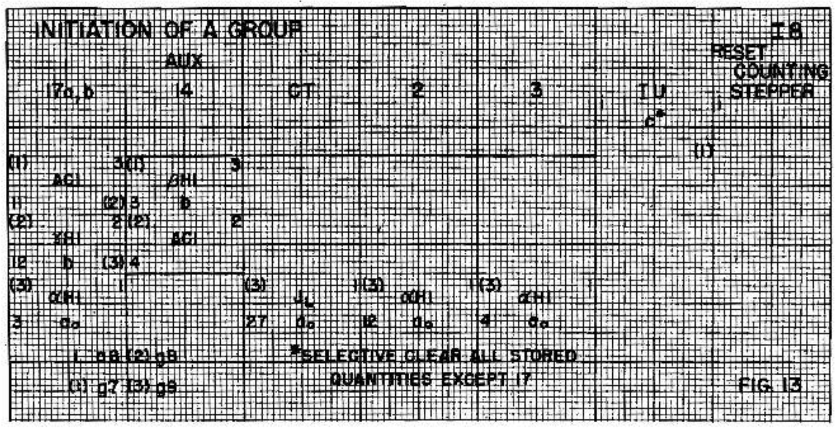
- "The problem of inverse interpolation may be stated as follows. Suppose we have a table giving values of a function $x(t)$ [...] for equally spaced values of the argument t. It is required to tabulate t for equally spaced values of x."
- "This problem is important in the calculation of firing tables. 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 setting or other purposes. This is a problem of inverse interpolation in two variables; it can be solved by two successive inverse interpolations on one variable."
- "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."

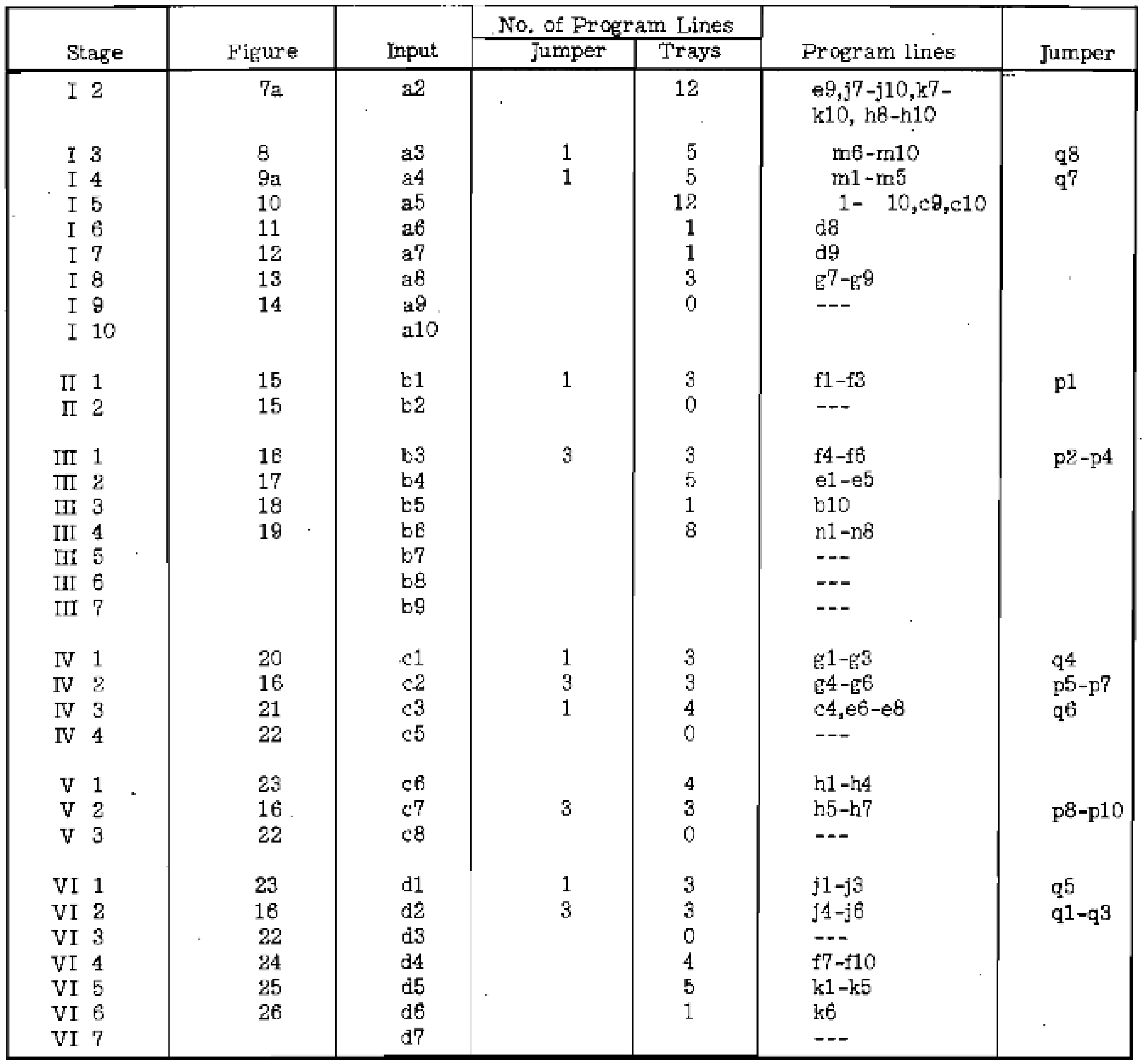
1946:"A study of inverse interpolation of the Eniac"
Stages and processes

- "The entire computation procedure can [...] be divided into certain major parts which are repeated over and over according to the programming. These major parts will be called **processes**."
- "Each process is broken into pieces called **stages** which are units in the following sense. Each stage is a program sequence with an input and one or more outputs. The input of each stage comes from the output of one or more other stages of the same or different processes; the ouputs all go to the input of some other stage or are blank [...]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."


PREPARATION FOR NEW ROUND
I6
CT
17
18
19
13
20
OCI
FIG 11


PREPARATION FOR NEW GROUP
I7
CT
17
8 AUX
FIG 12


INITIATION OF A GROUP
I8
AUX
17a, b
14
CT
2
3
TU
RESET COUNTING STEPPER
*SELECTIVE CLEAR ALL STORED QUANTITIES EXCEPT 17
FIG 13



Stage	Figure	Input	No. of Program Lines		Program lines	Jumper
			Jumper	Trays		
I 2	7a	a2		12	e9,j7-j10,k7-k10, h8-h10	
I 3	8	a3	1	5	m6-m10	q8
I 4	9a	a4	1	5	m1-m5	q7
I 5	10	a5		12	1- 10,c9,c10	
I 6	11	a6		1	d8	
I 7	12	a7		1	d9	
I 8	13	a8		3	g7-g9	
I 9	14	a9		0	---	
I 10		a10				
II 1	15	b1	1	3	f1-f3	p1
II 2	15	b2		0	---	
III 1	16	b3	3	3	f4-f6	p2-p4
III 2	17	b4		5	e1-e5	
III 3	18	b5		1	b10	
III 4	19	b6		8	n1-n8	
III 5		b7			---	
III 6		b8			---	
III 7		b9			---	
IV 1	20	c1	1	3	g1-g3	q4
IV 2	16	c2	3	3	g4-g6	p5-p7
IV 3	21	c3	1	4	c4,e6-e8	q6
IV 4	22	c5		0	---	
V 1	23	c6		4	h1-h4	
V 2	16	c7	3	3	h5-h7	p8-p10
V 3	22	c8		0	---	
VI 1	23	d1	1	3	j1-j3	q5
VI 2	16	d2	3	3	j4-j6	q1-q3
VI 3	22	d3		0	---	
VI 4	24	d4		4	f7-f10	
VI 5	25	d5		5	k1-k5	
VI 6	26	d6		1	k6	
VI 7		d7			---	

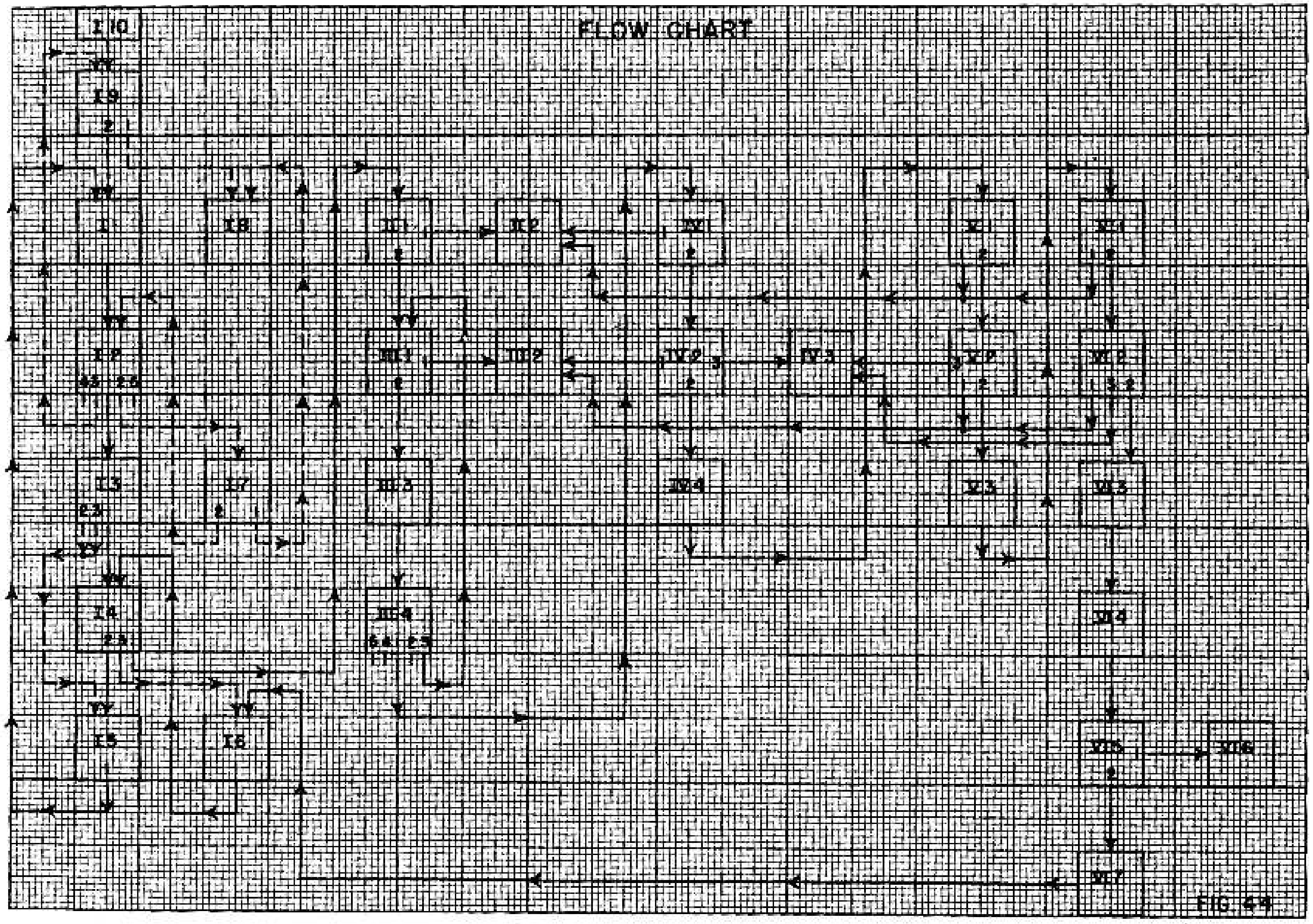

FLOW CHART
I 10
I 9
I 1
I 8
II 1
II 2
IV 1
V 1
VI 1
I 2
III 1
III 2
IV 2
IV 3
V 2
VI 2
I 3
I 7
III 3
IV 4
V 3
VI 3
I 4
III 4
VI 4
I 5
I 6
VI 5
VI 6
VI 7

FIG 44

1946:"A study of inverse interpolation of the Eniac"
Theoretical considerations in the 1946 report

- **Theorem 1** *Let the n accumulators $A_1, A_2, ..., A_n$ contain initially the n quantities $x_1, ..., x_n$ respectively. Then a necessary and sufficient condition, that $y_1, ..., y_n$ be such that there exist a series of transfers between the $A_1, ..., A_n$ without clearing which ends with $y_1, ..., y_n$ in $A_1, ..., A_n$ respectively is that there exist a matrix of integers $C = (C_{ij})$ such that:*

$$y_i = C_{i1}x_1 + C_{i2}x_2 + ... + C_{in}x_n \text{ and } i = 1, 2, ..., n \text{ and } \left|C_{ij}\right| = 1$$

- Basic scheme + modifications: "[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." Example: composite interpolation. "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)"

1946:"A study of inverse interpolation of the Eniac"
Theoretical considerations in the 1946 report (continued)

- "In this way we can build up more and more complicated programs. An examination has been made in this way of the programming of inverse interpolation on functions of two variables. 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) "

1949: On the composition of programs for automatic computing

1949: "On the composition of programs for automatic computing"

- **The problem of composition**: "In the present state of development of automatic digital computing machinery, a principal bottleneck is the planning of the computation...The present report is an attack on this problem from the standpoint of composition of computing schedules. By this is meant the following. Suppose that we wish to perform a computation which is a complex of simple processes that have already been planned. Suppose that for each of these component processes we have a plan recorded in the form of what is here called a program, by means of a system of symbolization called a code. It is required to form a program for the composite computation. This problem is here attacked theoretically by using techniques similar to those used in some phases of mathematical logic."
- **New notation and introduction of automated composition**"The present theory develops in fact a notation for program construction than the "flow charts" of [Goldstine and Von Neumann]. Flow charts 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.*"

1949: "On the composition of programs for automatic computing"
Content of the report

1949: "On the composition of programs for automatic computing"
Content of the report

IV Program composition

- **A** Simple substitution
- **B** Multiple substitution
- **C** Reduction to a single quantity program
- **D** Loop programs
- **E** Complex programs
- **F** Associativity properties

V Concluding Remarks

1949: “On the composition of programs for automatic computing”

Fundamental definitions and assumptions: words, programs and orders

Two types of Words: quantities and orders; “The distinction between quantities and orders is not a distinction of form. GIven a word written out in the code, it is impossible to tell from its appearance whether it is of type a or type b. [...]The machine this distinction according tot he situation. Making this classification of words in advance is a difficult problem” $\rightarrow$ theorem on type determination

“An assignment of $n+1$ words to the first $n+1$ locations will be called a program.” $X = M_0 M_1 ... M_n$

Order = datum number location, exit number location and operator

Classification of Orders: arithmetical, transfer, control, stop orders.

1949: "On the composition of programs for automatic computing"

Assumptions and restrictions "[T]he first stage in a study of programming is to impose restrictions on programs in order that the words in all the configurations of the resulting calculation can be uniquely classified into orders and quantities"

Mixed arithmetic order: arithmetical operation involving an order as datum

Type determination Restrictions on the assignment of types and control on types during calculation; re: distinction between orders and quantities; typically determinate

Fixed as to type the type of a word in a location never changes

Primary program typically determinate and no mixed arithmetic order

Secondary program at least one mixed arithmetic order

Table condition restriction on mixed arithmetic order and on location subsitution: all within table range

Regular program a primary program or one that satisfies the table condition; typically determinate; calculation terminates

1949: "On the composition of programs for automatic computing"

Transformations "In forming new programs from combinations of old ones, two fundamental concepts are those of transformation and replacement."

Given:

$$\begin{aligned} X &= M_0 M_1 M_2 \ldots M_p \\ Y &= N_0 N_1 N_2 \ldots N_q \\ Z &= L_0 L_1 L_2 \ldots L_r \\ T(k) &= k' \qquad\qquad k \leq m, k' \leq n,\ T \text{ is some numerical function} \end{aligned}$$

Transformation of the first kind: reshuffle of the words in a program (T)(X): gives the Y such that $q = p$ and every N_i is derived from M_i by replacing every location number k in every order of X by $T(k)$

Transformation of the second kind: change of datum and exit numbers in the orders so as to correspond with the reshuffle.

$$\{T\}(X) = Y = \begin{cases} N_0 = M_0 \\ N_{T(i)} = M_i \text{ if T is defined for i }, i > 0(*) \\ N_i = J \text{ else} \end{cases}$$

1949: "On the composition of programs for automatic computing"

Transformations and replacement

Replacement a program made up from two programs by putting, in certain locations of one program, words from corresponding locations in the other program. Let $\phi \subset \{0, 1, 2, ..., p\}$, $\frac{\phi}{Y} X = Z$ with:

$$L_i = \begin{cases} M_i \text{ if } i \notin \phi, i \leq p \\ N_i \text{ if } i \leq q \text{ and } i \in \phi \text{ or } i > p \\ J \text{ if } i \in \phi, i > q \end{cases}$$

$$\frac{X}{Y} \text{ when } \phi \text{ is void : X with spaces}$$

Transformations of the second kind with replacement

$$\{\frac{\phi, T}{Y}\} = \frac{\phi}{Y}(\{T\}(X))$$

Transformations of the third kind

$$\begin{cases} [T](x) = \{T\}(T)(x) & \text{combination of kind 1 and 2} \\ [\frac{T}{Y}] = \{\frac{T}{Y}\}(T)(x) & \text{combination ofkind 1 and 2 and replacement} \\ [\phi T](x) = \{\frac{\phi T}{0}\}(T)(x) & \text{combination of kind 1 and 2and take subprogram from X} \\ S(x) = [\frac{\phi T}{Y}](x) = \{\frac{\phi T}{Y}\}(T)(x) & \text{combination of kind 1 and 2 and replacement} \end{cases}$$

1949: "On the composition of programs for automatic computing"
Combinators and (*)

$$\begin{cases} \mathbf{C}xyz \Rightarrow xzy \\ \mathbf{W}xy \Rightarrow xyy \\ \mathbf{K}xy \Rightarrow x \end{cases}$$

$T(k) = i, i > 0(*)$

K-free: (*) always has at least one solution

W-free: (*) has no multiple solutions

C-free: T is monotone increasing

1949: "On the composition of programs for automatic computing"
Homomorphic Transformations and normal programs

"A homomorphic transformation does not change the results of a calculation, i.e., the successive configurations in the transformed calculation can be derived from those of the original by the same transformation"

Difference condition: $T(x) - T(y) = x - y$

Theorem: Let X be a regular program and T a numerical transformation which is W-free and satisfies the difference condition. Then, for any Y, $[\frac{T}{Y}]$ is a homomorphic transformation for the calculation initiated by X.

Normal Program: $X = AC$, A is an order program and C a quantity program

1949: "On the composition of programs for automatic computing"

Program composition Basic concept: substitution: "A program Z will be said to be formed by substitution of Y for a certain output in X, when Z carries on a calculation homomorphic to X until the control reaches that output, then starts a calculation homomorphic to Y using the quantities calculated by X as quantity program"

Five types of substitution:

A. Simple substitution

B. Multiple substitution

C. Reduction to a single quantity program

D. Loop programs

E. Complex programs

1949: “On the composition of programs for automatic computing”

Simple substitution

$$X = AC$$
$$Y = BC$$

X and Y are normal

m is the location number of $M \in A$ to be substituded, then

$$T_1(k) = \begin{cases} k & \text{for } 0 < k < m \\ m+n-1 & \text{for } k = m \\ k + |B| - 1 & \text{for } m < k \leq |A| + |C| \end{cases}$$

$$T_2(k) = \begin{cases} m+k-n & \text{for } n \leq k \leq n + |B| \\ |A| + k - n & \text{for } n + |B| < k \leq n + |B| + |C| \end{cases}$$

Then:

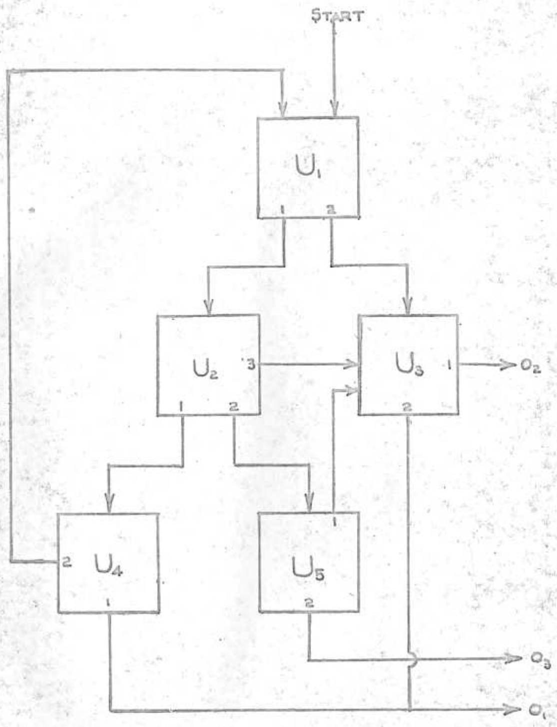
$$Z = [\frac{\phi T_1}{[T_2](Y)}](x)$$

Notation:$Z = X \rightarrow Y$

1949: “On the composition of programs for automatic computing”

Notations...

Flow charts



FLOW CHART FOR SAMPLE PROGRAM

FIG. I

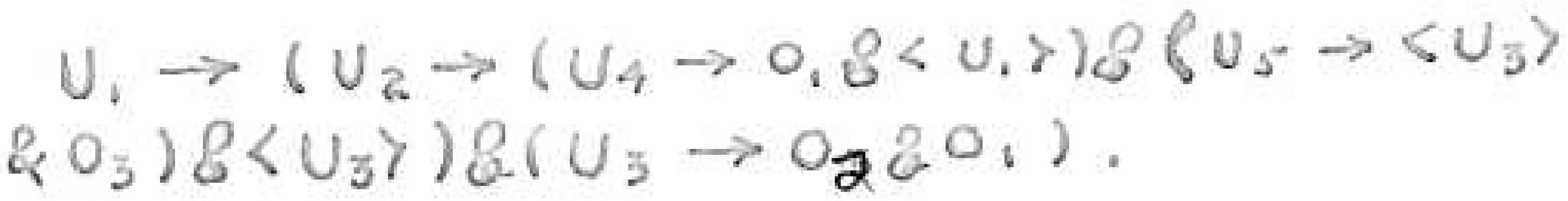
UNCLASSIFIED - 52 - NOLM 9805

PRNC-9902-3-28-40-120

1949: “On the composition of programs for automatic computing”

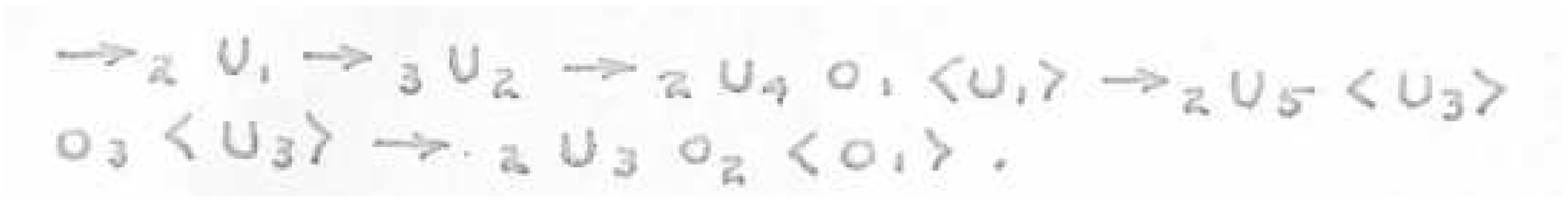
Notations...

Curry Notation:



$U_1 \rightarrow (U_2 \rightarrow (U_4 \rightarrow O_1 \& \langle U_1 \rangle) \& (U_5 \rightarrow \langle U_3 \rangle \& O_3) \& \langle U_3 \rangle) \& (U_3 \rightarrow O_2 \& O_1).$

Polish notation:

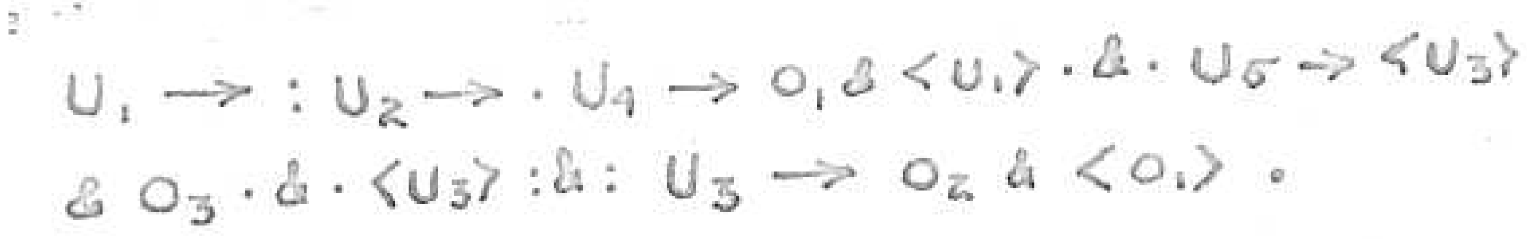


$\rightarrow_2 U_1 \rightarrow_3 U_2 \rightarrow_2 U_4 \; O_1 \langle U_1 \rangle \rightarrow_2 U_5 \langle U_3 \rangle \; O_3 \langle U_3 \rangle \rightarrow_2 U_3 \; O_2 \langle O_1 \rangle .$

1949: "On the composition of programs for automatic computing"

Notations...

Peano notation:

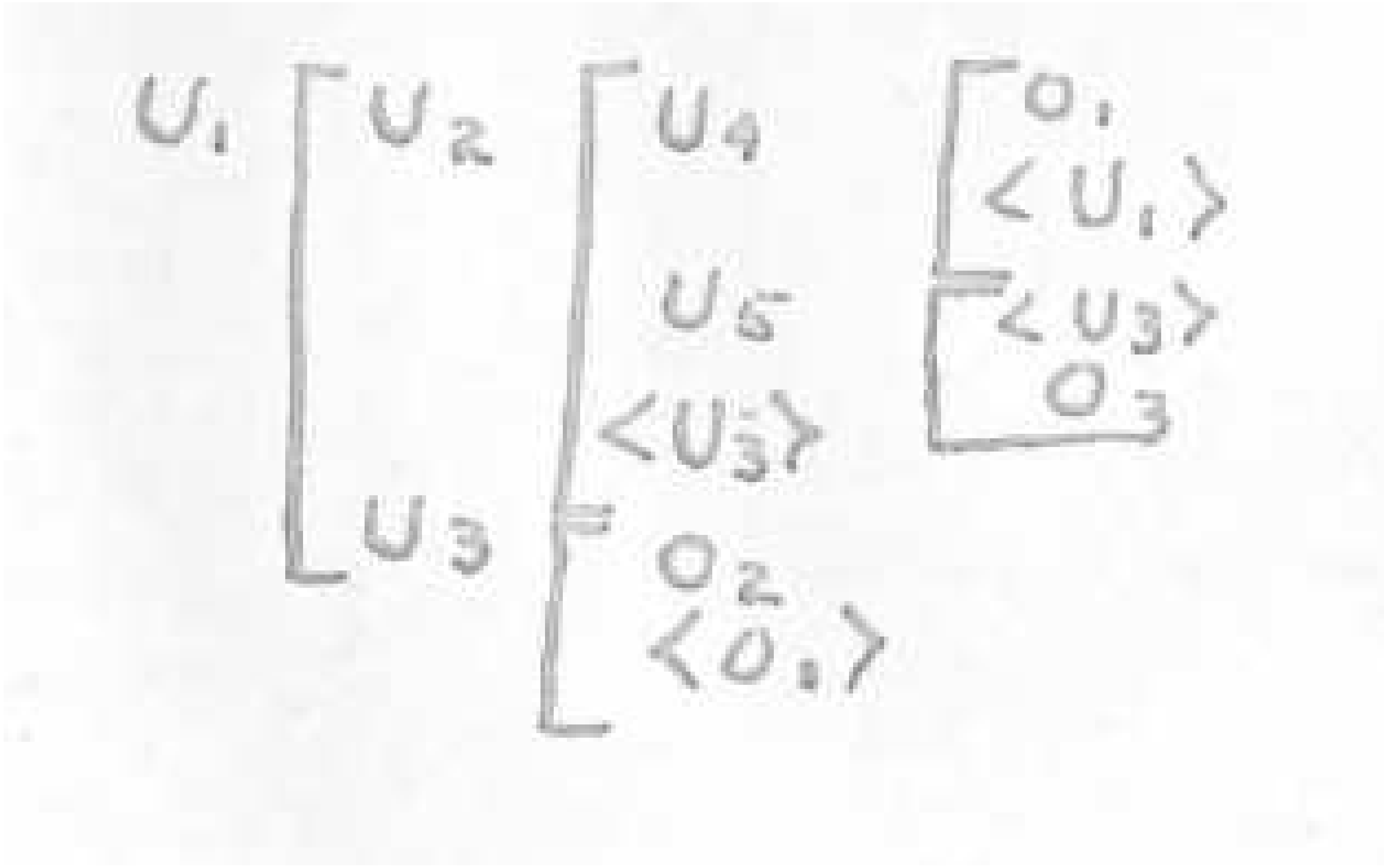


$$U_1 \rightarrow : U_2 \rightarrow . U_4 \rightarrow O_1 \,\&\, \langle U_1 \rangle . \&. U_5 \rightarrow \langle U_3 \rangle$$
$$\& \, O_3 . \&. \langle U_3 \rangle :\&: U_3 \rightarrow O_2 \,\&\, \langle O_1 \rangle .$$

1949: “On the composition of programs for automatic computing”

Notations...

Begriffschrift:



“Frege’s notation, it must be remembered, died with him”

1950: A program composition technique as applied to inverse interpolation

1950: “A program composition technique as applied to inverse interpolation”
Some highlights

Synthesis of program of inverse interpolation (re: analysis ’49)

Analysis into basic programs: “This analysis can, in principle at least, be carried clear down until the ultimate constituents are the simplest posible programs [...] Of course, it is a platitude that the practical man would not be interested in composition techniques for programs of such simplicity, but i is a common experience in mathematics that one can deepen ones insight into the most profound and abstract theories by considering trivially simple examples.”

Synthesis of programs (in general):

arithmetic programs: compiler for arithmetic procedures, i.e., “complete theory for the construction of an arbitrary such program. This program will not always be the shortest one possible to attain the required result; but, at least, it will be automatic as soon as certain decisions are made.”

Discrimination programs

Secondary programs

1950: “A program composition technique as applied to inverse interpolation”

One example in the new notation

$$Y \rightarrow (It(m,i) \rightarrow I \text{ and } O_2)$$

$$It(m,i) = \{i : A\} \rightarrow \{A+1 : A\} \rightarrow \{A : i\} \rightarrow \{m : A\} \rightarrow \{A-i : A\} \rightarrow \{A < 0\}$$

$$n! = \{1 : A\} \rightarrow \{A : x\} \rightarrow \{A : i\} \rightarrow Y \rightarrow It(n,i)$$

$$Y = \{ix : x\}$$

Discussion

Discussion

"The objective was to create a programming technique based on a systematic logical theory. Such a theory has the same advantages here that it has in other fields of human endeavor. Toward that objective a beginning has been made." (Curry,1950)

"This automatic proramming is anticipated by the author (Patterson in a review on Curry, 1957)
"Now it is an important fact that the actual construction of a program indicated in the above symbolism is a mechanical process. It can be carried out, at least in principle, by non-technical personnel or by a machine. The main machine itself may also be used for that purpose."

"When these processes [of composition] are combined with one another, there will be evidently be equivalences among the combinations. There will thus be a calculus of program composition. This calculus will resembles, in many respects the ordinary calculus of logic. It can be shown, for example, that the operation "$\rightarrow$" is associative. But the exact nature of the calculus has not, so far as I know, been worked out."

$\Rightarrow$ Work in progress!