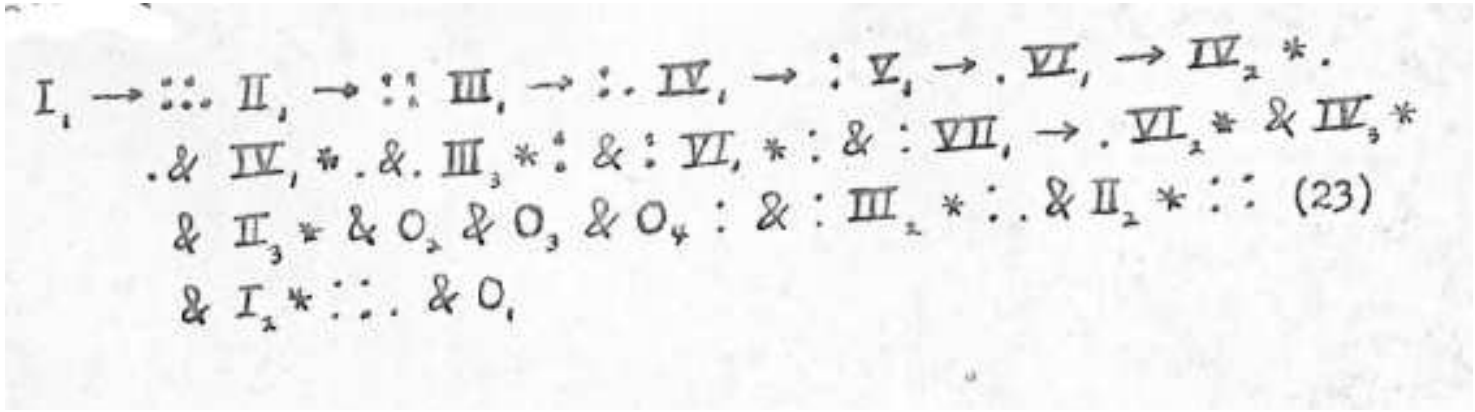
Haskell before Haskell. Curry's contribution to programming (1946–1950)



L. De Mol[1], M. Bullynck[2] and M. Carlé[3]

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

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

[3] Athens, mc@aiguphonie.com

Overview

- A logician and the Eniac
- On the composition of programs for automatic computing
- Curry vs. Goldstine-von Neumann
- Discussion

A logician and the Eniac

A logician and the Eniac

- Haskell B. Curry before World War II: Tour in Europe (Göttingen...); work on combinatory logic; from 1929 onwards professor at Penn University
  A guiding idea in Curry's work *"[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*** *"* (Curry 1942)
- Haskell B. Curry during World War II: 1942–1944: work at Frankford Arsenal and Applied Physics Laboratories; from 1944 Ballistic Research Laboratories (Aberdeen Proving Ground) who had ordered the building of ENIAC
- The Ballistic Research Laboratories 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. Its members were:
  * Leland B. Cunningham (an astronomer)
  * **Haskell B. Curry (a logician)**
  * Derrick H. Lehmer (a number theorist)

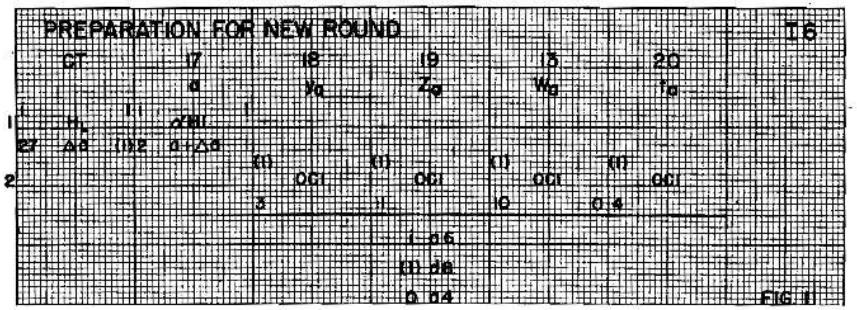
Curry and Wyatt's program on the ENIAC

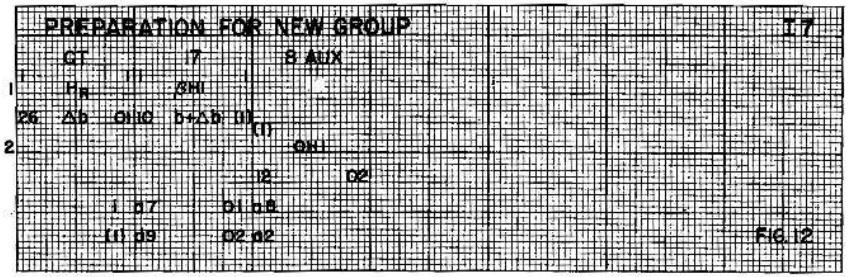
- In collaboration with Willa Wyatt, one of ENIAC's female programmers, Curry wrote up a technical report "A study of inverse interpolation of the Eniac" (1946, declassified in 1999)

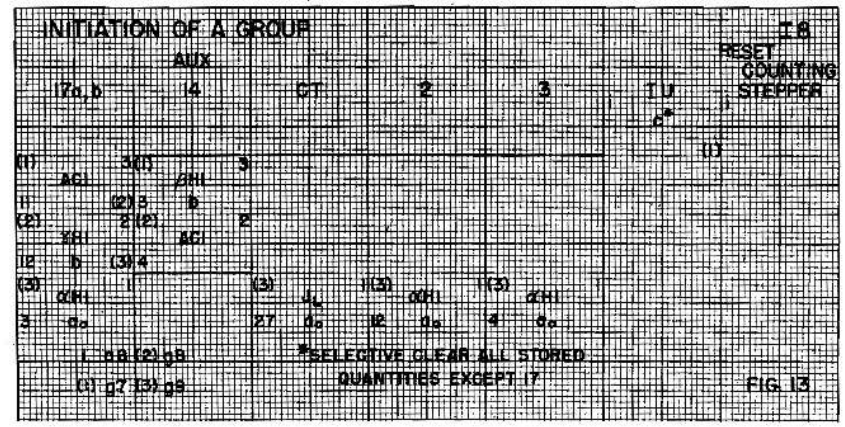

- *"The problem of inverse interpolation [...] is important in the calculation of firing tables. Suppose the trajectory calculations have given us the coordinates (x, y) of the projectile as functions of t (time) and ϕ (angle of departure). For the tables we want t and ϕ as functions of x and y; indeed we wish to determine ϕ so as to hit a target whose position (x, y) is known, and t is needed for the fuze 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."*

Curry and Wyatt's program on the ENIAC: Stages and processes

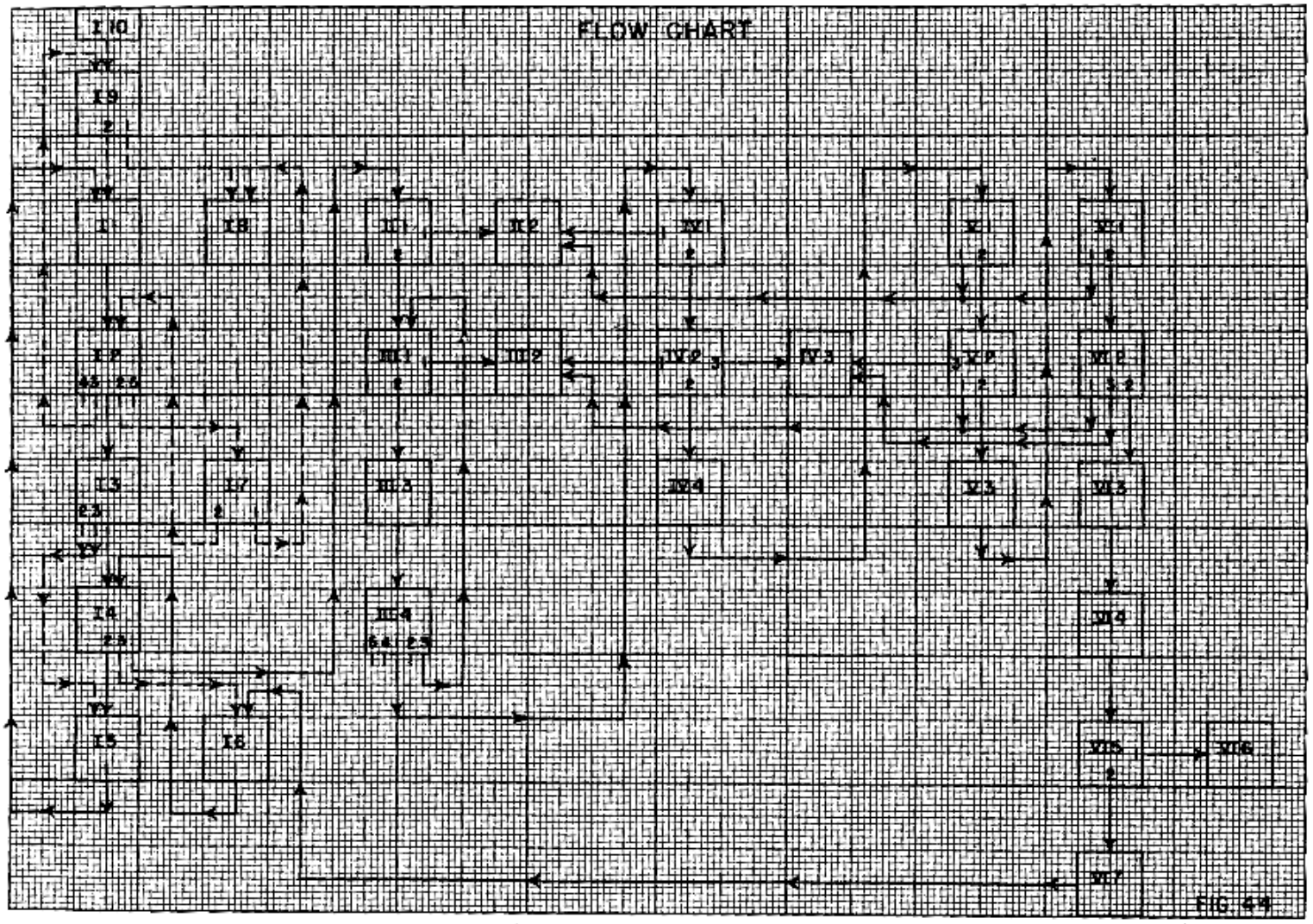
- *"In this report the problem of inverse interpolation is studied with reference to the programming on the ENIAC as a problem in its own right."*
- *"The entire computation procedure can [...] be divided into certain major parts which are repeated over and over according to the programming. These major 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
T6
CT
FIG 11


PREPARATION FOR NEW GROUP
T7
CT
8 AUX
FIG 12


INITIATION OF A GROUP
T8
AUX
CT
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	18	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
FIG 44

Theoretical considerations in the 1946 report

- *"[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 Eniac experience and the program of inverse interpolation triggers Curry's interest to develop the topic further:
  - *"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)
  - *"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)

The composition of programs

After the ENIAC experience

- Curry reads the John von Neumann - H.H. Goldstine reports
  - *Preliminary discussion of the logical design of an electronic computing instrument.* 1946–1947 (EDVAC report)
  - *Planning and coding of problems for an electronic computing instrument.* parts I,II and III, 1947–48.
- Building upon his readings and his ENIAC experience, Curry writes up two technical reports for the Navy Ordnance (unclassified)
  - 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 (= a short resumé of the two preceding reports)

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."*

On the composition of programs: Definitions and assumptions

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

Two types of Words: **quantities** and **orders**. Orders can be: 1) datum number location, 2) exit number location and 3) an operator. They can be classified as arithmetical, transfer, control, stop orders. Of special interest is the **Mixed arithmetic order**: arithmetical operation involving an order as datum (cfr. partial substition in Goldstine-von Neumann)

"The distinction between quantities and orders is not a distinction of form [...] The machine makes this distinction according to the situation. Making this classification of words in advance is a difficult problem [...] [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"

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

Normal Program: $X = AC$, A is an order program and C a quantity program (a normal program can be obtained from a regular program through transformations of the 1st and 2nd kind as Curry proves)

On the composition of programs: transformations

Given the programs X, Y, Z and numerical function T(k):

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

Transformation of the first kind: reorienting the location numbers in a program

Y=(T)(X): T(X) gives the Y such that $n = m$ (m is range of location numbers in X) 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: reshuffling the words to match up with the changes in location numbers.

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

On the composition of programs: 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 $\Theta \subset \{0, 1, 2, ..., p\}$ (a list of integers), then the replacement $\frac{\Theta}{Y}X = Z$ is:

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

When $\Theta = \emptyset$ then $\frac{X}{Y} = X$ with spaces after

On the composition of programs: Substitution

Simple **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"

Notation: $Z = X \to Y$

$X = AC$ and $Y = BC$ are normal, m is the location number ($m \in A$) at which Y ist to be substituted, then $Z = X \to Y = S_Y(X) = [\frac{\Theta T_1}{[T_2](Y)}](X) = \frac{\Theta T_1}{[T_2](Y)}(T_1)(X)$ is defined by

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

$$T_2(k) = \begin{cases} m + k - n & \text{for } n \leq k \leq n + |B| - 1 \\ |A| + k - n & \text{for } n + |B| < k \leq n + |B| + |C| - 1 \end{cases}$$

On the composition of programs: Substitution

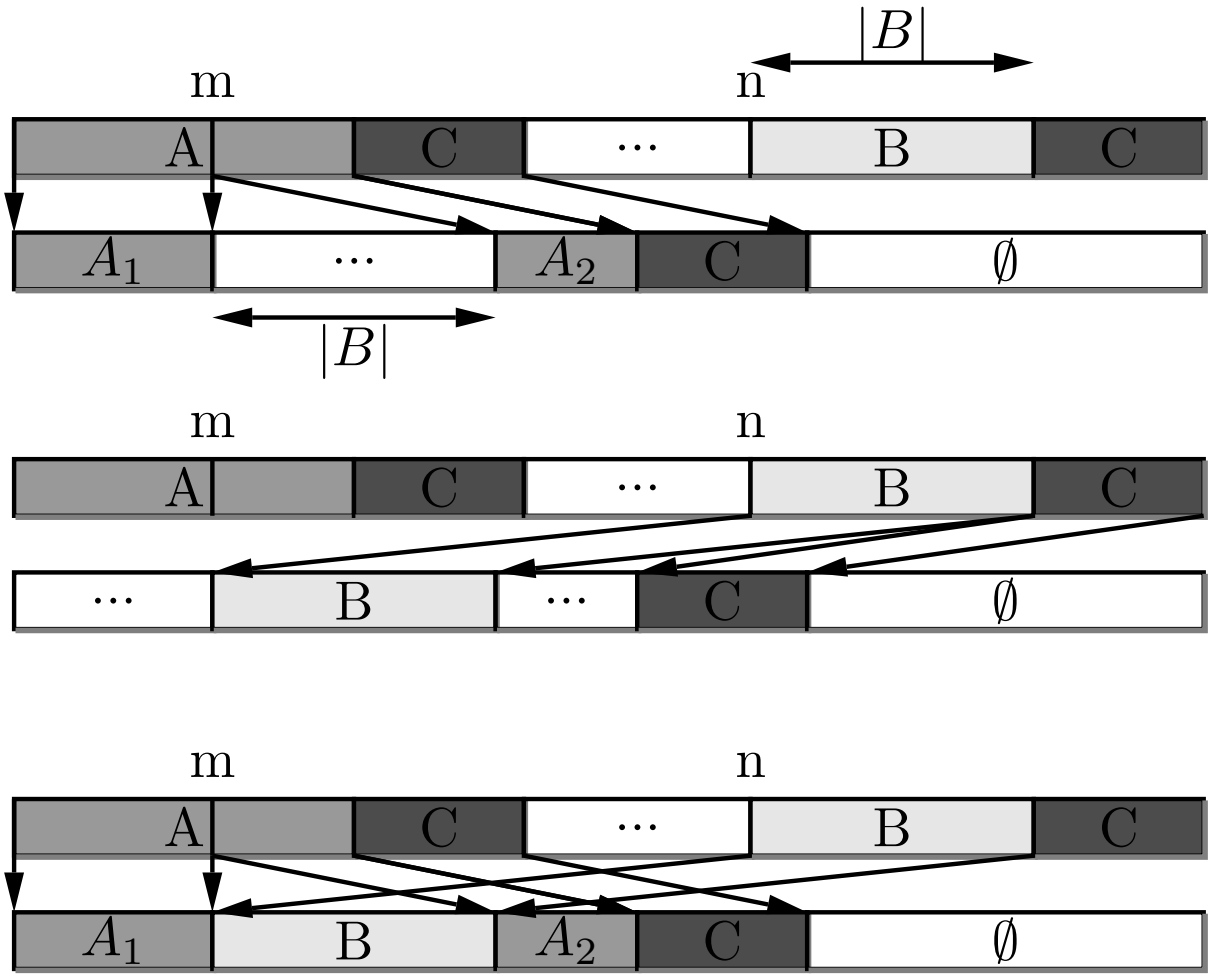


Figure 1: From top to bottom: The $T_1(X)$ transformation; the $T_2(Y)$ transformation; and finally the substitution $[\frac{\Theta T_1}{[T_2](Y)}](X)$ that substitutes Y in X at position m.

Curry vs. Goldstine-von Neumann

The Curry approach vs. Goldstine-von Neumann approach

A program composition technique as applied to inverse interpolation (1950):
Synthesis of inverse interpolation + discussion of many issues in programming

"(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"

The Curry approach vs. Goldstine-von Neumann approach: Determination of a basic instruction set (vs. EDVAC-order set)

Assignment: If ξ is a term and λ a locatum, then $\{\xi : \lambda\}$ is a program that calculates the term ξ and stores it in the locatum λ

Basic arithmetic instructions:

$$\pi_0(t) = +t \qquad \pi_1(t) = -t$$
$$\pi_2(t) = +|t| \qquad \pi_3(t) = -|t|$$

Two **mixed arithmetic orders**, i.e., $d(*)$ and $e(*)$ where $d(*)$ is an order that reads the location number of its own datum into the accumulator and $e(*)$ an order that reads the location number of its own exit number into the accumulator

A **conditional and unconditional jump and a stop** instruction.

Using these instructions **plus assignment** (assignment being a simple kind of composition), Curry produces all simple instructions in the Goldstine-von Neumann EDVAC vocabulary

The Curry approach vs. Goldstine-von Neumann approach: Compiling arithmetic expressions (vs. Ad-hoc algorithms)

arithmetic programs: compiler for arithmetic procedures, a *"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."*

Curry gives inductive definitions for forming (and thus analyzing) arithmetic formulae

If one converts the expression $(x+1)(y+1)(z+1)$ into Curry's notation for the composition of programs, one gets:

$$\{x:A\} \to \{A+1:A\} \to \{A:w\} \to \{y:A\} \to \{A+1:A\} \to \{A:R\} \to$$
$$\{wR:A\} \to \{A:w\} \to \{z:A\} \to \{A+1:A\} \to \{A:R\} \to \{wR:A\}$$

Curry gives similar 'partial compilers' for formulae in predicate logic, and for tabulating commands

The Curry approach vs. Goldstine-von Neumann approach: A calculus of programs (vs. automata theory?)

> *"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 "→" is associative. But the exact nature of the calculus has not, so far as I know, been worked out."* → Iuri Ianov's calculus of programs (1959)

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."*