

## Remarks on flowcharts

***Citation for published version (APA):***

Bruijn, de, N. G. (1983). *Remarks on flowcharts*. (Publicaties de Bruijn; Vol. M14). Technische Universiteit Eindhoven.

***Document status and date:***

Published: 01/01/1983

***Document Version:***

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

***Please check the document version of this publication:***

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

***General rights***

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

[www.tue.nl/taverne](http://www.tue.nl/taverne)

***Take down policy***

If you believe that this document breaches copyright please contact us at:

[openaccess@tue.nl](mailto:openaccess@tue.nl)

providing details and we will investigate your claim.

Remarks on flowcharts

by

N.G. de Bruijn

May 1983

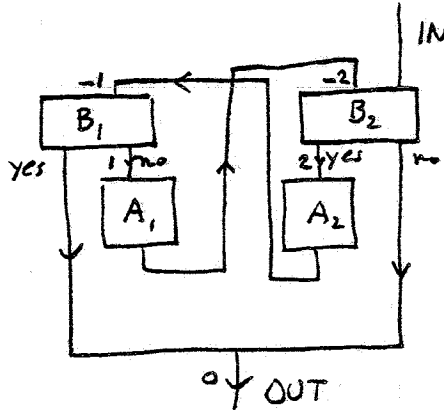
Department of Mathematics and Computing Science  
Eindhoven University of Technology  
PO Box 513, 5600 MB Eindhoven  
The Netherlands

Remarks on flowcharts.

1. Every flowchart can be written as a do-loop.

A flowchart can be described as a set of program fragments  $A_1, \dots, A_m$  each one of which has just a single exit, and a set  $B_1, \dots, B_n$  each one of which has two exits (yes and no). An  $A_j$  is followed by goto  $P(j)$ , and  $B_j$  at the exit "yes" by goto  $Q(j)$ , at the exit "no" to goto  $R(j)$ . The labels  $P(j), Q(j), R(j)$  belong to  $\{-n, \dots, m\}$ . Here  $+j$  is the label we have at the entrance of  $A_j$  ( $1 \leq j \leq m$ ) and  $-i$  is the label of the entrance of  $B_i$  ( $1 \leq i \leq n$ ). The label 0 refers to the program exit; the entry label is a.

Example :



Here we have  $P(1) = -2, P(2) = -1, Q(1) = 0, Q(2) = 2, R(1) = 1, R(2) = 0$ .

Starting with the declaration of an integer  $i$  we write the program like this:

```

i = -2;
do  i = 1 → A1 ; i := -2
    [ i = 2 → A2 ; i := -1
    [ i = -1 ∧ B1 → i := 0
    [ i = -1 ∧ ¬B1 → i := 1
    [ i = -2 ∧ B2 → i := 2
    [ i = -2 ∧ ¬B2 → i := 0
od
    
```

Sometimes one can describe the set of  $A_i$ 's as a single program  $A[i]$  with parameter  $i$ , and the  $B_i$ 's as  $B[i]$ . In general, the program now becomes

```

i := a;
do   i > 0 → A[i]; i := P[i]
     i > 0 ∧ B[-i] → i := Q[-i]
     i < 0 ∧ ¬B[-i] → i := R[-i]
od

```

## 2. A class of structured programs

We shall now remove the distinction between A's and B's. We consider programs with entries  $\{1, \dots, m\}$  and exits  $\{1, \dots, n\}$ .



If the index  $i$  indicates one of the entries, the assignment  $x := \text{exit}(M, i)$  is intended to say that  $M$  has to be executed with entry  $i$ , and that after the execution the value of the exit (i.e. one of the numbers  $1, \dots, n$ ) is assigned to  $x$ .

We have an index set  $B$ ; for every  $b \in B$  we have a program  $M(b)$  with  $P(b)$  as a set of entries and  $Q(b)$  as a set of exits.  $B$  is partitioned like this:  
 $B = B_0 \cup B_1 \cup B_2 \cup \dots$

If  $b \in B_0$  we consider  $M(b)$  as a given black box. If  $n > 0$  and  $b \in B_n$  then  $M(b)$  will be defined by means of the  $M(c)$ 's with  $c \in B_0 \cup \dots \cup B_{n-1}$ .

If  $b \in B_k$  with  $k > 0$  then there is a "label set"  $L(b)$  with subsets  $P(b)$  and  $Q(b)$  ("entries" and "exits", respectively;  $P(b)$  and  $Q(b)$  are not necessarily disjoint).

If  $b \in B_k$  with  $k > 0$  and  $j \in L(b) \setminus Q(b)$  then  $\varphi(b, j)$  is given as an element of  $B_0 \cup \dots \cup B_{k-1}$ , and  $\psi(b, j)$  as element of  $P(\varphi(b, j))$ .

If  $b \in B_k$  with  $k > 0$ , and  $i \in P(b)$ , then the program

```
x := exit(M(b), i)
```

is defined as

```

x := i;
do  x ∈ L(b) \ Q(b) → x := exit(M(φ(b, x)), ψ(b, x))
od

```

In order to get to programs with more than one exit one can assume that  $B_0$  contains "question blocks", i.e. programs with a single entry 1 and two exits ("yes" and "no"). If  $b \in B_0$  then  $x := \text{exit}(M,1)$  will not modify the state space but just indicate one of the two exits. We might even go so far that we get rid of the state space entirely, e.g. by taking the state space as a part of  $L(b)$ , like taking it as a cartesian factor of  $L(b)$ . Then we can consider the  $M$ 's as procedures, and selecting the actual values of the procedure parameter can be interpreted as selecting the entrance.

It seems, however, that the pursuit of this kind of abstractions conflicts with the tendencies in program languages; it may lead back to something like machine code.

One can get non-deterministic programs by starting from primitive programs of a kind where the selection of  $\text{exit}(M,i)$  is left to the "free will" of a computer.

Remark: Most of the contents in this note can be found in some form in "On Folk Theorems" by David Harel, Communications ACM. Vol. 20 (7) 379-389.