On Understanding Programs.

On a number of occasions I have stated the requirement that if we ever want to be able to compose really large programs reliably, we need a discipline such that the intellectual effort E (measured in some loose sense) needed to understand a program does not grow more rapidly than proportional to the program length L (measured in an equally loose sense) and that if the best we can attain is a growth of E proportional to, say, L^2, we had better admit defeat. As an aside I used to express my fear that many programs were written in such a fashion that the functional dependence was more like an exponential growth!

I now offer, for the reader's consideration, an example showing how the same program can be understood in two different ways: in the one way, E grows proportional to L, in the other way it grows (even worse than) exponentially.

I express E in the number of "steps of reasoning" needed, a number which is determined as follows.

Let us consider a "stretched" program of the form

"S_1; S_2; ...; S_N" , (1)

i.e. the corresponding computations are a time-succession of the executions of S_1 through S_N in that order. When the nett effect of the execution of each individual statement S_i is given, my measuring convention states that it takes N steps to understand program (1), i.e. to establish that the cumulative effect of the N successive actions satisfies the requirements imposed upon the computations evoked by program (1).

In the case of a program of the form

"<u>if</u> B <u>then</u> S_1 <u>else</u> S_2" (2)

where, again, the nett effect of the execution of the statements S_1 and S_2 has been given, my measuring convention states that it takes 2 steps to understand program (2), viz. one step for the case B and another for the case <u>non</u> B.

Consider now a program of the form

"if B_1 then S_{11} else S_{12};
if B_2 then S_{21} else S_{22};
⋮
if B_N then S_{N1} else S_{N2}" (3)

According to the measuring convention it takes 2 steps per alternative statement to understand it, i.e. to establish that the nett effect of

"if B_i then S_{i1} else S_{i2}"

is equivalent to that of the execution of an abstract statement S_i. Having N such alternative statements it takes us 2N steps to reduce program (3) to one of the form of program (1); to understand the latter form of the program takes us another N steps, giving 3N steps of reasoning in toto.

If we had not introduced the abstract statements S_i, but had tried to understand program (3) directly in terms of executions of the statements S_{ij}, each such computation would be the cumulative effect of N such statement executions and would as such require N steps to understand it. Trying to understand the algorithm in terms of the S_{ij}, however, implies that we have to distinguish between 2^N different routings through the program and this would lead to $N*2^N$ steps of reasoning!

If by now the reader protests that the second way to try to understand program (3) is utterly foolish, then I have him exactly in the position where I want him to be, for then we could not agree more fully. The point is, that any effort to improve upon the second method yields a method more similar to the first one: one will find oneself introducing the abstract statements S_i (or combinations of them)!

The whole demonstration is an urgent plea to use our powers of abstraction as consciously as possible (and to restrict ourselves in programming to those program structures in which these powers can be exploited at greatest advantage!) The reader who has followed me thus far may wonder whether he has been confronted

with something deep or something trivial. So did I, when I discovered this example; I have, however, decided to submit it for publication when I felt that it was probably both.

Edsger W.Dijkstra
Technological University
EINDHOVEN
The Netherlands

~~On the understandability of programs.~~ constructing understandable programs.

~~On Understanding Programs.~~

On a number of occasions I have stated the requirement that if we ever want to be able to compose really large programs reliably, we need a discipline such that the intellectual effort E (measured in some loose sense) needed to understand a program does not grow more rapidly than ~~proportional~~ in proportion to the program length L (measured in an equally loose sense) and that if the best we can attain is a growth of E proportional to, say, L^2, we had better admit defeat. As an aside I used to express my fear that many programs were written in such a fashion that the functional dependence was more like an exponential growth!

I now offer, for the reader's consideration, an example showing how the same program can be understood in two different ways: in the one way, E grows proportional to L, in the other way it grows (even worse than) exponentially.

I express E in the number of "steps of reasoning" needed, a number which is determined as follows.

Let us consider a "stretched" program of the form

$$\text{"}S_1;\ S_2;\ \ldots;\ S_N\text{"} \quad , \qquad (1)$$

i.e. the corresponding computations are a time-succession of the executions of S_1 through S_N in that order. When the nett effect of the execution of each individual statement S_i is given, my measuring convention states that it takes N steps to understand program (1), i.e. to establish that the cumulative effect of the N successive actions satisfies the requirements imposed upon the computations evoked by program (1).

In the case of a program of the form

$$\text{"}\underline{\text{if}}\ B\ \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2\text{"} \qquad (2)$$

where, again, the nett effect of the execution of the statements S_1 and S_2 has been given, my measuring convention states that it takes 2 steps to understand program (2), viz. one step for the case B and another for the case <u>non</u> B.

Consider now a program of the form

"if B_1 then S_{11} else S_{12};
if B_2 then S_{21} else S_{22};
⋮
if B_N then S_{N1} else S_{N2}" . (3)

According to the measuring convention it takes 2 steps per alternative statement to understand it, i.e. to establish that the nett effect of

"if B_i then S_{i1} else S_{i2}"

is equivalent to that of the execution of an abstract statement S_i. Having N such alternative statements it takes us 2N steps to reduce program (3) to one of the form of program (1); to understand the latter form of the program takes us another N steps, giving 3N steps of reasoning in toto.

If we had not introduced the abstract statements S_i, but had tried to understand program (3) directly in terms of executions of the statements S_{ij}, each such computation would be the cumulative effect of N such statement executions and would as such require N steps to understand it. Trying to understand the algorithm in terms of the S_{ij}, however, implies that we have to distinguish between 2^N different routings through the program and this would lead to $N*2^N$ steps of reasoning!

If by now the reader protests that the second way to try to understand program (3) is utterly foolish, then I have him exactly in the position where I want him to be, for then we could not agree more fully. The point is, that any effort to improve upon the second method yields a method more similar to the first one: one will find oneself introducing the abstract statements S_i (or combinations of them)!

The whole demonstration is an urgent plea to use our powers of abstraction as consciously as possible (and to restrict ourselves in programming to those program structures in which these powers can be exploited at greatest advantage!) The reader who has followed me thus far may wonder whether he has been confronted

with something deep or something trivial. So did I, when I discovered this example; I have, however, decided to submit it for publication when I felt that it was probably both.

Edsger W.Dijkstra
Technological University
EINDHOVEN
The Netherlands