A simple fixpoint argument without the restriction to continuity

by

Edsger W. Dijkstra & A.J.M. van Gasteren

Abstract

In programming language semantics, the introduction of unbounded nondeterminacy, which amounts to the introduction of noncontinuous predicate transformers, is needed for dealing with such concepts as fair interleaving. With the semantics of the repetition given as the strongest solution of a fixpoint equation, the weakest precondition expressed in closed form would then require transfinite ordinals. Here, however, it is shown that, even in the case of unbounded nondeterminacy, the fundamental theorem about the repetition can be proved by a simple and quite elementary argument.

Introduction

<u>Notation</u> In this text, the letters B, P, Q, R, X and Y stand for predicates on the state space of a program and a pair of square brackets is used as a notation for universal quantification of the enclosed over the program variables.

The letter S stands for a statement and DO stands for the repetitive construct <u>do</u> $B \rightarrow S$ <u>od</u> .

For any statement S and predicate R , the operational interpretation of the predicate $wp(S, R)$ is that any valid implementation of S , when started in any state satisfying $wp(S,R)$, should lead to a finite computation that ends in a state satisfying R (see [3]). (End of Notation.)

Requiring DO to be semantically equivalent to its first unfolding

<u>if</u> $B \rightarrow S; DO$ [] $\neg B \rightarrow$ skip <u>fi</u>

yields that $wp(DO, P)$ is a solution of the equation in predicate X

(0) $[X \equiv (B \wedge wp(S,X)) \vee (\neg B \wedge P)]$.

<u>Explanation</u> Equation (0) follows from the required semantic equivalence and

(i) the semantic definition of skip , viz.

$[wp(\text{skip}, R) \equiv R]$ for all R ;

(ii) the semantic definition of statement concatenation, viz.

$$[wp(\text{"S0;S1"}, R) \equiv wp(S0, wp(S1,R))]$$
for all S0, S1, and R ;

(iii) the semantic definition of the alternative construct, in particular

$$[wp(\text{"}\underline{\text{if}}\ B \rightarrow S0\ [\!]\ \neg B \rightarrow S1\ \underline{\text{fi}}\text{"}, R) \equiv (B \wedge wp(S0,R)) \vee (\neg B \wedge wp(S1,R))]$$
for all B , S0 , S1 , and R .

For further details, see [3]. (End of Explanation.)

<u>Notation and terminology</u> With the exception of wp , functional application is denoted by juxtaposition and iterated functional composition by exponentiation.

"X is at least as strong as Y" means "$[X \Rightarrow Y]$" .

"predicate transformer f is monotonic" means "$[X \Rightarrow Y] \Rightarrow [fX \Rightarrow fY]$ for all X and Y" .

"predicate transformer f is <u>or</u>-continuous" means $[f(\underline{E}n: n \in \mathbb{N}: R_n) \equiv (\underline{E}n: n \in \mathbb{N}: fR_n)]$ for any weakening sequence of predicates, i.e. such that $(\underline{A}n: n \in \mathbb{N}: [R_n \Rightarrow R_{n+1}])$".
(End of Notation and terminology.)

For monotonic predicate transformer f , the equation $[X \equiv fX]$ in X has a strongest solution (see [8]). If, in addition, f is <u>or</u>-continuous, its strongest solution is given in closed form by

(1) $(\underline{E}n: n \in \mathbb{N}: f^n \text{false})$.

<u>Proof sketch</u> To show that (1) is at least as strong as any solution of $[X \equiv fX]$, f's monotonicity suffices; the proof is most easily carried out by mathematical induction on n . To show that (1) itself is a solution of $[X \equiv fX]$, we need f's <u>or</u>-continuity (which implies its monotonicity). (End of Proof sketch.)

The predicate wp(DO, P) is (see [5]) defined as the strongest solution of (0) . Because wp(S,X) is a monotonic function of X , so is the right-hand side of (0) ; hence the strongest solution of (0) exists. If, in addition, wp(S,X) is an <u>or</u>-continuous function of X , so is the right-hand side of (0) , and an expression of the form of (1) gives a closed expression for wp(DO, P) . In passing we note that wp(DO, ¬B ∧ P) is defined as the strongest solution of the same equation (0) , i.e.

$$[wp(DO, \neg B \wedge P) \equiv wp(DO, P)] .$$

In programming terms, <u>or</u>-continuity of wp is the same as nondeterminacy being bounded. The assumption of bounded nondeterminacy is a usual one to make: the closed form for wp(DO, P) , which is then available, is traditionally considered an advantage because it readily caters for the avoidance of fancy — and in practice cumbersome (see [1], [2]) — techniques like transfinite induction.

Since unbounded nondeterminacy cannot be implemented, the restriction to <u>or</u>-continuity has for a long time been regarded as quite reasonable. It has, however, led to theorems in which the restriction to or-continuity has been introduced not because the theorems demanded it but for the sake of their proofs. The restriction also became a nuisance in the mathematical treatment of abstract programs. Firstly, an abstract program may well contain the as yet unrefined statement "establish P", where P, viewed as equation, may have infinitely many solutions, an observation we owe to [0]. Secondly, the modelling of concurrency as a "fair" interleaving of atomic actions introduces unbounded nondeterminacy (see [6]).

We are therefore very pleased to show for the main theorem about the repetitive construct an elementary proof that, though not relying on or-continuity, does not require transfinite formalisms.

The theorem

Notation In what follows, x and y stand for elements of a set D. The infix operator "$\in$" for set-membership has a higher binding power than the logical operators. Function t is a mapping from the state space (of the program) to D, i.e., in each point of the state space, the value of t is an element of D, which can be summarized by $[t \in D]$. Let C be a subset of D; note that then $t \in C$ stands for a predicate that may be true in some points of the state space, and false in others. (End of Notation.)

For the notion "well-founded" we refer to the appendix, in which we show that well-foundedness is the same as the validity of a proof by mathematical induction. For that reason, the design of a well-founded set that carries the argument is a regularly recurring theme in arguments about algorithms. The best-known (but very special) well-founded set is the set of natural numbers with "<" having its traditional meaning; in the theorem as formulated below, D could be the set of integers with the subset of natural numbers as C.

After the above preliminaries we are ready to formulate the — well-known —

<u>Theorem</u> Let $(D,<)$ be a partially ordered set; let C be a subset of D such that $(C,<)$ is well-founded; let statement S, predicates B and P, and function t on the state space satisfy:

the predicate transformer $wp(S,?)$ is monotonic;

$[t \in D]$;

(2) $[P \wedge B \Rightarrow t \in C]$;

(3) $[P \wedge B \wedge t=x \Rightarrow wp(S, P \wedge t<x)]$ for all x ;

then

(4) $[P \Rightarrow wp(DO, \neg B \wedge P)]$,

where $wp(DO, \neg B \wedge P)$ is defined as the strongest solution of the equation in X

(0) $[X \equiv (B \wedge wp(S,X)) \vee (\neg B \wedge P)]$.

In the above, the well-informed reader will recognize in P the "invariant" of the repetition and in t its "variant function", which carries the termination argument.

The proof

Our proof obligation (4) amounts to showing $[P \Rightarrow Q]$ for any Q that solves (0). We do so by showing separately

(i) $[P \wedge \neg\, t \in C \Rightarrow Q]$ and

(ii) $[P \wedge t \in C \Rightarrow Q]$.

Proof of (i)

$$\begin{array}{ll} & true \\ = & \{(2) \text{ and predicate calculus}\} \\ & [P \wedge \neg\, t \in C \Rightarrow \neg B \wedge P] \\ \Rightarrow & \{[\neg B \wedge P \Rightarrow Q] \text{ since } Q \text{ solves } (0)\} \\ & [P \wedge \neg\, t \in C \Rightarrow Q] \quad . \end{array}$$

(End of Proof of (i).)

Proof of (ii)

This part of the proof uses the fact that C is well-founded. First we manipulate our demonstrandum so as to make it amenable to a proof by mathematical induction:

$$\begin{array}{ll} & [P \wedge t \in C \Rightarrow Q] \\ = & \{\text{predicate calculus, in particular the one-point rule}\} \\ & [(\underline{A} x : t = x : P \wedge x \in C \Rightarrow Q)] \\ = & \{\text{range and term manipulation}\} \\ & [(\underline{A} x : x \in C : P \wedge t = x \Rightarrow Q)] \\ = & \{\text{interchange of universal quantifications}\} \\ & (\underline{A} x : x \in C : [P \wedge t = x \Rightarrow Q]) \quad . \end{array}$$

In view of C's well-foundedness, the latter is proved by deriving - for any x in C -

$[P \wedge t=x \Rightarrow Q]$ from

(5) $(\underline{A}y : y \in C \wedge y<x : [P \wedge t=y \Rightarrow Q])$.

To this end we observe:

$$
\begin{array}{l}
\quad (5) \\
= \{\text{interchange of universal quantifications}\} \\
\quad [(\underline{A}y : y \in C \wedge y<x : P \wedge t=y \Rightarrow Q)] \\
= \{\text{range and term manipulation}\} \\
\quad [(\underline{A}y : t=y : P \wedge y \in C \wedge y<x \Rightarrow Q)] \\
= \{\text{one-point rule}\} \\
\quad [P \wedge t \in C \wedge t<x \Rightarrow Q] \\
= \{[P \wedge \neg t \in C \wedge t<x \Rightarrow Q] \text{ on account of (i)}\} \\
\quad [P \wedge t<x \Rightarrow Q] \\
\Rightarrow \{wp(S,?) \text{ is monotonic}\} \\
\quad [wp(S, P \wedge t<x) \Rightarrow wp(S,Q)] \\
\Rightarrow \{(3)\} \\
\quad [P \wedge B \wedge t=x \Rightarrow wp(S,Q)] \\
= \{\text{predicate calculus}\} \\
\quad [P \wedge t=x \Rightarrow (B \wedge wp(S,Q)) \vee \neg B] \\
= \{\text{predicate calculus}\} \\
\quad [P \wedge t=x \Rightarrow (B \wedge wp(S,Q)) \vee (\neg B \wedge P)] \\
= \{Q \text{ solves (0)}\} \\
\quad [P \wedge t=x \Rightarrow Q] \quad .
\end{array}
$$

(End of Proof of (ii).)

And this concludes our proof of the theorem.

Conclusion

As far as we are aware, Floyd (see [4]) has been the first one to formulate termination arguments in terms of well-founded sets; he did, however, restrict himself to deterministic programs, for which the natural numbers suffice.

In the fixpoint theory that became en vogue during the seventies, continuity was strictly adhered to, with the result that, again, the natural numbers sufficed (see [7]).

To the best of our knowledge, the above argument is the first one to connect well-foundedness in its full generality to a non-operational notion of termination, i.e. to the strongest solution of a fixpoint equation. Its simplicity should dispel the myth that the restriction to continuity for the sake of convenience is justified.

Finally we would like the reader to regard the effectiveness and austere rigour of our argument as a plea for the calculational proof method employed.

Acknowledgements

We are greatly indebted to C.A.R. Hoare for his formulation of antecedent (3), which he showed to lead to a shorter proof than our original formulation, to the Eindhoven Tuesday Afternoon Club for strengthening the theorem by replacing a former antecedent by the weaker (2), and to W.H.J. Feijen and C.S. Scholten, who independently suggested simplifications of the actual proof. Finally, we are endebted to one of the anonymous referees, whose constructive criticism led to yet a few further improvements in the presentation. (The other anonymous referee, we were sorry to observe, had done a rather poor job.)

A short appendix on well-foundedness

In the following,

$(C,<)$	is a partially ordered set,
x, y	are elements of C,
S	is a subset of C, and
Q	is a predicate on C,

where S and Q are coupled by

(7) $\quad Qx \equiv \neg\, x\in S \quad , \quad$ or $\quad S = \{x \mid \neg Qx\}$;

as a result, we have

(8) $\quad S = \emptyset \;\equiv\; (\underline{A}\, x : x\in C : Qx)$.

" x is a minimal element of S " means
$$x\in S \;\wedge\; (\underline{A}\, y : y<x : \neg\, y\in S) \quad .$$
" C is well-founded " means that any non-empty subset of C contains a minimal element.

We observe

" C is well-founded "
= {definitions of minimal element, well-foundedness and $\emptyset$}
$(\underline{A}\, S :: S\neq\emptyset \equiv$
$\quad (\underline{E}\, x : x\in C : x\in S \;\wedge (\underline{A}\, y : y<x : \neg\, y\in S)))$
= {predicate calculus, de Morgan in particular}
$(\underline{A}\, S :: S=\emptyset \equiv$
$\quad (\underline{A}\, x : x\in C : \neg\, x\in S \;\vee (\underline{E}\, y : y<x : y\in S)))$

= {renaming the dummy with (7) and (8)}

$(\underline{A}\, Q :: (\underline{A}\, x : x \in C : Q\,x) \equiv$

$\qquad (\underline{A}\, x : x \in C : Q\,x \ \vee\ (\underline{E}\, y : y < x : \neg Q\,y)))$

= {definition of mathematical induction}

"mathematical induction over C is valid" .

Among mathematicians, well-foundedness is not as well-known as it deserves to be: there is, for instance, after half a century not yet a Dutch name for it. One cannot escape the impression that Emmy Noether, besides being Jewish and female, had the additional disadvantage of having been preceded by Georg Cantor with his stress on countability.

References

[0] Back, R.J.R., Correctness preserving program refinements: proof theory and applications. Mathematical Centre Tracts, nr 131, Amsterdam 1980 .

[1] Back, R.J.R., Proving Total Correctness of Nondeterministic Programs in Infinitary Logic. Acta Informatica, vol. 15, Fasc. 3, 1981, pp. 233-249 .

[2] Boom, H.J., A Weaker Precondition for Loops. ACM Transactions on Programming Languages and Systems, vol. 4, 1982, pp. 668-677 .

[3] Dijkstra, Edsger W., A discipline of programming. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1976 .

[4] Floyd, R.W., Assigning meanings to programs. Amer. Math. Soc. Symposia in Applied Mathematics, vol. 19, 1967, pp. 19-31.

[5] Park, David, On the semantics of fair parallelism. Lecture Notes in Computer Science, vol. 86, Springer-Verlag, Berlin-Heidelberg 1980, pp. 504-526.

[6] Park, David, Concurrency and automata on infinite sequences. Lecture Notes in Computer Science, vol. 104, Springer-Verlag, Berlin-Heidelberg 1981 .

[7] Stoy, J., Denotational Semantics: The Scott-Strachey Approach to Programming. MIT Press, Cambridge, MA, 1977 .

[8] Tarski, A., A Lattice-theoretical Fixpoint Theorem and its Applications. Pacific Journal of Mathematics, vol. 5, 1955, pp. 285-309.

Austin, 13 October 1985

drs. A.J.M. van Gasteren
BP Venture Research Fellow
Dept. of Mathematics and
Computing Science
University of Technology
5600 MB EINDHOVEN
The Netherlands

prof. dr. Edsger W. Dijkstra
Dept. of Computer Sciences
The University of Texas at Austin
AUSTIN, TX 78712-1188
United States of America