stichting
mathematisch
centrum

MC

AFDELING INFORMATICA
(DEPARTMENT OF COMPUTER SCIENCE)

IW 104/78 DECEMBER

H.J. BOOM

A WEAKER PRECONDITION FOR LOOPS

Preprint

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O).

AMS(MOS) subject classification scheme (1970): 68A05

ACM-Computing Reviews-categories: 5.24

A weaker precondition for loops*)

by

H.J. Boom

ABSTRACT

The purpose of this note is to argue that, contrary to Dijkstra's pronouncements [1],

- Unbounded nondeterminism can be a meaningful programming tool,
- Its usefulness does not require the existence of infinitely nondeterministic hardware, and
- Dijkstra's technical difficulties with unbounded nondeterminism lie with the loop axiom, and not with nondeterminism per se.

In fact, a small change in his loop axiom will give unbounded nondeterminism. The new axiom is proved correct using constructive logic with bar induction.

KEY WORDS & PHRASES: *unbounded nondeterminism, loop axiom, weakest precondition, ordinal numbers, transfinite induction, Ackerman function.*

*) This report will be submitted for publication elsewhere.

1. USEFULNESS

The usefulness of unbounded nondeterminism will be suggested using an artificial example.

Suppose one is designing a computer system with a paging system. The page size, main storage size, and secondary storage size have not yet been determined. In keeping with the spirit of top-down design, we wish to defer these decisions until the various programs in the system are nearly complete. The first version of the paging program therefore begins with:

```
page size := arbitrary positive integer;
number of pages := arbitrary positive integer;
secondary storage size := arbitrary positive integer.
```

A nondeterministic program feature can thus correspond to a stepwise refinement not yet taken. It is not important whether the refinement is ultimately carried out when the program is written, when it is compiled, or during execution, nor whether the ultimate physical machine is deterministic.

The arbitrariness of the positive integer is exactly the same as that of *if true -> cough [] true -> sneeze fi*, except that there are infinitely many positive integers. When an abstract machine runs either of these programs, some choice will be made. Afterward, it is a matter of metaphysics whether some other choice *could* have been made. The metaphysics is not substantially altered by the number of paths not taken.

2. AXIOMATICS

The weakest-precondition axiom for the infinite-choice assignments above can be stated as follows.

$$wp(x := \text{arbitrary positive integer}, P) = \forall\ \underline{\text{int}}\ i: i > 0 : wp(x := i, P).$$

The arbitrary assignment is thus seen to have a fairly simple predicate transformer.

Dijkstra's loop axiom can be stated in a simplified form as follows.

Let

IF = if P1 -> S1 [] P2 -> S2 [] ... [] Pk = Sk fi,

DO = do P1 -> S1 [] P2 -> S2 [] ... [] Pk = Sk od, and

PP = P1 ∨ P2 ∨ ... ∨ Pk.

We require that the Pi be testable; that is, that it be possible to determine their truth or falsity.

Then let

$wp(IF, P) = (P1 \wedge wp(S1, P)) \vee (P2 \wedge wp(S2, P)) \vee \ldots \vee (Pk \wedge wp(Sk, P))$.

$H_n(P) = (P \wedge \neg PP) \vee \exists \text{ int } m : n > m \geq 0 \wedge wp(IF, H_m(P))$

for int $n \geq 0$.

$wp_D(DO, P) = \exists \text{ int } n : n \geq 0 \wedge H_n(P)$.

The subscript "D" stands for "Dijkstra".

Dijkstra's loop axiom fails on the following example.

```
LL =
       i = 0 -> j := arbitrary positive integer; i := 1
       [] i ≠ 0 ∧ j > 0 -> j := j - 1
LLIF = if LL fi
LINF
     = do LL od
     = do
          i ≠ 0 -> j := arbitrary positive integer; i := 1
          [] i ≠ 0
       od
```

In [1], Dijkstra is unable to account axiomatically for the termination of LINF. The trouble is that the first step through the loop (with i = 0 initially) determines an arbitrarily large number of remaining steps. With such a first step, it is impossible to fix an a priori upper bound on the number of execution steps. Wp_D requires that such an upper bound exists.

The step to reach unbounded nondeterminacy is to permit n and m in the loop axiom to range over ordinal numbers instead of integers, thus:

$H_\nu(P) = (P \wedge \neg PP) \vee \exists$ ordinal $\mu : \nu > \mu \wedge wp(IF, H_\mu(P))$ for ordinal ν.
$wp(DO, P) = \exists$ ordinal $\nu : H_\nu(P)$.

The use of transfinite induction to reach fixed points is not new, but it is still impressive how neatly is can be combined with Dijkstra's loop axiom. In fact, the use of ordinals almost requires a simpler formulation than in [1]. For a successor ordinal $\nu = 1$, the above formula for $H_\nu(P)$ is equivalent to: $H_{\lambda+1}(P) = \neg PP \vee wp(IF, IF, H_\lambda(P))$.

Calculation of the $H_K(\underline{true})$ for LINF gives:

$$PP = i = 0 \vee (i \neq 0 \wedge j > 0)$$
$$= (i = 0) \vee (j > 0)$$

$$H_n(\underline{true}) = (j \leq n \wedge i \neq 0) \text{ for integers } n \geq 0,$$

$$H_\omega(\underline{true}) = (j \leq 0 \wedge i \neq 0) \vee \exists \text{ int } n : H_n(P)$$
$$= (j \leq 0 \wedge i \neq 0) \vee \exists \text{ int } n : (j \leq n \wedge i \neq 0)$$
$$= (j \leq 0 \wedge i \neq 0) \vee i \neq 0$$
$$= i \neq 0$$

$$H_{\omega+1}(\underline{true}) = (j \leq 0 \wedge i \neq 0) \vee wp(LLIF, i \neq 0)$$
$$= (j \leq 0 \wedge i \neq 0) \vee \underline{true}$$
$$= \underline{true}.$$

Thus wp(DO, true) = true, as desired. On the other hand, wp_D(DO, true) = ($\exists$ int n : H_n(true)) = (i ≠ 0). The use of ordinals here does not mean that execution will only terminate after an infinite amount of time. It is instead a means of summarizing an infinite number of distinct possibilities, each of which terminates after a different finite number of steps.

3. EXAMPLE

The use of ordinal numbers can also be useful in completely deterministic situations. The theory of termination functions presented in [1] carries through if we let a terminating function be an ordinal-valued function of the machine state which decreases with every loop iteration. Such an ordinal function may often be easier to find than an integral function. For example, MANNA & WALDINGER [3] have issued the challenge of formulating a termination function for the iterative version of the Ackerman function. This is quite difficult if the termination function must be integer-valued, involving recursions of the same complexity as the Ackerman function itself, but it is easy to formulate in terms of ordinal numbers. The iterative Ackerman program reads as follows:

```
top := 2; s[1]:=m; s[2]:=n;
do
   top > 1 ∧ s[top-1]=0 -> s[top-1] := s[top]+1; top-:=1
   [] top > 1 ∧ s[top-1] ≠ 0 ∧ s[top]=0 -> s[top-1]-:=1; s[top]:=1
   [] top > 1 ∧ s[top-1] ≠ 0 ∧ s[top] ≠ 0 ->
     s[top+1] := s[top]-1;
     s[top] := s[top-1];
     s[top-1] -:= 1;
     top +:= 1
od
```

The loop invariant is that Ack(m, n) = Ack(s[0], Ack(s[1], Ack(s[2], ..., Ack(s[top-1], s[top]) ...))). The termination function is

$$t = \omega^{\omega s[top-1]+s[top]} + \left(\sum_{i=top-2}^{1} \omega^{\omega s[i]+\omega} \right)$$

The task of verifying that this is indeed a termination function is left to the reader. The expressions of the form $\omega a+b$ arise from the double recursion in the definition of the Ackerman function; the powers and summation arise from recursion elimination.

4. CORRECTNESS

Let M be an abstract nondeterministic machine with a countable set of states {s0, s1, ...} on which programs can run. (For a nonconstructive proof, countability is not required.) We shall use the letters s, t, and u for states, and c, d, and e for sequences of states. For any statement S and states s and t, define s{S}t to mean that execution of the statement S starting with state s will terminate and that it is possible for a single execution of the statement S to bring the machine from state s to state t. The notation P|s will be used to mean that the predicate P holds in state s.

Let $c = c_0, c_1, \ldots$ be a nondeterministic "execution sequence", i.e, a choice sequence of states of M so that for each $i \geq 0$, either $PP|c_i$ or $wp(IF, \underline{true})|c_i \Rightarrow c_i\{IF\}c_{i+1}$. We permit sequences to continue after loop termination or undefined execution of the IF as a formal device to unify the treatment of terminating and nonterminating loops. The termination condition T(P, c) is:

$$T(P, c) = \exists\ \underline{int}\ j : (P \wedge \neg PP|c_j) \wedge \forall k < j: PP \wedge wp(IF, \underline{true})|c_k.$$

T(P, c)|s says that the execution c of DO starting with state s eventually terminates, i.e., comes to a state for which $\neg$ PP holds, and that when it does so {for the first time}, P holds.

We shall write T(P, s) for a state s iff $\forall$ c: c_0 = s => T(P, c). T(P, c) thus says that every computation starting with s will terminate in a state for which P holds. To prove correctness of the new loop axiom, we must show that $\forall$ state s: wp(DO, P)|s => T(P, s).

Proof.

Let P and s be such that wp(DO, P)|s holds.

Construct the partial function ϕ as follows: For each state t and ordinal ν such that $H_\nu(P)|t$ and PP|t, there must exist an ordinal μ such that $\nu > \mu$ and wp(IF, $H_\mu(P)$)|t. Let $\phi(t,\nu)$ be such a μ. For other t and ν, $\phi(t, \nu)$ is undefined.

Since wp(DO, P)|s, there exists an ordinal ν such that $H_\nu(P)|s$. Let c be any execution sequence starting with c_0 = s. Then define the sequence λ of ordinals:

$$\lambda_0 = n$$
$$\lambda_{i+1} = \phi(c_i, \lambda_i).$$

Then λ is a decreasing sequence of ordinals which terminates when $\phi(c_j, \lambda_j)$ becomes undefined, i.e., there is a first j for which $\neg (H_{\lambda_j}(P)|c_j \wedge PP|c_j)$, and this first j gives us the last term λ_j. An easy induction on k gives us $H_{\lambda_k}(P)|c_k)$ for all k for which λ_k exists, in fact for all $k \leq j$. Because λ_j exists, $H_{\lambda_j}(P)|c_j$. Therefore we conclude that $\neg PP|c_j$, and this in turn implies that $\neg$ wp(IF, X)|c_j for any predicate X. Since $\phi(c_k, \lambda_k)$ exists for $k < j$, $\forall\, k < j$: $PP|c_k$. By eliminating contradictory alternatives in $H_{\lambda_j}(P)|c_j$ we get $P|c_j$. We have thus proved all the clauses of T(P, c).

5. COMPLETENESS

To demonstrate completeness, we must show that $T(P, s) \Rightarrow wp(DO, P)|s$. To do this, we shall examine the branching structure of all possible executions starting with an arbitrary s for which $T(P, s)$ holds.

Suppose $T(P, s)$ holds.

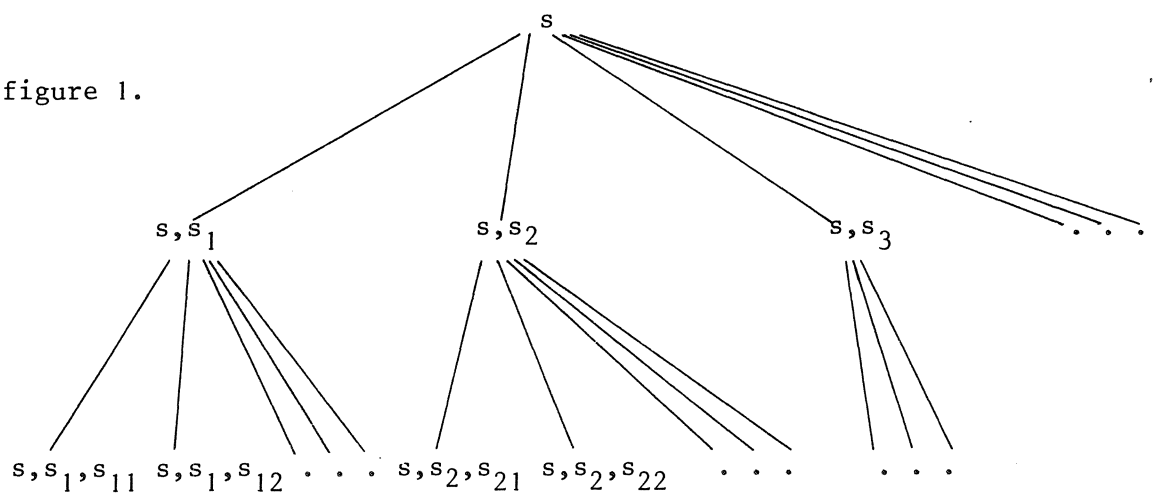
We shall use the notation 'i..j' for 'i,i+1,...,j', and '$c_{i..j}$' for 'c_i, c_{i+1}, ..., c_j'.

Consider all possible finite "execution segments" $c_{0..j}$, i.e., segments $c_{0..j}$ of states such that

$c_0 = s$, and
$\forall\, k: 0 \leq k < j: wp(IF, \underline{true})|c_k \wedge c_k\{IF\}c_{k+1} \wedge PP|c_k$.

(A segment is finite; a sequence need not be. Furthermore, segments terminate explicitly before they become undefined.) Define the tree-branch ordering $c_{0..j} > d_{0..k}$ iff $k > j$ and $c_{0..j} = d_{0..j}$. (Notice that the longer segments are called smaller.) Then the $c_{0..j}$ can be placed on a tree. The root is labelled with $c_{0..0}$, and the branches from the vertex labelled $c_{0..j}$ are labelled with the possible $d_{0..j+1}$ with $c_{0..j} > d_{0..j+1}$ (see figure 1). The ordering > then expresses relative position of the tree.

figure 1.



This tree is well-founded, since T(P, s).

To prove that wp(DO, P)|s, we shall prove that for each vertex $c_{0..j}$, $(\exists\ \nu\colon H_\nu(P)|c_j)$. This is done by bar induction [2] upwards on the tree structure.

Bar induction applies to trees for which every choice sequence of branches starting from the root terminates. The tree is thus well-founded, and satisfies the following induction principle:

If some property P holds at every leaf of a tree, and
if whenever P holds on all branches from some vertex X it also holds at X,
then P holds at the root, and, in fact at all vertexs of the tree.

In our case, it suffices to prove the following induction step:

for each vertex $c_{0..j}$,
if $(\forall\ d_{0..j+1} < c_{0..j}, \exists\ \nu : H_\nu(P)|d_{j+1})$,
then $\exists\ \nu\colon H_\nu(P)|c_j$.

No special starting step is needed, because at leaves of the tree the induction premiss is satisfied vacuously.

To prove this induction step, assume that for each $d_{0..j+1} < c_{0..j}$, $\exists \alpha_{d_{j+1}} : H_{\alpha_{d_{j+1}}}(P)|d_{j+1}$. Let β be the ordered sum of $\alpha_{d_{j+1}}$ over all d_{j+1} (strictly, over all $d_{0..j+1}$ such that $d_{0..j+1} < c_{0..j}$). For definiteness, we can well-order the d_{j+1} according to the countable enumeration of all states of the machine M. The requirement that the set of states of M is countable can be dropped for a nonconstructive proof; then we let β be the supremum of all the α's.

Let e be any execution sequence starting with the segment $e_{0..j} = c_{0..j}$. Then T(P, e); i.e., $\exists\ k\colon (P \wedge \neg PP|e_k) \wedge \forall\ 1 < k\colon PP \wedge wp(IF, \underline{true})|e_1$. Clearly $k \geq j$ because $c_{0..j}$ is an execution segment and $\neg PP|e_k$. There are

two cases, $k = j$ or $k > j$.

(1) If $k = j$, then $P \wedge \neg PP|c_j$, which is $H_0(P)|c_j$, and so finally $\exists \nu : H_\nu(P)|c_j$.

(2) If $k > j$, then $PP|c_j$ and $wp(IF, \underline{true})|c_j$. To show that $wp(IF, H_\beta(P))|c_j$, it will now suffice to show $\forall u: c_j\{IF\}u \Rightarrow H_\beta(P)|u$ because $wp(IF, --)$ is the correct predicate transformer for IF. Take any state u such that $c_j\{IF\}u$. Since $PP|c_j$, and $c_{0..j}$ is an execution segment, we know that $(c_{0..j}, u)$ is an execution segment; in fact, it is one of the $d_{0..j+1}$. Consider the α_u defined above, for which $H_{\alpha_u}(P)|u$. Then $\alpha_u < \beta$, so $H_\beta(P)|u$, and so we have proved $wp(IF, H_\beta(P))|c_j$. Therefore $\exists \mu : \beta+1 > \mu \wedge wp(IF, H_\mu(P))|c_j$, so $H_{\beta+1}(P)|c_j$, and finally $\exists \nu : H_\nu(P)|c_j$.

POSTSCRIPT

In Dijkstra's case of bounded nondeterminism, the completeness of his loop axiom does not need full bar-induction, but follows from the more restricted fan theorem.

REFERENCES

[1] DIJKSTRA, E.W., A Discipline of Programming, Prentice-Hall, 1976.

[2] TROELSTRA, A.S., Choice sequences, Oxford University Press 1977, ISBN 0 19 85313 X.

[3] MANNA, Z. & R. WALDINGER, Is "sometime" sometimes better that "always"? CACM 21, no 2, February, 1978, pp159-172.