On non-determinacy being bounded.

This is again a very formal chapter. In the chapter "The characterization of semantics." we have mentioned four properties that wp(S, R) , for any S considered as a function of R , should have if its interpretation as the weakest pre-condition for establishing R is to be feasible. (For non-deterministic mechanisms the fourth one of these was a direct consequence of the second one.)

In the next chapter "The semantic characterization of a programming language." we have given ways for constructing new predicate transformers, pointing out that these constructions should only lead to predicate transformers enjoying the aforementioned properties (i.e. if the whole exercise is to continue to make sense). For every basic statement ("skip", "abort" and the assignment statements) one has to verify that they enjoy the said properties; for every way of building up new statements from component statements (semicolon, alternative and repetitive constructs) one has to show that the resulting composite statements enjoy those properties as well, in which demonstration one may assume that the component statements enjoy them. We verified this up to and including the Law of the Excluded Miracle for the semicolon, leaving the rest of the verifications as an exercise to the reader. We leave it at that: in this chapter we shall prove a deeper property of our mechanisms, this time verifying it explicitly for the alternative and repetitive constructs as well. (And the structure of the latter verifications can be taken as an example for the omitted ones.)

Property 5. For any mechanism S and any infinite sequence of predicates C_0 , C_1 , C_2 , ... such that

for $r \geq 0$: $C_r \Rightarrow C_{r+1}$ for all states (1)

we have for all states

$$wp(S, (\underline{E}\ r: r \geq 0: C_r)) = (\underline{E}\ s: s \geq 0: wp(S, C_s)) \quad . \qquad (2)$$

For the statements "skip" and "abort" and for the assignment statements, the truth of (2) is a direct consequence of their definitions, assumption (1) not even being necessary. For the semicolon we derive

$$wp(\text{"S1; S2"}, (\underline{E}\ r: r \geq 0: C_r)) =$$

(by definition of the semantics of the semicolon)

$$wp(S1, wp(S2, (\underline{E}\ r: r \geq 0: C_r))) =$$

(because Property 5 is assumed to hold for S2)

$$wp(S1, (\underline{E}\ r': r' \geq 0: wp(S2, C_{r'}))) =$$

(because S2 is assumed to enjoy Property 2, so that $wp(S2, C_{r'}) \Rightarrow wp(S2, C_{r'+1})$ and S1 is assumed to enjoy Property 5)

$$(\underline{E}\ s: s \geq 0: wp(S1, wp(S2, C_s))) =$$

(by definition of the semantics of the semicolon)

$$(\underline{E}\ s: s \geq 0: wp(\text{"S1; S2"}, C_s))$$ QED.

For the alternative construct we prove (2) in two steps. The easy step is that the right-hand side of (2) implies its left-hand side. For, consider an arbitrary point X in state space, such that the right-hand side of (2) holds, i.e. there exists a non-negative value, s' say, such that in point X the relation $wp(S, C_{s'})$ holds. But because

$C_{s'} \Rightarrow (\underline{E}\ r: r \geq 0: C_r)$ and any S enjoys Property 2 , we conclude that

$$wp(S, (\underline{E}\ r: r \geq 0: C_r))$$

holds in point X as well. As X was an arbitrary state satisfying the right-hand side of (2) , the latter implies the left-hand side of (2). For this argument, antecedent (1) has not been used, but we need it for proving the implication in the other direction.

$$wp(IF, (\underline{E}\ r: r \geq 0: C_r)) =$$

(by definition of the semantics of the alternative construct)

$$BB\ \underline{\text{and}}\ (\underline{A}\ j: 1 \leq j \leq n: B_j \Rightarrow wp(SL_j, (\underline{E}\ r: r \geq 0: C_r))) =$$

(because the individual SL_j are assumed to enjoy Property 5)

$$BB\ \underline{\text{and}}\ (\underline{A}\ j: 1 \leq j \leq n: B_j \Rightarrow (\underline{E}\ s: s \geq 0: wp(SL_j, C_s))) \qquad . \quad (3)$$

Consider an arbitrary state X for which (3) is true, and let j' be a value for j such that $B_{j'}(X)$ = true ; then we have in point X

$$(\underline{E}\ s: s \geq 0: wp(SL_{j'}, C_s)) \qquad (4)$$

Because of (1) and the fact that $SL_{j'}$ enjoys Property 2, we conclude that

$$wp(SL_{j'}, C_s) \Rightarrow wp(SL_{j'}, C_{s+1})$$

and thus we conclude from (4) that in point X we also have

$$(\underline{E}\ s': s' \geq 0: (\underline{A}\ s: s \geq s': wp(SL_{j'}, C_s))) \qquad . \ (5)$$

Let $s' = s'(j')$ be the minimum value satisfying (5) . We now define smax as the maximum value of $s'(j')$ taken over the (at most n , and therefore the maximum exists!) values j' for which $B_{j'}(X)$ = true . In point X then holds on account of (3) and (5)

$$BB \ \underline{\text{and}} \ (\underline{A}\ j: 1 \le j \le n: B_j \Rightarrow wp(SL_j, C_{smax})) =$$

(by definition of the semantics of the alternative construct)

$$wp(IF, C_{smax}) \quad .$$

But the truth of the latter relation in state X implies there also

$$(\underline{E}\ s: s \ge 0: wp(IF, C_s)) \quad ;$$

but as X was an arbitrary state satisfying (3) , for S = IF the fact that the left-hand side of (2) implies its right-hand side as well, has been proved, and thus the alternative construct enjoys Property 5 as well. Note the essential role played by the antecedent (1) and the fact that a guarded command set is a <u>finite</u> set of guarded commands.

Property 5 is proved for the repetitive construct by mathematical induction.

Base: Property 5 holds for H_0 .

$$H_0(\underline{E}\ r: r \ge 0: C_r) =$$
$$(\underline{E}\ r: r \ge 0: C_r) \ \underline{\text{and}} \ \underline{\text{non}} \ BB =$$
$$(\underline{E}\ s: s \ge 0: C_s \ \underline{\text{and}} \ \underline{\text{non}} \ BB) =$$
$$(\underline{E}\ s: s \ge 0: H_0(C_s))$$

QED.

Induction step: From the assumption that Property 5 holds for H_k and H_0 follows that it holds for H_{k+1} .

$$H_{k+1}(\underline{E}\ r: r \ge 0: C_r) =$$

(by virtue of the definition of H_{k+1})

$$wp(IF, H_k(\underline{E}\ r: r \ge 0: C_r)) \ \underline{\text{or}} \ H_0(\underline{E}\ r: r \ge 0: C_r) =$$

(because Property 5 is assumed to hold for H_k and for H_0)

$\mathrm{wp}(IF, (\underline{E}\ r': r' \geq 0: H_k(C_{r'}))) \ \underline{or}\ (\underline{E}\ s: s \geq 0: H_0(C_s)) =$

(because Property 5 holds for the alternative construct and Property 2 is enjoyed by H_k)

$(\underline{E}\ s: s \geq 0: \mathrm{wp}(IF, H_k(C_s))) \ \underline{or}\ (\underline{E}\ s: s \geq 0: H_0(C_s)) =$

$(\underline{E}\ s: s \geq 0: \mathrm{wp}(IF, H_k(C_s)) \ \underline{or}\ H_0(C_s)) =$

(by virtue of the definition of H_{k+1})

$(\underline{E}\ s: s \geq 0: H_{k+1}(C_s))$. QED.

From base and induction step we conclude that Property 5 holds for all H_k , and hence

$\mathrm{wp}(DO, (\underline{E}\ r: r \geq 0: C_r)) =$

(by definition of the semantics of the repetitive construct)

$(\underline{E}\ k: k \geq 0: H_k(\underline{E}\ r: r \geq 0: C_r)) =$

(because Property 5 holds for all H_k)

$(\underline{E}\ k: k \geq 0: (\underline{E}\ s: s \geq 0: H_k(C_s))) =$

(because this expresses the existence of a (k, s)-pair)

$(\underline{E}\ s: s \geq 0: (\underline{E}\ k: k \geq 0: H_k(C_s))) =$

(by definition of the semantics of the repetitive construct)

$(\underline{E}\ s: s \geq 0: \mathrm{wp}(DO, C_s))$. QED.

* * *

Property 5 is of importance on account of the semantics of the repetitive construct

$$wp(DO, R) = (\underline{E}\ k: k \geq 0: H_k(R)) \qquad ;$$

such a pre-condition could be the post-condition for another statement. Because

for $k \geq 0$: $\qquad H_k(R) \Rightarrow H_{k+1}(R)$ for all states ,

--this is easily proved by mathematical induction-- the conditions under which Property 5 is relevant, are satisfied. We can, for instance, prove that in all initial states in which BB holds

$$\underline{do}\ B_1 \rightarrow SL_1 \,[]\, B_2 \rightarrow SL_2 \,[]\, \dots \,[]\, B_n \rightarrow SL_n\ \underline{od}$$

is equivalent to

$$\underline{if}\ B_1 \rightarrow SL_1 \,[]\, B_2 \rightarrow SL_2 \,[]\, \dots \,[]\, B_n \rightarrow SL_n\ \underline{fi};$$
$$\underline{do}\ B_1 \rightarrow SL_1 \,[]\, B_2 \rightarrow SL_2 \,[]\, \dots \,[]\, B_n \rightarrow SL_n\ \underline{od} \qquad .$$

(In initial states in which BB does not hold, the first program would have acted as "skip", the second one as "abort".) That is, we have to prove that

$$(BB\ \underline{and}\ wp(DO, R)) = (BB\ \underline{and}\ wp(IF, wp(DO, R))) \qquad .$$

$$BB\ \underline{and}\ wp(IF, wp(DO, R)) =$$

(on account of the semantics of the repetitive construct)

$$BB\ \underline{and}\ wp(IF, (\underline{E}\ k: k \geq 0: H_k(R))) =$$

(because Property 5 holds for IF)

$$BB\ \underline{and}\ (\underline{E}\ s: s \geq 0: wp(IF, H_s(R))) =$$

(because $(BB\ \underline{and}\ H_0(R)) = F$)

$$BB\ \underline{and}\ (\underline{E}\ s: s \geq 0: wp(IF, H_s(R))\ \underline{or}\ H_0(R)) =$$

(on account of the recurrence relation for the $H_k(R)$)

BB $\underline{\text{and}}$ $(\underline{E}\ s\colon s \geq 0\colon H_{s+1}(R))$ =

(because (BB $\underline{\text{and}}$ $H_0(R)) = F$)

BB $\underline{\text{and}}$ $(\underline{E}\ k\colon k \geq 0\colon H_k(R))$ =

(on account of the semantics of the repetitive construct)

BB $\underline{\text{and}}$ wp(DO, R) . QED.

* * *

Finally, we would like to draw attention to a very different consequence of the fact that all our mechanisms enjoy Property 5 . We could try to make the program S: "set x to any positive ~~integer~~" with the properties:

a) $wp(S, x > 0) = T$

b) $(\underline{A}\ s\colon s \geq 0\colon wp(S, 0 \leq x < s) = F)$.

Here property a) expresses the requirement that activation of S is guaranteed to terminate with x equal to some positive value, property b) expresses that S is a mechanism of unbounded non-determinacy, i.e. that no a priori upper bound for the final value of x can be given. For such a program S , we could, however, derive now:

$$\begin{aligned}
T &= wp(S, x > 0) \\
&= wp(S, (\underline{E}\ r\colon r \geq 0\colon 0 \leq x < r)) \\
&= (\underline{E}\ s\colon s \geq 0\colon wp(S, 0 \leq x < s)) \\
&= (\underline{E}\ s\colon s \geq 0\colon F) \\
&= F .
\end{aligned}$$

This, however, is a contradiction: for the mechanism S: "set x to any positive integer" <u>no</u> program exists!

As a result, any effort to write a program for "set x to any positive integer" must fail. For instance, we could consider:

```
go on:= true; x:= 1;
do go on → x:= x + 1
 ▯ go on → go on:= false
od                            .
```

This construct will continue to increase x as long as the first alternative is chosen; as soon as the second alternative has been chosen once, it terminates immediately. Upon termination x may indeed be "any positive integer" in the sense that we cannot think of a positive value X such that termination with $x = X$ is impossible. But termination is not guaranteed either! We can enforce termination: with N some large, positive constant we can write

```
go on:= true; x:= 1;
do go on and x < N → x:= x + 1
 ▯ go on → go on:= false
od
```

but then property b) is no longer satisfied.

The non-existence of a program for "set x to any positive integer" is reassuring in more than one sense. For, if such a program could exist, our definition of the semantics of the repetitive construct would have been subject to doubt, to say the least. With

```
S:      do x > 0 → x:= x - 1
         ▯ x < 0 → "set  x  to any positive integer"
        od
```

our formalism for the repetitive construct gives $wp(S, T) = (x \geq 0)$, while

I expect most of my readers to conclude that under the assumption of the existence of "set x to any positive integer" for $x < 0$ termination would be guaranteed as well. But then the interpretation of $wp(S, T)$ as the <u>weakest</u> pre-condition guaranteeing termination would no longer be justified. But when we substitute our first would-be implementation:

```
S:            do x > 0 → x:= x - 1
               ▯ x < 0 → go on:= true; x:= 1;
                         do go on → x:= x + 1
                          ▯ go on → go on:= false
                         od
              od
```

$wp(S, T) = (x \geq 0)$ is fully correct, both intuitively and formally.

The second reason for reassurance is of a rather different nature: a mechanism of unbounded non-determinacy but yet guaranteed to terminate would be able to make within a finite time a choice out of infinitely many possibilities: if such a mechanism could be formulated in our programming language, that very fact would present an unsurmountable barrier to the possibility of the implementation of that programming language.

<u>Acknowledgement</u>. I would like to express my great indebtness to John C.Reynolds for drawing my attention to the central role of Property 5 and to the fact that the non-existence of a mechanism "set x to any positive integer" is essential for the intuitive justification of the semantics of the repetitive construct. He is, of course, in no way to be held responsible for any of the above. (End of acknowledgement.)