The heuristics of a proof by Jan L.A. van de Snepscheut and Richard S. Bird

From JAN 161 "A little problem posed by R.S. Bird" d.d. 1989.12.11, I quote the statement of the problem:

"We are given a total function f that maps natural numbers to natural numbers. It has the peculiar property

\[ f(f(n)) < f(n+1) \quad (0) \]

for all \( n \geq 0 \). The problem is to show that \( f \) is the identity function."

Note For the sake of accuracy, in (0) Jan did not use the infix dot to denote function application. (End of Note.)

* * *

It is only natural, though superfluous, to check that the identity function of type \( \mathbb{N} \rightarrow \mathbb{N} \) satisfies (0); it does, since \( n < n+1 \).

It is more instructive to try to check that all givens are needed. Certainly we need (0), for it is not the case that any function of type \( \mathbb{N} \rightarrow \mathbb{N} \) is the identity function. Also the constraint that \( f \)
maps naturals to naturals is not void:
had it been integers to integers, \( f \) given by \( f \cdot n = n - 1 \), for all integer \( n \), would have satisfied (0). In other words, our proof has to contain a step that is valid for a natural domain, but invalid for an integer one. Because the naturals are well-founded whereas the integers are not, it is sweetly reasonable to propose:

\((\alpha)\) In our proof of

\[ f \cdot n = n \]  \hspace{1cm} (1)

for all \( n \geq 0 \), we shall try to use mathematical induction over the natural numbers.

In following (\( \alpha \)), it would be rash to conclude that (1) has to be our induction hypothesis; comparing (0) and (1), we observe similarities—comparisons of expressions with different depths of \( f \)-application—and a major difference: in the demonstrandum (1), the relational operator is \( = \), in the given (0) it is \( < \). In view of the obligation to conclude equality where inequalities are given, it is sweetly reasonable to propose

\((\beta)\) We shall try to construct a ping-pong argument in which
\[ f(n) \geq n \quad \text{and} \quad (2) \]
\[ f(n) \leq n \quad , \quad (3) \]

both for all \( n \geq 0 \), are dealt with separately.

In choosing which of the above two to prove by mathematical induction, the choice immediately falls on (2) since the base
\[ f(0) > 0 \]
is an immediate consequence of \( f \)'s type \( \mathbb{N} \rightarrow \mathbb{N} \). For the induction step we proceed:

\[
\begin{align*}
f(n+1) & \geq n + 1 \\
\{ \text{arithmetic} \} \hspace{1cm} & \\
f(n+1) & > n \\
\left\{ \text{ (0) } \right\} \hspace{1cm} & \\
f(f(n)) & \geq n
\end{align*}
\]

and here we are stuck, for this gives us no opportunity to appeal to the induction hypothesis \( f(n) \geq n \). So we had better backtrack and look for a stronger induction hypothesis.

Can we conclude a stronger base from the fact that \( f \) is of type \( \mathbb{N} \rightarrow \mathbb{N} \)? Well, we can take that fact itself; in order to be able to use the given that \( f \) is of type \( \mathbb{N} \rightarrow \mathbb{N} \), we formulate it without \( \mathbb{N} \):
\[ (\forall n : n \geq 0 \Rightarrow f.n \geq 0) \quad \text{(4)} \]

As asked for which induction hypothesis (4) acts as a proper base, any computing scientist that has designed invariants by replacing constants by variables will come up with the induction hypothesis
\[ (\forall n : n \geq j \Rightarrow f.n \geq j) \quad \text{(5)} \]

Remark: Concerning the decision to replace both 0's by \( j \), we point out
- induction hypothesis \((\forall n : n \geq j \Rightarrow f.n \geq 0)\) would lead to a trivial step
- induction hypothesis \((\forall n : n \geq 0 \Rightarrow f.n \geq j)\) leads to a step that cannot be proved
- (5) does the job since
\[ x \geq y \iff (\forall j : y \geq j \Rightarrow x \geq j) \quad \text{(6)} \]
(End of Remark.)

There is a totally different reason why it is more attractive to prove (5) inductively for all \( j \) than it is to prove (2) inductively for all \( n \). The reason is that (2) contains the induction variable as argument of (the unknown) function \( f \), whereas (5) contains the induction variable \( j \) in perfectly manageable positions.
The base having been taken care of by (4), we now turn to the induction step to prove (5) inductively over \( j \). To this end we observe for any natural \( n \) and \( j \)

\[
f(n) \geq j + 1
\]

= \{ arithmetic \}

\[
f(n) > j
\]

\[
\forall \{ (0) \ text{ with } n := n - 1 \}, \ i.e. \ f(n) > f(f(n)) \text{ for } n \geq 1\}

n \geq 1 \land f(n - 1) > j

\[
\forall \{ \text{ ex hyp.: (5) with } n := f(n - 1) \}

n \geq 1 \land n < j

\[
= \{ j > 0 \}

n \geq j + 1
\]

Thus we have dealt with ping, i.e. (2).

\[
* \quad * \quad *
\]

For pong we observe that mathematical induction over \( n \) is (as yet) not indicated because the base is not obvious. To relate (3) to (0):

\[
f(f(n)) < f(n + 1)
\]

we rewrite (3) as

\[
f(n) < n + 1
\]


i.e. the given (0) has at both sides up < an f-application more than the demonstrandum (3). Now this looks very similar to monotonicity!

The usual way of expressing that \( f \) is monotonic is

\[
\langle \forall x, y :: x \geq y \implies f.x \geq f.y \rangle,
\]

which, by taking the contrapositive, yields

\[
\langle \forall x, y :: x < y \iff f.x < f.y \rangle \quad . \tag{7}
\]

Under the assumption of \( f \)'s monotonicity, the demonstration of (3) is a walkover: we observe for any \( n \)

\[
f.n \leq n
\]

\[
\text{\{arithmetic\}}
\]

\[
f.n < n+1
\]

\[
\iff \quad \text{\{7\} with } x, y := f.n, \ n+1 \}
\]

\[
f.(f.n) < f.(n+1)
\]

\[
\text{\{0\}}
\]

\[
\text{true,}
\]

but this still leaves us with the obligation to demonstrate that \( f \) is monotonic. (Note that the assumption was safe in the sense that the identity function is, indeed, monotonic.)
Monotonicity of a function $f$ on naturals can be expressed by an expression like (7) which quantifies over 2 dummy's, or by

$$ \langle \forall x :: f.x \leq f.(x+1) \rangle \quad , \quad (8) $$

which quantifies over a single dummy. The latter is usually the most convenient form to demonstrate monotonicity; the former, which includes the consequences of transitivity, is the most convenient characterization for the exploitation of monotonicity.

Remark The above paragraph covers a standard ingredient of the intellectual baggage of professional reasoners about sorting.

(End of Remark.)

In order to demonstrate the monotonicity of $f$, we prove (8) by observing for any natural $x$

$$ f.x \leq f.(f.x) \quad \{ (2) \text{ with } n := f.x \} $$

$$ f.(f.x) \leq f.(x+1) \quad \{ (0) \text{ with } n := x \} $$

which concludes pong, and thus the whole proof. * * *
JAN 161-2 contains Bird's proof of pong. It is about 8 steps long, using that f is increasing rather than just monotonic. I had not set out to simplify their argument, my only intention was to provide the heuristics. The subsequent simplification was a pleasant surprise.

I am grateful to the members of the ATAC with whom this problem has been discussed and whose comments contributed to the design of the above heuristics.

Austin, 6 April 1994

prof. dr. Edsger W. Dijkstra
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712-1188
USA