A conversion routine revisited

In the following, \( c(k:1\leq k) \) is a sequence of integers satisfying

(0) \[ c_k \geq 2 \quad \text{for all } k \geq 1, \]

but, for the time being, only \( c_k \neq 0 \) is relevant: it allows us to define \( w(k:0 \leq k) \) by

(1) \[ w_0 = 1 \]

(2) \[ w_{k+1} = c_k \cdot w_k \quad \text{for all } k \geq 1. \]

We now consider the following program defining the sequence \( d(k:1\leq k) \) in terms of the real number \( R \)

\[
\begin{align*}
n, f &:= 0, R \quad \{P0\} \\
; \text{ do true } \Rightarrow n &:= n + 1 \quad \{P1\} \\
; \quad d_n, f := \text{INFRAS.} (f \cdot c_n) \quad \{P0\}
\end{align*}
\]

where, for the time being, our only knowledge about the pair-valued function \( \text{INFRAS.} \) of a real argument is

(3) \[ x, y = \text{INFRAS.} z \quad \Rightarrow \quad x + y = z \]

The above suffices to demonstrate the invariance of \( P0: \ R = (\Sigma k: 0 < k \land k \leq n; d_k \cdot w_k) + f \cdot w_n \)

(leaving the fact that \( n \) is natural understood).
In order to verify that the initialization establishes \( P_0 \) we observe

\[
wp. \quad "n, f := 0, R". \quad P_0
\]

\[
= \{ \text{axiom of assignment, definition of } P_0 \}
\]

\[
R = (\Sigma k: 0 < k \land k \neq 0: d_k \ast w_k) + R \ast w_0
\]

\[
= \{ \text{definition of empty sum, (1)} \}
\]

\[
R = 0 + R \ast 1
\]

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

\[
\text{true}
\]

In order to verify the invariance of \( P_0 \) we observe – after introducing the abbreviation \( S \):

\[
S = d_n, f := \text{INTFRA.} (f \ast c_n)
\]

for \( n \geq 1 \)

\[
wp. S. \quad P_0
\]

\[
= \{ \text{definition of } P_0 \}
\]

\[
w_0. S. \quad (R = (\Sigma k: 0 < k \land k \neq n: d_k \ast w_k) + f \ast w_n)
\]

\[
= \{ \text{splitting the range, as } n \geq 1 \}
\]

\[
w_0. S. \quad (R = (\Sigma k: 0 < k \land k \leq n-1: d_k \ast w_k) + (d_n + f) \ast w_n)
\]

\[
= \{ \text{axiom of assignment, definition of } S, (3) \}
\]

\[
(\Sigma k: 0 < k \land k \leq n-1: d_k \ast w_k) + f \ast c_n \ast w_n
\]

\[
= \{ (2) \text{ with } k := n \}
\]

\[
(\Sigma k: 0 < k \land k \leq n-1: d_k \ast w_k) + f \ast w_{n-1}
\]

\[
= \{ \text{definition of } P_0 \text{ and of substitution} \}
\]

\[
(n := n-1). \quad P_0
\]

\[
= \{ \text{our nomenclature} \}
\]

\[
P_1
\]
Since \( \text{wp.""}_n \cdot \text{P}_1 \implies \text{P}_0 \), the invariance of \( \text{P}_0 \) has now been established.

\[ * * * \]

Our next target is

\[ (4) \quad R = (\Sigma k: 0 < k: d_k \ast w_k) \]

a relation which follows from \( \text{P}_0 \) if

\[ (5) \quad \lim_{k \to \infty} f \ast w_k = 0. \]

In view of (0) - which now becomes relevant - and (2), \( \lim_{k \to \infty} w_k = 0 \), so (5) follows if \( f \) is bounded.

A look at the program tells us that boundedness of \( f \) depends on the function \( \text{INTFRA} \), about which, up till now, only (3) has been given. For given \( z \), the relation \( x + y = z \) of (3)'s consequent can be satisfied in many ways: the one extreme, viz. \( x = 0 \land y = z \), leads to unbounded \( f \); the other extreme, viz. \( x = z \land y = 0 \) leads, instead of to the general (4), to \( R = d_i \ast w_i \), which is not interesting. So we choose something in between.

The left element of \( \text{INTFRA} \cdot z \) is constrained by

\[ (6) \quad x, y = \text{INTFRA} \cdot z \implies x \text{ is integer }, \]

which allows us to restrict \( y \) to a unit interval. For
the constraint on \( y \) we consider the following two alternatives:

(i) \( x,y = \text{INTRA.2} \Rightarrow 0 \leq y \land y < 1 \) and

(ii) \( x,y = \text{INTRA.2} \Rightarrow 0 < y \land y \leq 1 \).

We now restrict our investigation of these alternatives to \( 0 < R \land R \leq 1 \); in the following it becomes relevant that \( c \) is a sequence of integers.

Alternative (i) leads to the invariance of

\[ 0 \leq f \land f < 1 \]

alternative (ii) leads to the invariance of

\[ 0 < f \land f \leq 1 \]

In both cases we have for all \( k \)

\[ 0 \leq d_k \land d_k < c_k \]

The minimum value \( 0 \) and the maximum value \( c_k - 1 \) can in both cases occur for \( d_k \).

For a given sequence \( c \), we call an R-value sensitive if its corresponding sequence \( d \) depends on whether (i) or (ii) has been chosen, otherwise we call it insensitive. We shall see below that for any sequence \( c \), values of both types exist.
Because of $0 < R \land R < 1$, the initialization of the program establishes

(7) $0 < f \land f < 1$

If with one of the alternatives for INFRA, $f \cdot c_n$ is never integer, (7) is an invariant and the computation is indistinguishable from the one with the same $R$ but the other alternative for INFRA: $R$ is insensitive.

(E.g. $R = \frac{1}{2}$ for $c_k = 3$.)

We now investigate the case in which, sooner or later, the repeatable statement falsifies (7). That occurs when, with $f$ satisfying (7), $f \cdot c_n$ is integer, let then

$f \cdot c_n, n = q, N$

Because $f$ satisfies (7), $q$ satisfies

$1 < q \land q < c_n$

The rest of the computation depends on which alternative has been chosen for INFRA. With (i), the assignment is effectively

$a_N, f := q, 0$

and, $f=0$ being stable with (i), the program defines

$(A)_k: N < k: a_k = 0)$
i.e. sequence $d$ ends with $d_N > 0$, followed by "the minimum tail".

With (ii), the assignment is effectively

$$d_N, f \equiv q-1, 1,$$

and, $f=1$ being stable with (ii), the program defines

$$(\forall k: N < k: d_k = c_k - 1),$$

i.e. sequence $d$ ends with $d_N < c_N - 1$, followed by "the maximum tail".

We conclude that an $R$ giving rise to an integer $f \times c_n$ is sensitive.

**Remark** With, for some positive $N$, $R = w_N$, $R$ is sensitive. In view of (4), the above analysis tells us

$$(8) \quad w_N = (\sum k: N < k: (c_k - 1) \times w_k) \quad .$$

(End of Remark.)

If $R$ and $d$ satisfy (4), we say that "$d$ is a representation of $R$", and in that terminology we just found that insensitive values have at least 1 representation and sensitive values have at least 2 representations. Under the constraint

$$(\forall k: 0 < k: 0 \leq d_k \land d_k < c_k)$$
values don't have more representations. The argument relies primarily on the monotonicity of addition. We ask ourselves under what circumstances two different sequences can represent the same value. Let $X$ be the common prefix of the two sequences, in the one followed by "p tail0" and in the other by "q tail1", with $p < q$.

We now consider the following sequences and values:

\[
\begin{align*}
X & \quad p \text{ tail0} & R_0 \\
X & \quad p \text{ maximum tail} & R_1 \\
X & \quad p+1 \text{ minimum tail} & R_2 \\
X & \quad q \text{ minimum tail} & R_3 \\
X & \quad q \text{ tail1} & R_4
\end{align*}
\]

Because in tail0 $d_k < c_k$, $R_0 \leq R_1$ and $R_0 = R_1$ \(\Rightarrow\) tail0 = maximum tail. Because of (8), $R_1 = R_2$. Because of $p < q$, $R_2 \leq R_3$; $R_2 = R_3 \Rightarrow p + 1 = q$. Because in tail1 $0 \leq d_k$, $R_3 \leq R_4$ and $R_3 = R_4$ \(\Rightarrow\) minimum tail = tail1. Because the sequence of $R$-values is ascending, $R_0 = R_4$ implies that they are all equal, hence:

- tail0 = maximum tail
- $p + 1 = q$
- tail1 = minimum tail,

and the two sequences representing the sensitive value are the ones generated according to (ii) and (i) respectively. Also: a sequence not ending in an extreme tail is insensitive.
In other words, the shape of the sequence—i.e. whether or not it has an extreme tail—determines whether the value represented by the sequence is sensitive or not.

Each sensitive \( R \) is rational. The fraction \( \frac{1}{q} \) is sensitive provided, for some \( N \), \( q \) divides the product of the first \( N \) \( c \)-values. Hence, each rational \( R \) is sensitive provided

\[
(9) \quad (A_q : 0 < q : (\mathbb{E}N :: q \text{ divides } (\prod k : 0 < k \land k < N : c_k )))
\]

Combining the two observations, we conclude

\[
(9) \Rightarrow (R \text{ is sensitive}) \equiv (R \text{ is rational})
\]

Condition (9) is clearly satisfied for \( c_k = k+1 \). Hence, \( e \), the basis of the natural logarithm, is irrational.

Nuenen, 6 January 1991

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