Representation Conversion Revisited

0 Representation Conversion

In Chapter 8 of "Programming by Calculation" I have constructed a functional program for the conversion of binary numbers into ternary numbers. This resulted in a function *c23* mapping binary lists to ternary lists and an auxiliary function f. We repeat the definitions of these functions here – in which, for technical reasons, I have renamed parameter s to r –:

$$\begin{array}{lll} & c23\cdot[\,] & = & [\,] \\ \& & c23\cdot(r \triangleleft b) & = & f\cdot b\cdot(c23\cdot r) \end{array}$$

$$\begin{array}{lll} & f\cdot b\cdot[\,] & = & [b] \\ \& & f\cdot b\cdot(t \triangleleft c) & = & f\cdot(h\,\mathsf{div}\,3)\cdot t \triangleleft h\,\mathsf{mod}\,3 \\ & & & \mathsf{whr}\ \ h = 2*c+b\ \ \mathsf{end} \end{array}$$

We now apply the tail-recursion theorems to transform these recursive definitions into a sequential program for the conversion of binary into ternary numbers. Moreover, apart from applying these theorems, we will also have to choose suitable representations of the lists involved.

1 Transformation into Tail Recursion

We start with *c23*. This is a function on lists and the recursion pattern of its definition matches the tail-recursion theorem (ii) for lists. Here function f corresponds to binary operator $\oplus$ in the theorem: we may define $\oplus$ by, for binary digit b and ternary list t:

$$b \oplus t \ = \ f\cdot b\cdot t \ .$$

The tail-recursion theorem for lists now yields that:

$$c23\cdot r \ = \ G\cdot[\,]\cdot r \ ,$$

provided we define function G by:

$$\begin{array}{lll} & G\cdot t\cdot[\,] & = & t \\ \& & G\cdot t\cdot(b \triangleright r) & = & G\cdot(f\cdot b\cdot t)\cdot r \end{array}$$

Next we consider the implementation of function f, that is, the implementation of the substitution $t := f\cdot b\cdot t$.

Observing that $(\triangleleft\ h\,\mathsf{mod}\,3)$ is equivalent to $(+\!\!+\ [h\,\mathsf{mod}\,3])$, such that the non-associative operator $\triangleleft$ is replaced by the associative $+\!\!+$, we prepare for application of the first tail-recursion theorem, by reformulating f's definition as follows:

$$\begin{array}{lll} & f\cdot b\cdot[\,] & = & [b] \\ \& & f\cdot b\cdot(t \triangleleft c) & = & f\cdot(h\,\mathsf{div}\,3)\cdot t +\!\!+ [h\,\mathsf{mod}\,3] \\ & & & \mathsf{whr}\ \ h = 2*c+b\ \ \mathsf{end} \end{array}$$

Now we can easily generalize f to a function f, F, with this specification:

$$F{\cdot}s{\cdot}b{\cdot}t \;=\; f{\cdot}b{\cdot}t +\!\!+ s \quad .$$

Now we have $f{\cdot}b{\cdot}t = F{\cdot}[\,]{\cdot}b{\cdot}t$, and thus we obtain for *c23*:

$$\begin{array}{lll} & c23{\cdot}r & = \; G{\cdot}[\,]{\cdot}r \\ \& & G{\cdot}t{\cdot}[\,] & = \; t \\ \& & G{\cdot}t{\cdot}(b \triangleright r) & = \; G{\cdot}(F{\cdot}[\,]{\cdot}b{\cdot}t){\cdot}r \\ \& & F{\cdot}s{\cdot}b{\cdot}[\,] & = \; b \triangleright s \\ \& & F{\cdot}s{\cdot}b{\cdot}(t \triangleleft c) & = \; F{\cdot}(h\mathsf{mod}3 \triangleright s){\cdot}(h\mathsf{div}3){\cdot}t \\ & & & \quad \mathsf{whr}\;\; h = 2*c+b \;\; \mathsf{end} \end{array}$$

2 Array segments representing lists

A (finite) list can be represented as an array segment by enumeration of the list's elements. This requires, of course, that the length of that array segment equals the length of the list.

So, for list s of length n we say that array segment $x[\,p\,..\,q\,)$ –which has length $q-p$, for $p \le q$– represents s if (and only if):

$$q-p \;=\; n \;\;\wedge\;\; (\forall i : 0 \le i < n : s{\cdot}i = x[\,p+i\,]) \quad ,$$

which, in what follows, we will abbreviate to:

$$s \;=\; x[\,p\,..\,q\,) \quad .$$

Notice that in this representation we now have that:

0. The condition $s = [\,]$ amounts to $p = q$.
1. The operation $s := b \triangleright s$ amounts to $p := p-1 \;;\; x[\,p\,] := b$.
2. The operation $s := s \triangleleft b$ amounts to $x[\,q\,] := b \;;\; q := q+1$.
3. The expression $s{\cdot}0$ equals $x[\,p\,]$.
4. The operation $s := tl{\cdot}s$ amounts to $p := p+1$, provided s is nonempty, i.e.: $p < q$.
5. Let list t and integer b be such that $s = t \triangleleft b$. Then, the operation $s, c := t, b$ amounts to $q := q-1 \;;\; c := x[\,q\,]$.

3 Implementation as Sequential programs

From the tail-recursive definition for function G we obtain the following sequential program for the computation of $c23{\cdot}r0$, for some given, fixed list $r0$, of length N. In the following program the list parameter r is represented by an array segment $x[\,p\,..\,N\,)$, according to this representation invariant:

$P0$: $\quad 0 \le p \le N \ \wedge \ r = x[p\,..\,N)$.

That is, the elements of r are stored consecutively in array segment $x[p\,..\,N)$. We assume that the initial value $x[0\,..\,N)$ represents the whole list $r0$. Variable t is assumed to be a list; its representation and the implementation of $t := F{\cdot}[\,]{\cdot}b{\cdot}t$ will be taken care of in the next step.

$\{\ r0 = x[0\,..\,N)\ \}$
$t, p := [\,], 0$
; $\{$ invariant: $\ P0 \ \wedge \ c23{\cdot}r0 = G{\cdot}t{\cdot}r\ \}$
do $p \neq N \ \rightarrow \ b := x[p]\ ;\ p := p+1$
$\quad ;\ \{\ 1 \le p \le N \ \wedge \ r = x[p\,..\,N)\ \}$
$\quad\ t := F{\cdot}[\,]{\cdot}b{\cdot}t$
od
$\{\ t = c23{\cdot}r0\ \}$

In terms of operations on lists, the assignment $t := F{\cdot}[\,]{\cdot}b{\cdot}t$ can be implemented as follows as a sequential program, based, of course, on the tail-recursive definition of function F. Here we need auxiliary names, $b0$ and $t0$ (say), in order to be able to distinguish the initial values of the variables from their actual values:

$\{\ 1 \le p \le N \ \wedge \ r = x[p\,..\,N)\ \}$
$s, b0, t0 := [\,], b, t$
; $\{$ invariant: $\ F{\cdot}[\,]{\cdot}b0{\cdot}t0 = F{\cdot}s{\cdot}b{\cdot}t\ \}$
do $t \neq [\,] \ \rightarrow \quad u, c :\ t = u \triangleleft c$
$\quad ;\ h := 2 * c + b$
$\quad ;\ s, b, t := h\mathsf{mod}3 \triangleright s\,,\ h\mathsf{div}3\,,\ u$
od
; $\{\ b \triangleright s = F{\cdot}[\,]{\cdot}b0{\cdot}t0\ \}$
$t := b \triangleright s$
$\{\ t = F{\cdot}[\,]{\cdot}b0{\cdot}t0\ \}$

* * *

Now we are left with the choice of a suitable representation of list t, and of the local list variable s that occurs in the above repetition. For this purpose we introduce an array y and an additional variable q, in which list t will be represented. Together with the (existing) representation of r this yields the following representation invariant for the main repetition – that is, the repetition for the computation of G–:

$P1$: $\quad 0 \le p \le N \ \wedge \ r = x[p\,..\,N) \ \wedge \ 0 \le q \ \wedge \ t = y[0\,..\,q)$.

The initialization $t := [\,]$ now simply boils down to $q := 0$.

List variable s is a local variable of the computation of function F. It can be conveniently represented as a segment in the very same array y, thus:

$s \ = \ y[q{+}1\,..\,q0{+}1)$,

where $q0$ is a name for the initial value of q –that is, $y[\,0\,..\,q0\,)$ represents $t0$–.

Thus s is represented "conveniently" indeed, because it leaves exactly one array element "unused", namely $y[q]$. Upon termination of the repetition for the computation of F we have $t=[\,]$, that is, $q=0$, and, hence, $s=y[\,1\,..\,q0+1\,)$. In this case the "unused" array element is $y[0]$; consequently, the final assignment $t := b \triangleright s$ can now be implemented in $\mathcal{O}(1)$ time, without the obligation to copy one array segment into another; this boils down to: $y[0] := b \;;\; q := q0+1$.

Thus we obtain the following sequential program for the implementation of $t := F{\cdot}[\,]{\cdot}b{\cdot}t$:

$$
\begin{array}{l}
\{\ 0\le q\ \wedge\ t=y[\,0\,..\,q\,)\ \wedge\ 1\le p\le N\ \wedge\ r=x[\,p\,..\,N\,)\ \} \\
\quad q0,b0,t0 := q,b,t \\
;\ \{\ \mathsf{invariant}\colon\ F{\cdot}[\,]{\cdot}b0{\cdot}t0 = F{\cdot}s{\cdot}b{\cdot}t\ \wedge\ 0\le q\le q0\ \wedge\ t=y[\,0\,..\,q\,)\ \wedge\ s=y[\,q{+}1\,..\,q0{+}1\,)\ \} \\
\quad \mathsf{do}\ q\neq 0\ \rightarrow\ \ q := q-1\ ;\ c := y[q] \\
\qquad\qquad\qquad ;\ h := 2*c+b \\
\qquad\qquad\qquad ;\ y[q+1]\,,b\ :=\ h\mathsf{mod}3\,,\ h\mathsf{div}3 \\
\quad \mathsf{od} \\
;\ \{\ b\triangleright s\ =\ F{\cdot}[\,]{\cdot}b0{\cdot}t0\ \wedge\ t=[\,]\ \wedge\ s=y[\,1\,..\,q0{+}1\,)\ \} \\
\quad y[0] := b\ ;\ q := q0+1 \\
\quad \{\ t\ =\ F{\cdot}[\,]{\cdot}b0{\cdot}t0\ \wedge\ t=y[\,0\,..\,q\,)\ \wedge\ r=x[\,p\,..\,N\,)\ \}
\end{array}
$$

4 A completely in situ solution

Notice that the net effect, on variable q, of the two assignments $q0 := q$ and $q := q0+1$ is equivalent to $q := q+1$. That is, the substitution $t := F{\cdot}[\,]{\cdot}b{\cdot}t$ effectively increases q –which is the length of list t– by exactly 1. Because, initially we have $p=q$ and because each step of the main repetition also increases p by exactly 1, it follows that $p=q$ is an invariant of the main repetition too.
This means, however, that variable q becomes superfluous, as its role can be taken over by p. Hence, invariant *P1* can now be rephrased as follows:

$$P2\colon\quad 0\le p\le N\ \wedge\ r=x[\,p\,..\,N\,)\ \wedge\ t=y[\,0\,..\,p\,)\ .$$

Now, however, we observe that:

- A precondition of (the program for) $t := F{\cdot}[\,]{\cdot}b{\cdot}t$ is $r=x[\,p\,..\,N\,)$ and the elements of array segment $x[\,0\,..\,p\,)$ are no longer relevant;
- In the program for $t := F{\cdot}[\,]{\cdot}b{\cdot}t$ segment $y[\,0\,..\,p\,)$ is the only part of array y that is actually used.

Therefore, the two arrays x and y can be combined. That is, the values stored in $y[\,0\,..\,p\,)$ can equally well be stored in $x[\,0\,..\,p\,)$. As a result we obtain a completely in situ solution, with the property that array x initially contains the binary representation of the number and eventually it will contain the ternary representation of the same number.

That is, the representation invariant for the main repetition becomes:

$$P3\colon\quad 0\le p\le N\ \wedge\ r=x[\,p\,..\,N\,)\ \wedge\ t=x[\,0\,..\,p\,)\ .$$

Carrying through these changes, combining the two program fragments, omitting the annotation, and cleaning up the code, we obtain the following complete program for binary to ternary number conversion:

```
  p := 0
; do p ≠ N → b := x[p] ; p := p+1 ; q := p−1
            ; do q ≠ 0 → q := q−1 ; c := x[q]
                       ; h := 2 ∗ c + b
                       ; x[q+1], b := h mod 3, h div 3
              od
            ; x[0] := b ; q := p
  od
```

5 Exercises

0. The ternary representation resulting from *c23·s* not necessarily is the *shortest* possible: it may contain redundant leading zeroes. Modify the functional programs for *c23*, and subsequently also the sequential program, in such a way that *c23·s* equals the shortest possible ternary list that satisfies the specification.

1. The property that the ternary representation of a number needs not be longer than its binary representation is due to the fact that $2 < 3$. When we convert numbers from a representation in a larger base to one in a smaller base the resulting representation may be longer than the input. Design a sequential program for the conversion from ternary to binary, preferably still in situ. How do you choose the list representations in this case?

Eindhoven, 9 june 2013

Rob R. Hoogerwoord
department of mathematics and computing science
Eindhoven University of Technology
postbus 513
5600 MB Eindhoven