BIT 25 (1985), 327–334

TOWERS OF HANOI PROBLEMS: DERIVING ITERATIVE SOLUTIONS BY PROGRAM TRANSFORMATIONS

ALBERTO PETTOROSSI

Istituto di Analisi dei Sistemi ed Informatica del C.N.R., Viale Manzoni 30, 00185 Roma, Italy

Abstract.

We present the iterative solutions of the Towers of Hanoi problems (standard, cyclic, and generalized) using the program transformation methodology of Burstall-Darlington. We derive algorithms with minimal time × space requirements. Their correctness proofs are trivial, as usual when applying the program transformation technique.

Keywords: Towers of Hanoi problems, recursion vs. iteration, cyclic motion, program transformation.

1. Introduction.

In the *standard Towers of Hanoi problem* there are three pegs: A, B and C. It is required to move a pile of n disks, with $n \geqq 0$, from peg A to peg B using the third peg C as an auxiliary peg.

There exists a very elegant recursive solution for the standard problem [4, 7] and it is also possible to obtain simple iterative solutions [2, 7]. As far as simplicity is concerned, some authors have been in favour of the recursive solution, others of the iterative ones.

Some variants of the standard problem have been recently proposed in the literature [1, 5] and for them the recursive solutions seem to be simpler than the iterative ones.

Subsequent efforts have been devoted to the discovery of iterative algorithms for those variants in order to show that the iterative approach is, so to speak, as simple as the recursive one, at least for that class of problems [6, 11].

However the iterative solutions were hard to find [11].

We show that, using the program transformation technique of Burstall-Darlington [3], we can easily derive the iterative algorithms from the recursive ones, for the standard Towers of Hanoi problem and its cyclic and generalized variants.

Furthermore these solutions are very efficient and their correctness can be proved with small effort (unlike the iterative solutions presented in [6, 11]).

Received April 1984. Revised November 1984.

2. Deriving an iterative solution for the standard Towers of Hanoi problem.

There are 3 pegs A, B and C and n disks of different sizes which are stacked as a tower on peg A with smaller disks on top of larger disks (the largest disk is at the bottom). We are required to move the tower on peg A to peg B by moving only one disk at the time and placing smaller disks on top of larger ones only.

Moves $= \{AB, BC, CA, BA, CB, AC, \text{skip}\}$ denotes the set of possible moves. For instance, AB denotes the move that takes the top disk from peg A to the top of peg B, skip denotes the "empty move".

:: denotes concatenation of moves. For instance, $AB :: CB$ denotes the move AB followed by the move CB, and $AB ::$ skip$:: CB ::$ skip is equal to $AB :: CB$.

Moves* denotes the monoid on the set Moves with the operation ::.

The following recursive function $f(n, A, B, C)$ solves the standard Towers of Hanoi problem, for n disks to be moved from peg A to peg B by using the third peg C as auxiliary peg.

$$
\begin{aligned}
&f : \text{number} \times \text{peg} \times \text{peg} \times \text{peg} \to \text{Moves}^* \\
&f(0, A, B, C) = \text{skip} \\
&f(n+1, A, B, C) = f(n, A, C, B) :: AB :: f(n, C, B, A)
\end{aligned}
$$

In order to obtain an iterative program which computes $f(n, A, B, C)$ we will proceed in two steps.

Step i): *transformation from general recursion to linear recurison.*

In order to do so we perform the symbolic evaluation of $f(n, A, B, C)$ as shown in fig. 1, where $g \bullet\!-\!\bullet h$ denotes that the evaluation of the function g (written above) requires the one of h (written below). From fig. 1 we see that the triple of functions: $\langle f(n-2, A, B, C),\ f(n-2, B, C, A),\ f(n-2, C, A, B)\rangle$ divides the symbolic evaluation graph in an upper part and a lower part and it can be expressed in terms of itself with $n-2$ replaced by $n-4$.

The explicit definition of the function:

$t(n) = \langle f(n, A, B, C), f(n, B, C, A), f(n, C, A, B)\rangle$ is:

$t(0) = \langle \text{skip}, \text{skip}, \text{skip}\rangle$; $\quad t(1) = \langle AB, BC, CA\rangle$;

$$
\begin{aligned}
t(k+2) = \langle & u :: AC :: v :: AB :: w :: CB :: u, \\
& v :: BA :: w :: BC :: u :: AC :: v, \\
& w :: CB :: u :: CA :: v :: BA :: w\rangle \quad \textbf{where} \quad \langle u, v, w\rangle = t(k).
\end{aligned}
$$

(The derivation of $t(n)$ is an application of the "tupling strategy" [9]). We can express $f(n, A, B, C)$ in terms of $t(n, A, B, C)$ as follows:

$f(0, A, B, C) = \text{skip}$
$f(1, A, B, C) = AB$
$f(n+2, A, B, C) = u :: AC :: v :: AB :: w :: CB :: \text{u}$ **where** $\langle u, v, w\rangle = t(n)$.

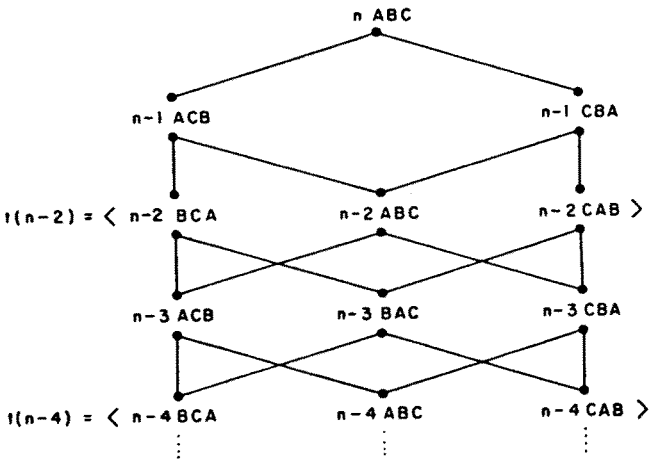

Fig. 1. The symbolic evaluation of $f(n, A, B, C)$; f is omitted above.

Step ii): *transformation from linear recursion to iteration.*

We use the simple equivalence [10] of the two schemata:

$$t(0) = a; \qquad t(1) = b; \qquad t(k+2) = c(t(k))$$

and $\{k = K \geqq 0\}$

if even(k) **then** $T := a$ **else** $T := b$;
while $k > 1$ **do** $T := c(T)$; $k := k-2$ **od**
$\{T = t(K)\}$

Standard Hanoi Program $\{n \geqq 0\}$

if $n = 0$ **then** $F :=$ skip **elseif** $n = 1$ **then** $F := AB$ **else**
begin $n := n-2$;
if even(n) **then** $T := \langle$skip, skip, skip$\rangle$ **else** $T := \langle AB, BC, CA\rangle$;
while $n > 1$ **do** $T := \langle T1 :: AC :: T2 :: AB :: T3 :: CB :: T1,$
$T2 :: BA :: T3 :: BC :: T1 :: AC :: T2,$
$T3 :: CB :: T1 :: CA :: T2 :: BA :: T3\rangle$;
$n := n-2$ **od** $\{T = t(n-2)\}$
$F := T1 :: AC :: T2 :: AB :: T3 :: CB :: T1$
end $\{F = f(n, A, B, C)\}$

where Ti denotes the ith component of the tuple T, for $i = 1, 2, 3$.

The iterative solution we derived is linear in time and exponential in space.

With n disks the loop is performed $O(n)$ times and the length of the sequences of moves in T is $O(\exp n)$. Therefore its time × space requirement is equal to the time × space requirement of the iterative solutions in [2, 7].

In what follows we will apply the same derivation steps (i.e. the application of the tupling strategy followed by the removal of the linear recursion) for obtaining the iterative algorithms for two proposed variants of Towers of Hanoi problem. These variants were claimed to have iterative solutions which were difficult to find [1, 6].

3. Deriving an iterative solution for the cyclic Towers of Hanoi problem.

In the cyclic Towers of Hanoi problem we consider the 3 pegs as being arranged cyclically and we allow moves in the clockwise (or anticlockwise) direction only. In [1] the following mutually recursive procedures are given for solving the cyclic Towers of Hanoi problem:

$$\begin{aligned}
&\text{clock} : \text{number} \times \text{peg} \times \text{peg} \times \text{peg} \to \text{Moves}^* \\
&\text{anticlock} : \text{number} \times \text{peg} \times \text{peg} \times \text{peg} \to \text{Moves}^* \\
&\text{clock}(0, A, B, C) = \text{skip} \\
&\text{clock}(n+1, A, B, C) = \text{anticlock}(n, A, C, B) :: AB :: \text{anticlock}(n, C, B, A) \\
&\text{anticlock}(0, A, B, C) = \text{skip} \\
&\text{anticlock}(n+1, A, B, C) = \text{anticlock}(n, A, B, C) :: AC :: \text{clock}(n, B, A, C) \\
&\qquad\qquad :: CB :: \text{anticlock}(n, A, B, C)
\end{aligned}$$

where $\text{clock}(n, A, B, C)$ is a sequence of *clockwise* moves for moving n disks from peg A to peg B, using C as a spare peg, i.e. $\text{clock}(n, A, B, C) \in \{AB, BC, CA, \text{skip}\}^*$. Analogously $\text{anticlock}(n, A, B, C)$ is a sequence of *anticlockwise* moves for moving n disks from peg A to peg B, using C as a spare peg, i.e. $\text{anticlock}(n, A, B, C) \in \{AC, CB, BA, \text{skip}\}^*$.

In order to derive an iterative algorithm for the function $\text{clock}(n, A, B, C)$ we will proceed exactly as for $f(n, A, B, C)$, and we will consider the function:

$$z(n) = \langle c(n, CAB), a(n, ACB), a(n, BAC), a(n, CBA), c(n, ABC), c(n, BCA) \rangle,$$

where $c(n, xyz)$ stands for $\text{clock}(n, x, y, z)$ and $a(n, xyz)$ stands for $\text{anticlock}(n, x, y, z)$.

Here $z(n)$ is derived from the symbolic evaluation graph of $\text{clock}(n, A, B, C)$ in the same way in which $t(n)$ was derived from the graph of $f(n, A, B, C)$. We obtain:

$$\begin{aligned}
z(0) &= \langle \text{skip}, \text{skip}, \text{skip}, \text{skip}, \text{skip}, \text{skip} \rangle \\
z(n+1) &= \langle z4 :: CA :: z3, z2 :: AB :: z1 :: BC :: z2, z3 :: BC :: z5 :: CA :: z3,
\end{aligned}$$

$$z4 :: CA :: z6 :: AB :: z4, z2 :: AB :: z4, z3 :: BC :: z2\rangle$$
$$\textbf{where } \langle z1, z2, z3, z4, z5, z6\rangle = z(n)$$

clock$(0, A, B, C)$ = skip

clock$(n+1, A, B, C)$ = $z2 :: AB :: z4$ **where** $\langle z1, z2, z3, z4, z5, z6\rangle = z(n)$.

Using the equivalence of the two schemata:

$z(0) = a$; $z(n+1) = c(z(n))$ and

$Z := a$; **while** $n > 0$ **do** $Z := c(Z)$; $n := n-1$ **od**, we get:

Clockwise Hanoi Program $\{n \geqq 0\}$

if $n = 0$ **then** itclock := skip **else**
begin $n := n-1$; $Z := \langle$skip, skip, skip, skip, skip, skip$\rangle$;
while $n > 0$ **do** $Z := \langle Z4 :: CA :: Z3,$
$Z2 :: AB :: Z1 :: BC :: Z2,$
$Z3 :: BC :: Z5 :: CA :: Z3,$
$Z4 :: CA :: Z6 :: AB :: Z4,$
$Z2 :: AB :: Z4, \quad Z3 :: BC :: Z2\rangle$;
$n := n-1$ **od**
itclock := $Z2 :: AB :: Z4$
end $\{$itclock = clock$(n, A, B, C)\}$

where Zi denotes the ith projection of Z for $i = 1, \ldots, 6$. The iterative algorithm for the cyclic Towers of Hanoi problem has linear running time and exponential space requirements. It has the same time × space performances as the one in [11].

The *anticlockwise* problem can be solved as the clockwise one by considering the symbolic evaluation graph of anticlock(n, A, B, C). In that case the function which transforms the general recursion into a linear one is the 6-tuple obtained from $z(n)$ by interchanging B and C. Therefore the iterative anticlockwise Towers of Hanoi program can be obtained from the clockwise one by performing that interchange and replacing in the last assignment the sequence

$$Z2 :: AB :: Z4 \quad \text{by} \quad Z2 :: AC :: Z1 :: CB :: Z2.$$

Using suitable conditional expressions one may make some minor improvements on the iterative programs we have presented. For instance, one may take advantage of the fact that for $n \leqq 2$ only $Z2$ and $Z4$ are needed for computing itclock. We leave these improvements to the reader.

4. Deriving an iterative solution for the generalized Towers of Hanoi problem.

The generalized Towers of Hanoi problem [5] differs from the standard one in that the disks are initially placed on p pegs with $1 \leqq p \leqq 3$. We always enforce

the constraint that larger disks cannot be on top of smaller disks. Disks are denoted by the numbers $n, n-1, \ldots, 1$ with the usual $>$ ordering.

Let us denote a "configuration" of n disks on the 3 pegs A, B, C by a list l of A's, B's or C's whose length is n, such that $hd(l)$ is the peg where the biggest disk is placed and $tl(l)$ satisfies the same condition with respect to all other disks. (hd and tl are the usual *car* and *cdr* functions on lists). For instance: $[CACBA]$ denotes the configuration "disks 4 and 1 on peg A, disk 2 on peg B and disks 5 and 3 on peg C", $[BBBB]$ denotes the configuration "disks 4, 3, 2 and 1 on peg B, no disks on pegs A and C".

Let us consider the following auxiliary functions:

other : peg × peg → peg

$\text{other}(A, B) = C$; $\text{other}(A, C) = B$; $\text{other}(B, C) = A$; $\text{other}(x, y) = \text{other}(y, x)$

length: list of pegs → num

$\text{length}(NIL) = 0 \qquad \text{length}(x : l) = 1 + \text{length}(l)$

where : is the infix notation of cons for lists.

The following function G solves a generalized Tower of Hanoi problem.

G: list of peg × peg → Moves*.

$G(l, y)$ gives the sequence of moves which transforms the configuration l into the configuration $(yy \ldots y]$ (where the number of y's is equal to length(l)), i.e. $G(l, y)$ moves all disks from the "initial configuration" l to the "target peg" y.

$$G(NIL, y) = \text{skip}$$
$$G(x : l, y) = \textbf{if } x = y \textbf{ then } G(l, y) \textbf{ else } G(l, \text{other}(x, y)) :: xy \\ :: f(\text{length}(l), \text{other}(x, y), y, x)$$

where NIL is the empty list, and $f(n, x, y, z)$ solves the standard Towers of Hanoi problem for n disks.

The correctness proof of the given recursive program is as follows. If the largest disk is already on the target peg y, the solution does not depend on the presence of that disk (see: "**if** $x = y$ **then** $G(l, y)$").

If the largest disk is *not* on the target peg y, we take all other disks away from the target peg and we leave the largest disk alone on the peg where it is at the beginning, say peg x, by performing the sequence of moves $G(l, \text{other}(x, y))$. After these moves we will have all other disks on the peg $\text{other}(x, y)$. Then we move the largest disks to the target peg by the move "xy". Finally we move all other disks from the peg "$\text{other}(x, y)$" to the target peg y by the sequence of moves obtained by solving the standard Towers of Hanoi problem for those disks which are as many as the length of the list l.

One can easily prove by induction on the number of sisks that the recursive solution above computes the shortest possible seuqnce of moves.

In order to obtain the iterative solution for the generalized Towers of Hanoi problem we may use the following schema – flowchart equivalence.

Given a monoid M, with the (associative) operation $\cdot$ and the identity ε, the following function $G: X \to M$

$$G(x) = \begin{cases} \varepsilon & \textbf{if } p(x) \\ G(b(x)) & \textbf{if } \neg p(x) \wedge q(x) \\ G(c(x)) \cdot d(x) & \textbf{if } \neg p(x) \wedge \neg q(x) \end{cases}$$

can be computed by the following flowchart:

$$\{x = N\}$$
$$\text{res} := \varepsilon;\ \{G(N) = G(x) \cdot \text{res}\}$$
$$\textbf{while } \neg p(x) \textbf{ do if } q(x) \textbf{ then } x := b(x)$$
$$\textbf{else begin } \text{res} := d(x) \cdot \text{res};\ x := c(x) \textbf{ end } \textbf{ od}$$
$$\{\text{res} = G(N)\}$$

as it can easily be proved by induction of the number of executions of the **while-do** loop.

Therefore the recursive program for the function G can be transformed into the following program:

$$\{\langle l, y\rangle = \langle L, Y\rangle\}$$
$$\text{res} :- \text{skip};\ \{G(L, Y) = G(l, y) :: \text{res}\}$$

$$\textbf{while } l \neq NIL \textbf{ do if } hd(l) = y \textbf{ then } l := tl(l)$$
$$\textbf{else begin } \text{res} := hd(l) :: f(\text{length}(tl(l)), \text{other}(hd(l), y), y, hd(l)) :: \text{res};$$
$$l, y := tl(l), \text{other}(hd(l), y) \textbf{ end } \textbf{ od}$$
$$\{\text{res} = G(L, Y)\}$$

(where the last assignment is a parallel one) using the above schema-flowchart equivalence where:

ε is skip, $\cdot$ is ::, $x = \langle l, y\rangle \in \text{list} \times \text{peg}$, $p(x)$ is $l = NIL$,

$q(x)$ is $hd(l) = y$, $b(x)$ is $\langle tl(l), y\rangle$, $c(x)$ is $\langle tl(l), \text{other}(hd(l), y)\rangle$,

N is $\langle L, Y\rangle$ and $d(x)$ is $hd(l)y :: f(\text{length}(tl(l)), \text{other}(hd(l), y), y, hd(l))$.

The iterative solution of the generalized Towers of Hanoi problem can be derived from the program above by substituting for $f(n, A, B, C)$ the iterative program we have already derived for the standard Towers of Hanoi problem.

Some minor improvements can be performed on the derived program (as, for instance, the elimination of the variable b). We leave them to the reader.

Both the recursive and iterative solutions of the generalized Towers of Hanoi problem have linear time and exponential space requirements as the ones in [5, 6]. They compute exactly the same sequences of moves.

Generalized Hanoi Program

$\{\langle l, y\rangle = \langle L, Y\rangle$, L is the initial configuration of the disks and Y is the target peg$\}$

res := skip;
while $l \neq NIL$
do if $hd(l) = y$ **then** $l := tl(l)$ **else**
begin $n :=$ length$(tl(l))$;
$a :=$ other$(hd(l), y)$; $b := y$; $c := hd(l)$;
if $n = 0$ **then** $F :=$ skip **elseif** $n = 1$ **then** $F := ab$ **else**
begin $n := n-2$;
if even(n) **then** $T := \langle$skip, skip, skip$\rangle$ **else** $T := \langle ab, bc, ca\rangle$;
while $n > 1$ **do** $T := \langle T1 :: ac :: T2 :: ab :: T3 :: cb :: T1,$
$T2 :: ba :: T3 :: bc :: T1 :: ac :: T2,$
$T3 :: cb :: T1 :: ca :: T2 :: ba :: T3\rangle$;
$n := n-2$ **od**
$F := T1 :: ac :: T2 :: ab :: T3 :: cb :: T1$ **end**;
res := $cb :: F ::$ res;
$l, y := tl(l), a$; **end** **od**
{res = $G(L, Y)$}

REFERENCES

1. M. D. Atkinson, *The cyclic Towers of Hanoi*, Information Processing Letters 13 (1981), 118–119.
2. P. Buneman and L. Levy, *The Towers of Hanoi problem*, Information Processing Letters 10 (1980), 243–244.
3. R. M. Burstall and J. Darlington, *A transformation system for developing recursive programs*, J.A.C.M., Vol. 24, No. 1 (1977) 44–67.
4. E. W. Dijkstra, *A short introduction to the art of programming*, EWD316 (1971).
5. M. C. Er, *The generalized Towers of Hanoi problem*, Private Communication.
6. M. C. Er, *An iterative solution to the generalized Towers of Hanoi problem*, BIT 23 (1983), 295–302.
7. P. J. Hayes, *A note on the Towers of Hanoi problem*, Computer Journal 20 (1977), 282–285.
8. M. S. Paterson and C. E. Hewitt, *Comparative schematology*, Conference on Concurrent Systems and Parallel Computation Project MAC, Woods Hole, Mass. June 2–5 (1970), 119–127.
9. A. Pettorossi, *Methodologies for program transformation and memoing*. Ph.D. Thesis, Computer Science Department, Edinburgh University (1984).
10. S. A. Walker and H. R. Strong, *Characterization of flowchartable recursions*, Journal of Computer and System Sciences 7(4) (1973), 404–447.
11. T. R. Walsh, *Iteration strikes back at the Cyclic Towers of Hanoi*. Information Processing Letters, 16 (1983), 91–93.