Appendix D

The Maple Program

This is the contents of the file procfile.maple, which contains the Maple procedure definitions and the initialization of two variables *nerrors* and *nchecks*.

```
check := proc(state)
local setcontaininglast, setcontainingfirst, rest,
        combinefirstandlast, succ0, succ1, ratio;
global n, x, maxratio, minratio, nerrors, nchecks;
   nchecks := nchecks + 1 ;
   setcontaininglast, rest := selectremove(has, state, n) ;
      # split state into the part containing n (if it exists) and the rest
   if setcontaininglast = {{n}} then succ0 := 0
   elif setcontaininglast = {} then succ0 := map(shift1, state)
   else succ0 := map(shift1, rest union {setcontaininglast[1] minus {n}})
   end if;
   setcontainingfirst, rest := selectremove(has, rest, 1) ;
   combinefirstandlast :=
      map(op, setcontaininglast) union map(op, setcontainingfirst);
   succ1 := map(shift1, rest union {(combinefirstandlast union {0}) minus {n}}) ;
   if not [state] in [indices(x)] then
      nerrors := nerrors + 1 ; error "Value %1 not initialized.", state
   end if;
   if not [succ1] in [indices(x)] then
      nerrors := nerrors + 1 ;
      error "Value %1 (succ1) not  initialized for %2.", succ1, state
   end if;
   if succ0 ≠ 0 and not [succ0] in [indices(x)] then
      nerrors := nerrors + 1 ;
      error "Value %1 (succ0) not  initialized for %2.", succ0, state
   end if;
   if succ0 = 0 then    # xnew[state] := xold[succ1]
      ratio := x[succ1]/x[state]
   else       # xnew[state] := xold[succ1] + xnew[succ0]
```

```
        ratio := x[succ1]/(x[state] − x[succ0])
    end if;
    minratio := min(minratio, ratio) ;
    maxratio := max(maxratio, ratio)
end proc ;

shift1 := proc(part) map(x → x + 1, part) end proc ;

setx := proc(state, value, flag)
global x;
    x[state] := value ; if flag = 0 then check(state) ; x := table() end if
end proc ;

checkx := proc(state, value, flag)
global rememberstate, remembervalue, nerrors, nchecks;
    if flag = 0 then
        nchecks := nchecks + 1 ; rememberstate := state ; remembervalue := value
    else
        if rememberstate ≠ state then
            nerrors := nerrors + 1 ;
            error “incorrect state %1,  should be %2”, state, rememberstate
        end if;
        if remembervalue ≠ value then
            nerrors := nerrors + 1 ;
            error “incorrect value %2 for state %1, should  be %3”,
                    state, value, remembervalue
        end if
    end if
end proc ;

init := proc(w)
global n, x, maxratio, minratio, nerrors, nchecks;
    n := w ; x := table() ; minratio := ∞ ; maxratio := 0 ; nerrors := −1 ; nchecks := 0
end proc ;

finish := proc()
local scale;
global minratio, maxratio, min1, max1, nchecks;
    scale := 10^9 ;
    min1 := floor(minratio ∗ scale) ;
    max1 := ceil(maxratio ∗ scale) ;
    print(minratio, maxratio, evalf([minratio, maxratio]), cat(min1, “*10ˆ-9”),
        cat(max1, “*10ˆ-9”))
end proc ;
```

```
terminate := proc()
global nerrors, nchecks;
    printf("  %d configurations were  checked.\n", nchecks);
    if nerrors = 0 then printf("  OK.\n")
    elif 0 < nerrors  then printf("There were %d errors.\n", nerrors)
    end if
end proc;

nerrors := 0;    # initialisation
nchecks := 0;
```