Fibonacci Reveals the Secrets of the Towers of Hanoi

Two Exercises on Automatic Program Derivation

Alberto Pettorossi
alberto.pettorossi@uniroma2.it
DISP - Universita' di Roma Tor Vergata
Via del Politecnico,1 I-00133 Rome, Italy

Roma, March 19th, 2004

Program Transformation (Burstall-Darlington)

```
fib(0) = 0
fib(1) = 1
fib(n+2) = fib(n+1)+ fib(n)
```

exponential time

transformation rules

```
fib(0) = 0
fib(1) = 1
fib(2) = 1
fib(n+3) = 2u+v     where <u,v> = g(n)
g(0) = <1,0>
g(n+1) = <u+v, u> where <u,v> = g(n)
```

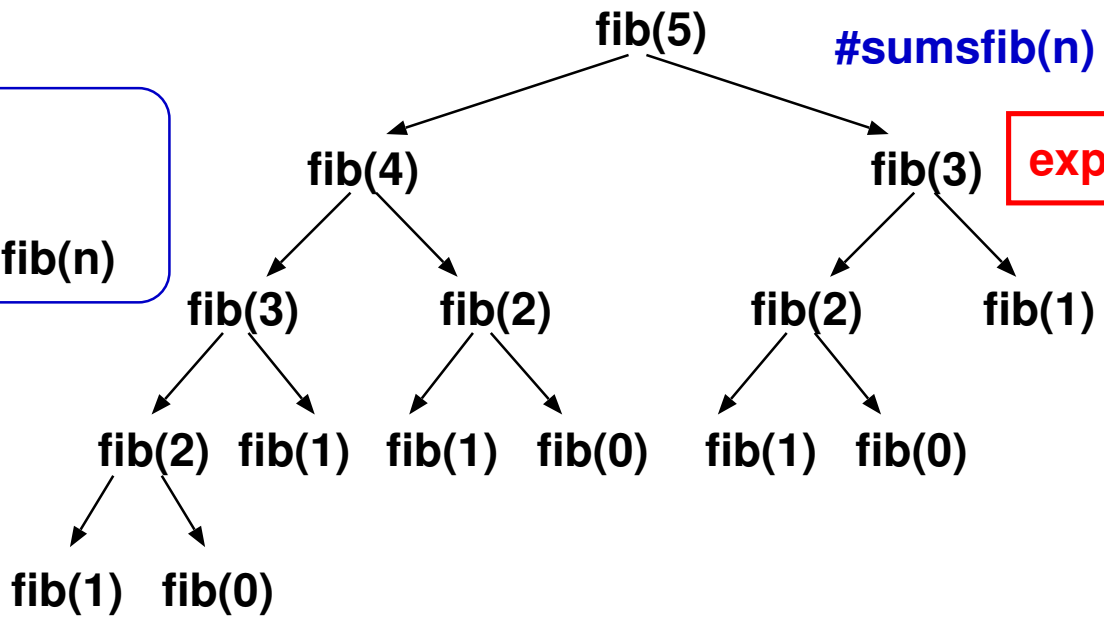
linear time

R.M. Burstall, J. Darlington: "A Transformation System for Developing Recursive Programs" Journ. ACM. Jan. 1977

Fibonacci (1)

using sums:

fib(0) = 0
fib(1) = 1
fib(n+2) = fib(n+1) + fib(n)

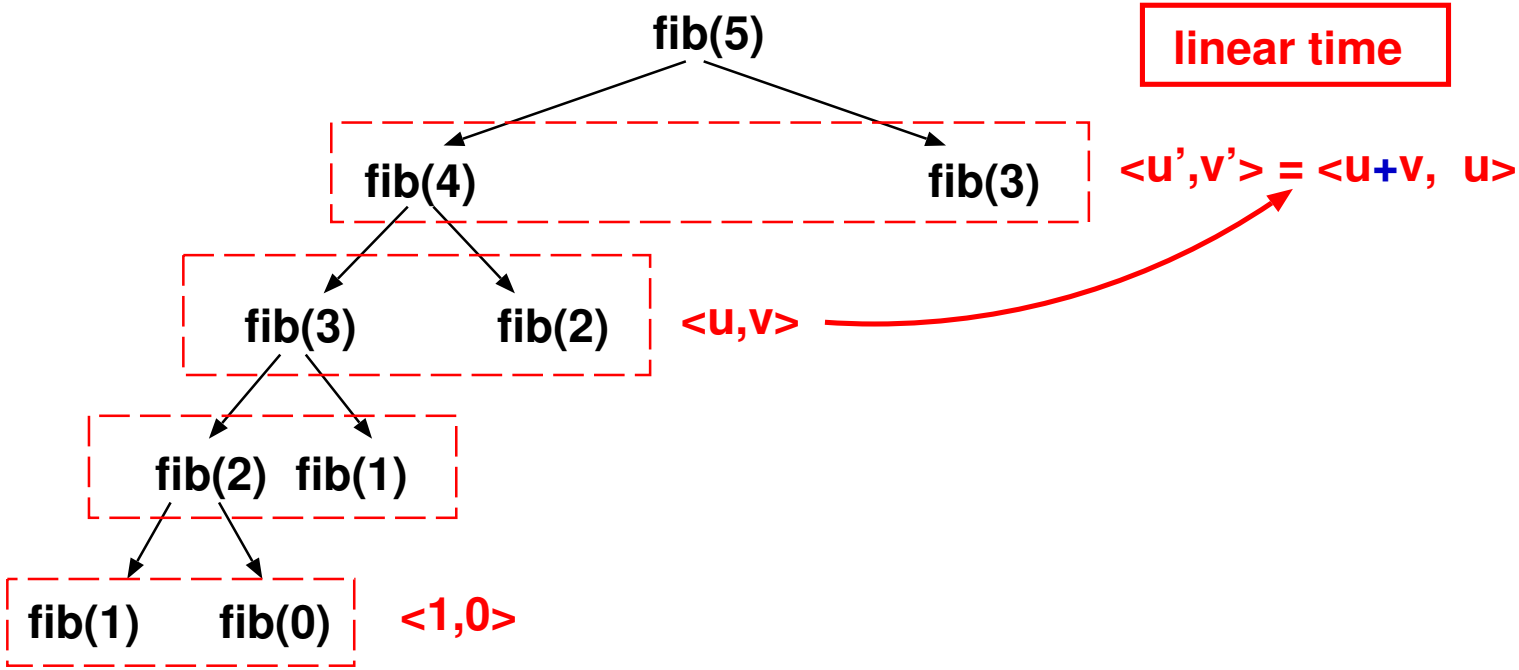


#sumsfib(n) = fib(n+1) -1

exponential time

---

using sums:



linear time

Fibonacci (2)

1. fib(0) = 0
2. fib(1) = 1
3. fib(n+2) = fib(n+1)+ fib(n)

n stands for $s^n(0)$

↓ unfolding 3 using 2 and 3

4.1 fib(2) = 1 + fib(0)

4.2 fib(n+3) = fib(n+1) + fib(n) + fib(n+1)

↓ unfolding 4.1 using 1

5. fib(2) = 1

↓ where-abstraction of 4.2

6. fib(n+3) = 2 u + v where <u,v> = <fib(n+1), fib(n)>

↓ definition

7. g(n) = <fib(n+1), fib(n)>

↓ folding 6 using 7

8. fib(n+3) = 2 u + v where <u,v> = g(n)

↓

...

Fibonacci (3)

...

↓ unfolding of 7 using 2 and 3

9.1 g(0) = <1, fib(0)>
9.2 g(n+1) = <fib(n+1)+fib(n), fib(n+1)>

↓ unfolding of 9.1 using 1

10. g(0) = <1,0>

↓ where-abstraction of 9.2

11. g(n+1) = <u+v, u> where <u,v>=<fib(n+1), fib(n)>

↓ folding of 11 using 7

12. g(n+1) = <u+v, u> where <u,v>=g(n)

Fibonacci (4)

1. fib(0) = 0
2. fib(1) = 1
5. fib(2) = 1
8. fib(n+3) = 2u+v where <u,v> = g(n)
10. g(0) = <1,0>
12. g(n+1) = <u+v, u> where <u,v> = g(n)

linear time

Fibonacci (5)

using:

$g(0) = a$
$g(n+1) = b(g(n))$

≡

```
G:=a;
while n>0 do n:=n-1;
                G:=b(G)
             od
```

we get:

```
if n=0 then F:=0 else
if n=1 then F:=1 else
if n=2 then F:=1 else
begin n := n-3;
        G:=<1,0>;
        while n>0 do n:=n-1;  G:=<G1 + G2, G1 > od
        F := 2G1+ G2
end
```

Computing Fibonacci (1)

using multiplications and matrices (Dijkstra et al.):

$$\begin{bmatrix} fib(n+1) \\ fib(n) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

logarithmic time. 4 cells

Example : n=8

$$\begin{bmatrix} fib(9) \\ fib(8) \end{bmatrix} = \begin{bmatrix} 34 & 21 \\ 21 & 13 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

---

using multiplications (Burstall-Darlington):

logarithmic time. 2 cells

$fib(n) = \pi 1 \; r(n)$

$r(0) = \langle 0,1 \rangle$

$r(2k) = \langle u^2 + 2uv, u^2 + v^2 \rangle$ where $\langle u,v \rangle = r(k)$

$r(2k+1) = \langle (u+v)^2 + u^2, u^2 + 2uv \rangle$ where $\langle u,v \rangle = r(k)$

$\pi 1 \langle a,b \rangle = a$

Computing Fibonacci (2)

using exponentiations:

$$fib(n+2) = fib(n+1) + fib(n)$$

$$x^2 = x+1 \qquad x^2-x-1 = 0 \qquad x_{1,2} = \frac{1 \pm \sqrt{5}}{2}$$

$$fib(n) = c_1 x_1^n + c_2 x_2^n$$

$$fib(n) = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

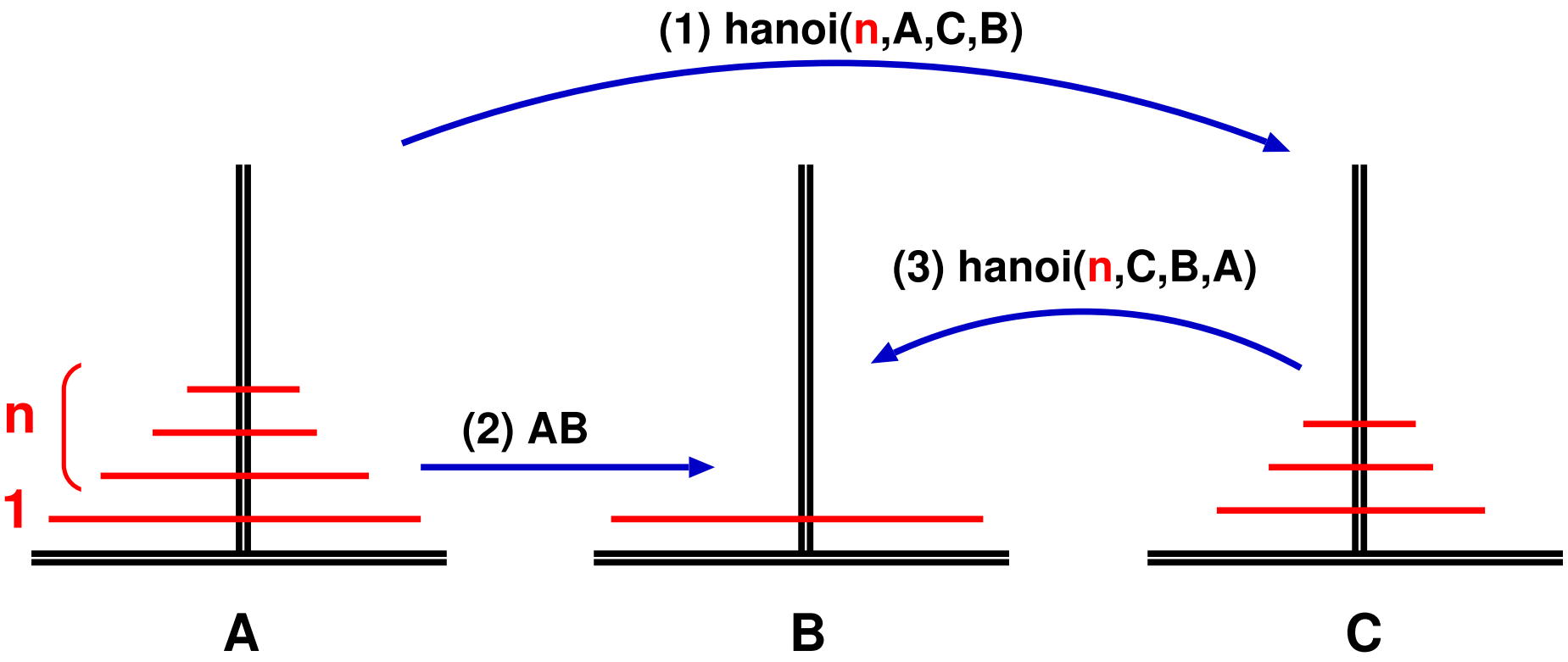
dominant term

modulo < 1

constant time

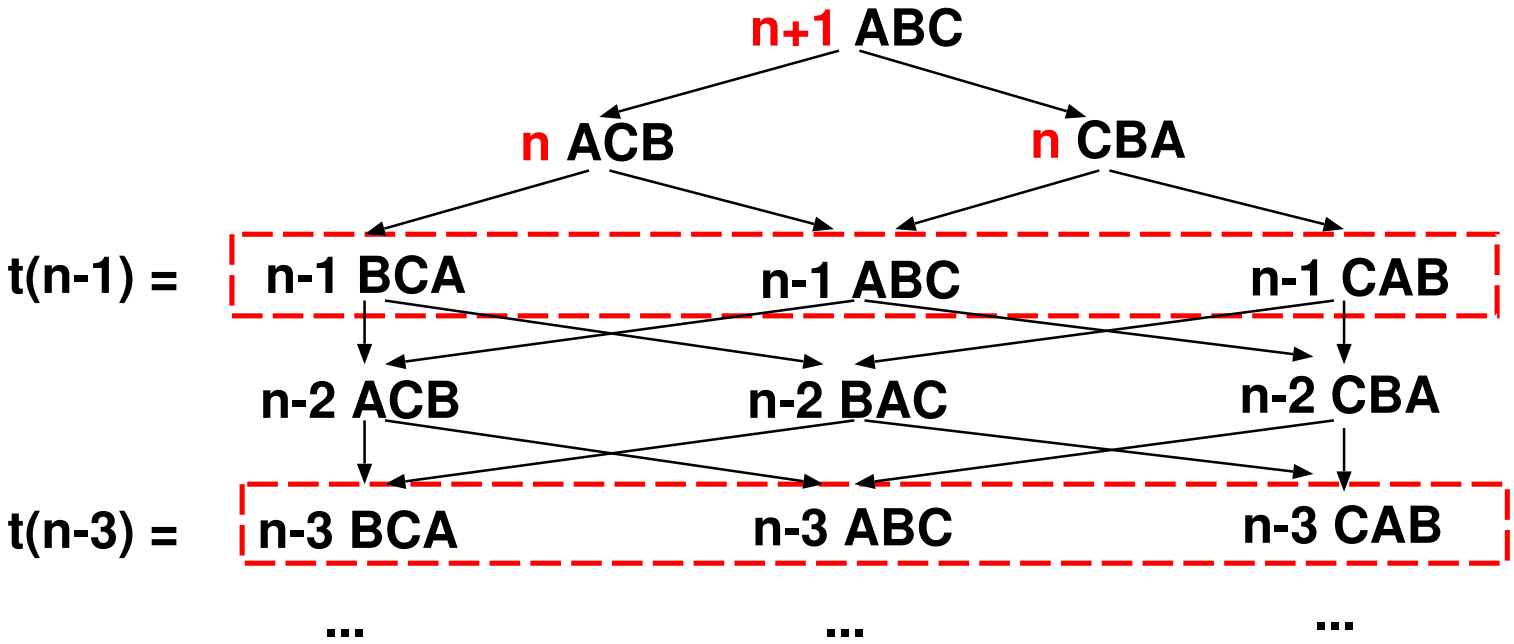
Towers of Hanoi (1)

hanoi(0,A,B,C) = skip
hanoi(n+1,A,B,C) = hanoi(n,A,C,B) : AB : hanoi(n,C,B,A)



Towers of Hanoi (2)

hanoi(0,A,B,C) = skip
hanoi(n+1,A,B,C) = hanoi(n,A,C,B) : AB : hanoi(n,C,B,A)



t(n) = < hanoi(n,A,B,C), hanoi(n,B,C,A), hanoi(n,C,A,B) >

---

t(0) = < skip, skip, skip >
t(1) = < AB, BC, CA >
t(n+2) = < u : AC : v : AB : w : CB : u,
v : BA : w : BC : u : AC : v,
w : CB : u : CA : v : BA : w > where <u,v,w> = t(n)

Towers of Hanoi (3)

Using

```
h(0) = a
h(1) = b
h(n+2) = c(h(n))
```

≡

```
                { N ≥ 0 }
n := N;
if even(n) then res := a else res := b;
while n>1 do res := c(res);
                    n:= n-2
                od
          { res = h(N) }
```

we get:

```
                                        { N ≥ 0 }
var T, TT : array [1..3] of string;
var i,k : integer;
begin i:= N;
     if  even(i)  then  for k := 1 to 3  do T[k] := "   "
                  else  begin T[1]:= " AB"; T[2]:= " BC"; T[3]:= " CA" end;
     while i>1 do
          begin for k := 1 to 3  do TT[k] := T[k];
              T[1] := TT[1] + " AC" + TT[2] + " AB" + TT[3] + " CB" + TT[1];
              T[2] := TT[2] + " BA" + TT[3] + " BC" + TT[1] + " AC" + TT[2];
              T[3] := TT[3] + " CB" + TT[1] + " CA" + TT[2] + " BA" + TT[3];
               i := i-2
          end
     res:=T[1];
end                          { res = hanoi(N,A,B,C) }
```