Summing Even Fibbonacci Numbers

May 17, 2015

1 Specification

$$t, a, b \in \mathbb{N} \rightsquigarrow s_{t,a,b} = \sum_{k \in \mathbb{N}} f_k(a,b) \cdot \boldsymbol{\tau}(f_k(a,b); t) \tag{1}$$

Where $\boldsymbol{\tau}(\alpha)$ is defined as:

$$\boldsymbol{\tau}(\alpha; t) = \begin{cases} 1 & even(\alpha) \wedge (\alpha < t) \\ 0 & otherwise \end{cases}$$

and, $\mathbf{even}(\alpha)$ is defined as:

$$\mathbf{even}(\alpha) = \begin{cases} true & \alpha \textbf{ mod } 2 = 0 \\ false & otherwise \end{cases}$$

The specification (1) dictates a program where given preconditions $t, a, b \in \mathbb{N}$ terminate in a state where the variable s is set to $\sum_{k \in \mathbb{N}} f_k(a,b) \cdot \boldsymbol{\tau}(f_k(a,b); t)$. This expression is given in a similar form within the problem statement and provides an unambiguous representation of the sum of even terms in a generalised fibonacci sequence, since the indicator function on odd terms and terms $\geq t$ maps the value for that particular $f_k(a,b)$ to 0.

2 Refinement

For the purposes of brevity [1] [2];
$Q = s_{t,a,b} = \sum_{k \in \mathbb{N}} f_k(a,b) \cdot \boldsymbol{\tau}(f_k(a,b); t)$

$\beta = t, a, b \in \mathbb{N}$

[1] Concrete code will not appear in sequential refinement steps. So for example (2) $\sqsubseteq$ any program that refines $\beta \wedge g_1 \rightsquigarrow Q$.

[2] In this particular derivation variables and expressions not appearing in sequential steps are assumed to retain their value.

$g_1 = even(a) \wedge even(b)$
Using the if refinement rule:

$$\begin{aligned} (1) \sqsubseteq \textbf{ if } g_1 \textbf{ then } \beta \wedge g_1 \rightsquigarrow Q & \quad (2) \\ \textbf{else } \beta \wedge \neg g_1 \rightsquigarrow Q \textbf{ fi} & \quad (3) \end{aligned}$$

From here we define the predicate:

$$P(k) = \forall i \in \mathbb{N}(f_i(a,b) < t \iff f_i(a,b) \leq f_k(a,b)) \wedge (f_k(a,b) < t)$$

Applying the sequential composition rule to (2), we obtain:

$$\begin{aligned} (2) \sqsubseteq \beta \wedge g_1 \rightsquigarrow P(N); & \quad (4) \\ P(N) \rightsquigarrow Q & \quad (5) \end{aligned}$$

To obtain N, an index for the largest fibonacci number in a sequence of strictly even fibonacci numbers, we look to the closed form for a generalised fibonacci term:

$$f(n) = t_1\phi^n + t_2\psi^n$$

Where $\phi = \frac{1+\sqrt{5}}{2}$ and $\psi = \frac{1-\sqrt{5}}{2}$

General expressions for t_1 and t_2 can also be found by setting boundary conditions $f(0) = a$ and $f(1) = b$ and performing elimination, namely:

$$t_1 = \frac{b - a\psi}{\phi - \psi} \text{ , } t_2 = \frac{b - a\phi}{\psi - \phi}$$

From here we set:

$$t = t_1\phi^n + t_2\psi^n$$

Reduction of this form leads to a system of 2 quadratic equations. Since we wish n $\in \mathbb{R}$ to form a maximal coordinate vector satisfying the equation we select the largest root between:

$$\phi^n = \frac{t + \sqrt{t^2 \pm 4t_1t_2}}{2t_1}$$

Namely:

$$\phi^n = \frac{t + \sqrt{t^2 + |4t_1t_2|}}{2t_1}$$

Solving for n:

$$n = \frac{ln\left(\frac{t+\sqrt{t^2+|4t_1t_2|}}{2t_1}\right)}{ln(\phi)}$$

A successive application of the sequential composition rule is necessary on (4):

$$\begin{aligned} (4) \sqsubseteq \beta \wedge g_1 \rightsquigarrow n = G_f; & \quad (6) \\ n = G_f \rightsquigarrow P(N) & \quad (7) \end{aligned}$$

Where $G_f = \frac{ln\left(\frac{t+\sqrt{t^2+|4t_1t_2|}}{2t_1}\right)}{ln(\phi)}$

Rearranging (6) into the form: $(\beta \wedge g_1)[^{G_f}/_n] \rightsquigarrow (\beta \wedge g_1)$ we can apply the assignment rule:

$$(6) \sqsubseteq n := G_f \tag{8}$$

Unfortunately (at least for the sake of elegance) we wish to refrain from including the case where $f_k(a,b) = t$ in our s, as per the overall specification. This means that the program $N = \lfloor n \rfloor$ is insufficient to capture the specification of (7), to remedy this an alternation is used. Using the if rule:

$$(7) \sqsubseteq \textbf{if } g_2 \textbf{ then } n = G_f \wedge g_2 \rightsquigarrow P(N) \tag{9}$$

$$\textbf{else } n = G_f \wedge \neg g_2 \rightsquigarrow P(N) \textbf{ fi} \tag{10}$$

Where $g_2 = n \in \mathbb{N}$

If $n \in \mathbb{N}$, we have the case that $f_n(a,b) = t$, since clearly $t \not< t$, we wish to obtain i, $\forall i < n(\exists j < n \cdot i \leq j)$ since $i, n \in \mathbb{N}$ by well ordering, $i = n - 1$. Thus, we can rearrange (9) into the form $(n = G_f \wedge g_2)[^{n-1}/_N] \rightsquigarrow (n = G_f \wedge g_2 \wedge N = n - 1)$, since $(n = G_f \wedge g_2) \implies P(N)$. Using the assignment rule:

$$(9) \sqsubseteq N := n - 1 \tag{11}$$

Alternatively, if we have the situation where the coordinate vector lies between two natural numbers i.e. $\neg g_2$, trivially the nearest whole integer $< n$ must be the index of the largest fibonacci number in the sequence $< t$. Rearranging (10) into the correct form to apply assignment follows an structure almost identical to (9): $(n = G_f \wedge g_2)[^{\lfloor n \rfloor}/_N] \rightsquigarrow (n = G_f \wedge g_2)$, since $(n = G_f \wedge g_2 \wedge N = \lfloor n \rfloor) \implies P(N)$.

$$(10) \sqsubseteq N := \lfloor n \rfloor \tag{12}$$

Given objects of this recurrence are given by $f(n) = t_1\phi^n + t_2\psi^n$, the closed form sum of $f(0) + f(1) + ... + f(k)$ can be easily obtained by the taking the geometric sum of elements:

$$F(k) = t_1\Big(\frac{1-\phi^{k+1}}{1-\phi}\Big) + t_2\Big(\frac{1-\psi^{k+1}}{1-\psi}\Big)$$

Since N is the coordinate vector of the largest element $< t$, we obtain an expression for s, namely $s = F(N)$. (5) becomes $P(N)[^{F(N)}/_s] \rightsquigarrow s = F(N)$, since $F(N) = \sum_{k \in \mathbb{N}} f_k(a,b) \cdot \boldsymbol{\tau}(f_k(a,b);t)$ and $s = F(N)$, it follows Q is true. Thus by the assignment rule:

$$(5) \sqsubseteq s := F(N) \tag{13}$$

Much of the refinement of (3) follows a methodology very similar to that of (2) and is only wholly included for the purposes of austerity, as such any specific proof or reasoning obligations to re-arise are omitted. In order to proceed with the solution, the following lemma is necessary:

Lemma: For the generalised fibonacci sequence given by the boundary conditions (a, b), if $\exists k \in \mathbb{N} \cdot (a = 2k+1) \vee (b = 2k+1)$, then a recurrence relation exists between the even terms given by $f_{n+2}(a, b) = 4f_{n+1}(a, b) + f_n(a, b)$.

Proof. First, we must prove that there exists a pattern common to all generalised Fibonacci sequences that do not have a and b both even. We must note that (for $k, j, q, p \in \mathbb{Z}$):

$$\begin{aligned}
(2k+1) + (2j+1) &= 2k + 2j + 2 = 2(k+j+1) = 2q \\
(2k+1) + 2j &= 2k + 2j + 1 = 2(k+j) + 1 = 2p + 1 \\
2k + (2j+1) &= 2k + 2j + 1 = 2(k+j) + 1 = 2p + 1
\end{aligned}$$

Hence, we can consider the form of our generalised fibonacci sequence as being:

$$..., O, O, E, O, O, E, O, O, E, ...$$

Translating our recurrence relation $f_{n+2}(a, b) = 4f_{n+1}(a, b) + f_n(a, b)$ back into the form of the fibonacci sequence:

$$f_{n+6}(a, b) = 4f_{n+3} + f_n(a, b)$$

From here, using our fibonacci recurrence $f_{n+2} = f_{n+1} + f_n$:

$$\begin{aligned}
LHS &= f_{n+6}(a, b) \\
&= f_{n+5}(a, b) + f_{n+4}(a, b) \\
&= 2f_{n+4}(a, b) + f_{n+3}(a, b) \\
&= 2(f_{n+3}(a, b) + f_{n+2}(a, b)) + f_{n+3}(a, b) \\
&= 3f_{n+3}(a, b) + f_{n+2}(a, b) + f_{n+1}(a, b) + f_n(a, b) \\
&= 4f_{n+3}(a, b) + f_n(a, b) = RHS
\end{aligned}$$

$\square$

From here we define the predicate:

$$\begin{aligned}
L(\gamma, \delta) = \forall l \in V(even(f_\gamma(a, b)) \wedge f_\gamma(a, b) \leq l) \\
\wedge(even(f_\delta(a, b)) \wedge (l \neq f_\gamma(a, b) \iff f_\delta(a, b) \leq l)
\end{aligned}$$

Where $V = \{f_k(a, b), f_{k+3}(a, b), ...\}$ and $even(f_k(a, b))$

Applying the sequential composition rule:

$$\begin{aligned}
(3) \sqsubseteq \beta \wedge \neg g_1 \rightsquigarrow L(A, B); \quad (14) \\
L(A, B) \rightsquigarrow Q \quad (15)
\end{aligned}$$

Using the if rule:

$$(14) \sqsubseteq \textbf{if } g_3 \textbf{ then } \beta \wedge \neg g_1 \wedge g_3 \rightsquigarrow L(A,B) \tag{16}$$
$$\textbf{else } \beta \wedge \neg g_1 \neg g_3 \rightsquigarrow L(A,B) \textbf{ fi} \tag{17}$$

Where $g_3 = even(a) \wedge \neg even(b)$

From here we can rearrange (16) into the form $(\beta \wedge \neg g_1 \wedge g_3)[^a/_A][^{a+2b}/_B] \rightsquigarrow (\beta \wedge \neg g_1 \wedge g_3) \wedge (A = a \wedge B = a + 2b)$. Here the post condition $\implies L(A,B)$ since clearly if even(a), a being $f_0(a,b)$ then a is the smallest even number in the sequence, from the proof of the lemma we can also conclude that if $\neg even(b)$ then the second smallest even number belonging to the sequence is $f_3(a,b) = f_2(a,b) + f_1(a,b) = 2f_1(a,b) + f_0(a,b) = a + 2b$. Using the assignment rule:

$$(16) \sqsubseteq A := a; B := a + 2b \tag{18}$$

Since we have no **if else if else** rule. We are required to nest another if statement within the else on (17).
Using the if rule:

$$(17) \sqsubseteq \textbf{if } g_4 \textbf{ then } \beta \wedge \neg g_1 \wedge \neg g_3 \wedge g_4 \rightsquigarrow L(A,B) \tag{19}$$
$$\textbf{else } \beta \wedge \neg g_1 \wedge \neg g_3 \wedge \neg g_4 \rightsquigarrow L(A,B) \textbf{ fi} \tag{20}$$

Where $g_4 = \neg even(a) \wedge even(b)$

By a similar argument to the previous assignment, we have $(\beta \wedge \neg g_1 \wedge \neg g_3 \wedge g_4)[^b/_A][^{2a+3b}/_B] \rightsquigarrow (\beta \wedge \neg g_1 \wedge \neg g_3 \wedge g_4)$ and application of the assignment rule:

$$(19) \sqsubseteq A := b; B := 2a + 3b \tag{21}$$

The final case (20) is given by $h_1 = \neg even(a) \wedge \neg even(b)$, here no guard is necessary since

$$\begin{aligned}
&\neg g_1 \wedge \neg g_3 \wedge \neg g_4 \\
\equiv&(\neg even(a) \vee \neg even(b)) \\
\wedge&(\neg even(a) \vee even(b)) \\
\wedge&(even(a) \vee \neg even(b)) \\
\equiv&\neg even(a) \wedge \neg even(b) \\
\equiv&h_1
\end{aligned}$$

With an argument similar to the other cases we produce $(\beta \wedge h_1)[^{a+b}/_A][^{3a+5b}/_B] \rightsquigarrow (\beta \wedge h_1)$, applying the assignment rule yet again:

$$(20) \sqsubseteq A := a + b; B := 3a + 5b \tag{22}$$

To finish the program we look to the general closed form for the recurrence relation $f_{n+2}(a,b) = 4f_{n+1}(a,b) + f_n(a,b)$, which is given by:

$$r(n) = s_1\mu^n + s_2\nu^n$$

Where $\mu = 2 + \sqrt{5}$ and $\nu = 2 - \sqrt{5}$

From this point our derivation is <u>exactly</u> the same as steps (4)-(13) except the fibonacci sequence is replaced by the sequence consisting solely of the even terms and μ, ν replace ϕ, ψ in our expressions, respectively.
We define the predicate:

$$Q(k) = \forall i \in \mathbb{N}(r_i(A,B) < t \iff r_i(A,B) \leq r_k(A,B)) \land (r_k(A,B) < t)$$

Our expressions for s_1, s_2

$$s_1 = \frac{B - A\nu}{\mu - \nu} \ , \ s_2 = \frac{B - A\mu}{\nu - \mu}$$

Setting:

$$t = s_1\mu^n + s_2\nu^n$$

Our n:

$$n = \frac{ln\left(\frac{t+\sqrt{t^2+|4s_1s_2|}}{2s_1}\right)}{ln(\mu)}$$

Sequential composition:

$$(15) \sqsubseteq L(A,B) \rightsquigarrow Q(N); \tag{23}$$
$$Q(N) \rightsquigarrow Q \tag{24}$$

Successively:

$$(23) \sqsubseteq L(A,B) \rightsquigarrow n = H_f; \tag{25}$$
$$n = H_f \rightsquigarrow Q(N) \tag{26}$$

Where $H_f = \frac{ln\left(\frac{t+\sqrt{t^2+|4s_1s_2|}}{2s_1}\right)}{ln(\mu)}$
Assignment:

$$(25) \sqsubseteq n := H_f \tag{27}$$

Producing N:

$$(26) \sqsubseteq \textbf{if } g_2 \textbf{ then } n = H_f \land g_2 \rightsquigarrow Q(N) \tag{28}$$
$$\textbf{else } n = H_f \land \neg g_2 \rightsquigarrow Q(N) \textbf{ fi} \tag{29}$$

Where $g_2 = n \in \mathbb{N}$
Associated assignments:

$$(28) \sqsubseteq N := n - 1 \tag{30}$$

$$(29) \sqsubseteq N := \lfloor n \rfloor \tag{31}$$

Closed form Sum:

$$R(k) = s_1\Big(\frac{1-\mu^{k+1}}{1-\mu}\Big) + s_2\Big(\frac{1-\nu^{k+1}}{1-\nu}\Big)$$

Final assignment:

$$(24) \sqsubseteq s := R(N) \tag{32}$$

3 Changes during C Implementation

Our C implementation, particularly given that it only involved the standard fixed precision arithmetic involved some minor changes to the code produced during refinement. To begin with, instead of refining and introducing the C function $even(\alpha)$, it was substituted by the function $even(\alpha) = (\alpha mod 2 == 0)$, which is correct by definition of the even function.

While constants ϕ (PSI), ψ (PHIC), μ (MU), ν (MUC) were introduced into the C program as #define constants, for the purposes of code clarity t_1, t_2, s_1, s_2 are also defined within the function using the const keyword to avoid reassignment, const is also used in setting the value of n, which remains unchanged.

The final assignment $s := F(N)/R(N)$ is replaced by the return value of the function sef(t,a,b). So informally Q is satisfied if the function returns a value with respect to the parameters that satisfy the RHS of Q. Since the entire program is essentially an alternation, we can include 2 return statements since g_1 and $\neg g_1$ cannot simultaneously hold.

Although we are guaranteed a s from the refined program is an integer, producing this solution using fixed precision arithmetic proves inherently problematic. Luckily for small numbers, we still have a very close approximation to the correct solution. In order to ensure C correctly evaluate this approximation, a rounding function was implemented:

```
double round(double d)
{
  if (d - floor(d) > 0.5) return ceil(d);
  return (double) (unsigned int) floor(d);
}
```

As a final note, the results of both the poor precision of data types and the c evaluation of arithmetic operations within this implementation make it difficult to guarantee bounds on t, a, b to ensure a correct return solution. However, testing has shown that maximum bounds allowed are significantly lower than the size of a unsigned long, sometimes up to 2 orders of magnitude.