Algorithmic Manipulation of Fibonacci Identities

Stanley Rabinowitz
12 Vine Brook Road
Westford, MA 01886 USA

1. Introduction.

Methods for manipulating trigonometric expressions, such as changing sums to products, changing products to sums, expanding functions of multiple angles, etc., are well-known [1]. In fact, the process of verifying trigonometric identities is algorithmic (see [2] or [5]). Roughly speaking, all trigonometric identities can be derived from the basic identity $\sin^2 x + \cos^2 x = 1$.

Methods for manipulating expressions involving Fibonacci and Lucas numbers are also known; however, they do not seem to be systematically collected in one place. The fact that verifying Fibonacci identities is algorithmic seems to be less well-known, especially considering the strong relationship there is between Fibonacci numbers and trigonometric functions (see equation (7) below).

For example, the problem of establishing the identity

$$\begin{aligned} F_{14r}(F_{n+4r}^7 + F_n^7) - 7F_{10r}(F_{n+4r}^6 F_n + F_{n+4r}F_n^6) + 21F_{6r}(F_{n+4r}^5 F_n^2 + F_{n+4r}^2 F_n^5) \\ - 35F_{2r}(F_{n+4r}^4 F_n^3 + F_{n+4r}^3 F_n^4) = F_{4r}^7 F_{7n+14} \end{aligned} \tag{1}$$

is still considered "hard" as witnessed by the fact that this problem appeared in the advanced problem section of *The Fibonacci Quarterly* [10]. Yet this identity (and all such identities) can be proven using a straighforward algorithm that we shall describe below. Roughly speaking, all Fibonacci identities can be derived from the basic identity $L_n^2 - 5F_n^2 = 4(-1)^n$.

2. The Basic Algorithms.

The key idea to algorithmically proving identities involving polynomials in F_{an+b} and L_{an+b} is to first reduce them to polynomials in F_n and L_n. To do that, we need reduction formulas for F_{m+n} and L_{m+n}. These are well-known. In fact, all the formulas presented in this paper can be found in the literature (see, for example, [9]). Most of these formulas were known to Lucas in the 19th century [6].

ALGORITHM "FibEvaluate" TO NUMERICALLY EVALUATE F_k AND L_k:
Given an integer constant k, to evaluate F_k or L_k numerically, apply the following algorithm:
STEP 1: [Make subscript positive]. If $k < 0$, apply algorithm "FibNegate" (given below).
STEP 2: [Recurse]. If $k > 1$, apply the recursion:

$$\begin{aligned} F_n &= F_{n-1} + F_{n-2} \\ L_n &= L_{n-1} + L_{n-2}. \end{aligned} \tag{2}$$

Reprinted from Applications of Fibonacci Numbers, volume 6 (edited by G. E. Bergum, et al.), Kluwer Academic Publishers, Dordrecht, The Netherlands: 1996, pp. 389–408

Repeat step 2 until $k < 2$. Step 2 reduces the subscript by 1, so the recursion must eventually terminate.
STEP 3: [Initial values]. If k is 0 or 1, then use the following initial values:

$$F_0 = 0, \quad F_1 = 1; \quad L_0 = 2, \quad L_1 = 1. \tag{3}$$

Reprinted from "Applications of Fibonacci Numbers, volume 6, (edited by G. E. Bergum, et al.), pp. 389–408
Corrected 6/98

NOTE: While this may not be the fastest way to evaluate F_k and L_k, it is nevertheless an effective algorithm. It is our purpose here to present effective methods to show that the manipulations described are all algorithmic and it is not our purpose to find the most efficient algorithms possible. Faster algorithms can be obtained by using the double argument formulas given below in display (12). For a description and analysis of fast methods of numerically evaluating F_k and L_k, see chapter 7 of [7].

ALGORITHM "FibNegate" TO REMOVE NEGATIVE SUBSCRIPTS:
Use the identities:

$$\begin{aligned} F_{-n} &= (-1)^{n+1} F_n \\ L_{-n} &= (-1)^n L_n. \end{aligned} \tag{4}$$

ALGORITHM "FibReduce" TO REMOVE SUMS IN SUBSCRIPTS:
Use the identities:

$$\begin{aligned} F_{m+n} &= \frac{F_m L_n + L_m F_n}{2} \\ L_{m+n} &= \frac{5F_m F_n + L_m L_n}{2}. \end{aligned} \tag{5}$$

These are also called the "addition formulas".

3. The Fundamental Identity Connecting F and L.

The Fibonacci and Lucas numbers are connected by the well-known identity:

$$L_n^2 - 5F_n^2 = 4(-1)^n. \tag{6}$$

In the same way that all trigonometric identities are consequences of the fundamental identity $\sin^2 x + \cos^2 x = 1$, all Fibonacci/Lucas identities are consequences of the fundamental identity (6). That is, equation (6) is the unique identity connecting F_n and L_n.

The proof is a consequence of the fact that

$$\begin{aligned} F_n &= \frac{2}{\sqrt{5}} i^{n+1} \sin(-\frac{in}{2} \ln \frac{1+\sqrt{5}}{1-\sqrt{5}}) \\ L_n &= 2i^n \cos(-\frac{in}{2} \ln \frac{1+\sqrt{5}}{1-\sqrt{5}}). \end{aligned} \quad \text{or} \quad \begin{aligned} F_n &= \frac{2}{\sqrt{5}} i^n \sinh\Big(\frac{n}{2} \ln \frac{1+\sqrt{5}}{1-\sqrt{5}}\Big) \\ L_n &= 2i^n \cosh\Big(\frac{n}{2} \ln \frac{1+\sqrt{5}}{1-\sqrt{5}}\Big) \end{aligned} \tag{7}$$

so that the result is equivalent to the result for trigonometric polynomials, whose proof can be found in [2].

The fundamental identity allows us to take any polynomial in F_n and L_n and remove all powers of L_n (of degree 2 or higher) by using the following algorithm.

ALGORITHM "RemovePowersOfL":

Use the identity:

$$L_n^2 = 5F_n^2 + 4(-1)^n. \tag{8}$$

Continue applying this substitution until no L_n term has an exponent larger than 1.

Similarly, we could remove powers of F_n instead.

ALGORITHM "RemovePowersOfF":

Use the identity:

$$F_n^2 = \frac{L_n^2 - 4(-1)^n}{5}. \tag{9}$$

Continue applying this substitution until no F_n term has an exponent larger than 1.

4. The Simplification Algorithm.

Let us be given a polynomial function of elements of the form F_x and L_x, where the subscripts of F and L are of the form $a_1n_1 + a_2n_2 + \cdots + a_kn_k + b$ where b and the a_i are integer constants and the n_i are variables. To put this expression in "canonical form", we apply the following algorithm:

ALGORITHM "FibSimplify" TO TRANSFORM AN EXPRESSION TO CANONICAL FORM.

STEP 1: [Remove sums in subscripts]. Apply algorithm "FibReduce" to remove any sums (or differences) in subscripts.

STEP 2: [Make multipliers positive]. All subscripts are now of the form cx where c is an integer. For any term in which the multiplier c is negative, apply algorithm "FibNegate". After performing this step, you must take care to replace any expressions of the form $(-1)^{an+b}$ where a and b are constants by a properly evaluated $[(-1)^a]^n(-1)^b$, to avoid winding up with expressions like $(-1)^{2n}$ in your final canonical form.

STEP 3: [Remove multipliers]. All subscripts are now of the form cx where c is a positive integer. For any term in which the multiplier c is not 1, apply algorithm "FibReduce" successively until all subscripts are variables.

STEP 4: [Evaluate constants]. If any term involves an expression of the form F_c^k or L_c^k where c is an integer constant, use algorithm "FibEvaluate" to replace F_c and L_c by their numerical equivalents.

STEP 5: [Remove Fibonacci Powers]. If any term involves an expression of the form F_n^k where $k > 1$ and n is a variable, apply algorithm "RemovePowersOfF" to leave only linear terms in F_n.

PROVING IDENTITIES.

To prove that an expression is identically 0, apply algorithm "FibSimplify". The expression is identically 0 if and only if algorithm "FibSimplify" transforms it to 0.

This is because algorithm "FibSimplify" leaves us with a polynomial in variables of the form F_n and L_n with the degree of F_n being 0 or 1. Converting to trigonometric form using formula (7), we get a polynomial in variables of the form $\sin cx$ and $\cos cx$ with the sine terms having degree 0 or 1. Such polynomials cannot be identically 0 since it is known that all trigonometric identities are consequences of the fundamental identity $\sin^2\theta+\cos^2\theta=1$ (see, for example, [5]) and so some sine term would have to have degree 2 or more.

Example. Let us see how to prove

$$F_{n+2w}F_{n+w}-2L_wF_{n+w}F_{n-w}-F_{n-w}F_{n-2w}=(L_{3w}-2L_w)F_n^2$$

with w odd, an identity that comes from [11].

Algorithmic proof.

Since we are given that w is odd, we make the substitution $w=2m+1$ before we begin. We must show that $F_{n+4m+2}F_{n+2m+1}-2L_{2m+1}F_{n+2m+1}F_{n-2m-1}-F_{n-2m-1}F_{n-4m-2}-F_n^2(L_{6m+3}-2L_{2m+1})$ is identically 0.

First we apply algorithm "FibReduce" (step 1) to get: $(F_n^2(80F_{2m}+16L_{2m}-80F_{6m}-32L_{6m})-((F_n(-5F_{-4m}+3L_{-4m})+(3F_{-4m}-L_{-4m})L_n)(5F_{-2m}F_n-F_nL_{-2m}-F_{-2m}L_n+L_{-2m}L_n)-(5F_{2m}+L_{2m})(5F_{-2m}F_n-F_nL_{-2m}-F_{-2m}L_n+L_{-2m}L_n)(5F_{2m}F_n+F_nL_{2m}+F_{2m}L_n+L_{2m}L_n)+(5F_{2m}F_n+F_nL_{2m}+F_{2m}L_n+L_{2m}L_n)(F_n(5F_{4m}+3L_{4m})+(3F_{4m}+L_{4m})L_n))/16$.

Then we apply algorithm "FibNegate" (step 2) to get: $(80F_{2m}F_n^2+125F_{2m}^3F_n^2+50F_{2m}F_{4m}F_n^2-80F_{6m}F_n^2+16F_n^2L_{2m}+75F_{2m}^2F_n^2L_{2m}+10F_{4m}F_n^2L_{2m}+15F_{2m}F_n^2L_{2m}^2+F_n^2L_{2m}^3+30F_{2m}F_n^2L_{4m}+6F_n^2L_{2m}L_{4m}-32F_n^2L_{6m}-5F_{2m}^3L_n^2+6F_{2m}F_{4m}L_n^2-11F_{2m}^2L_{2m}L_n^2+6F_{4m}L_{2m}L_n^2-7F_{2m}L_{2m}^2L_n^2-L_{2m}^3L_n^2+2F_{2m}L_{4m}L_n^2+2L_{2m}L_{4m}L_n^2)/16$.

Applying algorithm "FibReduce" again (step 3) to get rid of scalar multiples in subscripts yields: $(80F_m^2F_n^2-125F_m^6F_n^2+160F_mF_n^2L_m-250F_m^5F_n^2L_m+16F_n^2L_m^2+25F_m^4F_n^2L_m^2+100F_m^3F_n^2L_m^3+5F_m^2F_n^2L_m^4-10F_mF_n^2L_m^5-F_n^2L_m^6)/32$.

Having evaluated constants like F_1 all along, we proceed to step 5 and apply algorithm "RemovePowersOfF". Upon expanding and gathering like terms together, we find that the result is 0. Thus our example is in fact an identity.

5. Other Algorithms.

Sometimes we want to transform an expression from one form to another, rather than put it in canonical form. The following algorithms can be used for such purposes.

ALGORITHM "ConvertToF" TO REMOVE LUCAS NUMBERS:

Use the identity:

$$L_n=F_{n-1}+F_{n+1}. \tag{10}$$

ALGORITHM "ConvertToL" TO REMOVE FIBONACCI NUMBERS:

Use the identity:

$$F_n=\frac{L_{n-1}+L_{n+1}}{5}. \tag{11}$$

THE DOUBLE ARGUMENT FORMULAS.

Letting $m = n$ in formula (5) gives us the following formulas.

$$\begin{aligned} F_{2n} &= F_n L_n \\ L_{2n} &= \frac{5F_n^2 + L_n^2}{2}. \end{aligned} \tag{12}$$

These correspond to the "double angle" formulas in trigonometry.

TO REMOVE SCALAR MULTIPLES OF ARGUMENTS IN SUBSCRIPTS:

Repeatedly apply the reduction formulas successively. In this manner, we obtain the identities:

$$\begin{aligned} F_{3n} &= \frac{3F_n L_n^2 + 5F_n^3}{4} \\ L_{3n} &= \frac{L_n^3 + 15F_n^2 L_n}{4} \\ F_{4n} &= \frac{F_n L_n (5F_n^2 + L_n^2)}{2} \\ L_{4n} &= \frac{25F_n^4 + 30F_n^2 L_n^2 + L_n^4}{8} \\ F_{5n} &= \frac{5F_n(5F_n^4 + 10F_n^2 L_n^2 + L_n^4)}{16} \\ L_{5n} &= \frac{125F_n^4 L_n + 50F_n^2 L_n^3 + L_n^5}{16}. \end{aligned} \tag{13}$$

SHORTCUT:

Apply the following recurrences:

$$\begin{aligned} F_{kn} &= \frac{F_{(k-1)n} L_n + L_{(k-1)n} F_n}{2} \\ L_{kn} &= \frac{5F_{(k-1)n} L_n + L_{(k-1)n} F_n}{2}. \end{aligned} \tag{14}$$

Even more straightforward is to use a direct formula:

$$\begin{aligned} F_{kn} &= \frac{1}{2^{k-1}} \sum_{i=0}^{\lfloor \frac{k-1}{2} \rfloor} \binom{k}{2i+1} 5^i F_n^{2i+1} L_n^{k-1-2i} \\ L_{kn} &= \frac{1}{2^{k-1}} \sum_{i=0}^{\lfloor k/2 \rfloor} \binom{k}{2i} 5^i F_n^{2i} L_n^{k-2i}. \end{aligned} \tag{15}$$

Example:

$$\begin{aligned} 256F_{9n} &= 625F_n^9 + 4500F_n^7 L_n^2 + 3150F_n^5 L_n^4 + 420F_n^3 L_n^6 + 9F_n L_n^8 \\ 256L_{9n} &= 5625F_n^8 L_n + 10500F_n^6 L_n^3 + 3150F_n^4 L_n^5 + 180F_n^2 L_n^7 + L_n^9. \end{aligned}$$

Because of the fundamental identity, these can also be put in other forms, such as the following, which converts directly to our canonical form.

$$\begin{aligned} F_{kn} &= F_n \sum_{i=0}^{\lfloor \frac{k-1}{2} \rfloor} \binom{k-1-i}{i} (-1)^{i(n+1)} L_n^{k-1-2i} \\ L_{kn} &= \sum_{i=0}^{\lfloor k/2 \rfloor} \frac{k}{k-i} \binom{k-i}{i} (-1)^{i(n+1)} L_n^{k-2i}. \end{aligned} \tag{16}$$

In general, F_{kn} cannot be replaced by sums of powers of F_n alone, so no formula of this type is given.

Also of interest are the following compact formulas.

$$\begin{aligned} F_{kn} &= \sum_{i=0}^{k} \binom{k}{i} F_n^i F_{n-1}^{k-i} F_i \\ L_{kn} &= \sum_{i=0}^{k} \binom{k}{i} F_n^i F_{n-1}^{k-i} L_i. \end{aligned} \tag{17}$$

ALGORITHM "FibExpand" TO TURN PRODUCTS INTO SUMS:

Use the identities:

$$\begin{aligned} F_m F_n &= \frac{L_{m+n} - (-1)^n L_{m-n}}{5} \\ L_m L_n &= L_{m+n} + (-1)^n L_{m-n} \\ F_m L_n &= F_{m+n} + (-1)^n F_{m-n}. \end{aligned} \tag{18}$$

To remove products of more than two terms, the above expansion formulas can be used repeatedly, expanding out the results after each step.

Removing powers is the same as removing products. For example,

$$L_n^2 = L_n \cdot L_n = L_{2n} + (-1)^n L_0.$$

Similarly for F_n^2. So we have

$$\begin{aligned} F_n^2 &= \frac{L_{2n} - 2(-1)^n}{5} \\ L_n^2 &= L_{2n} + 2(-1)^n. \end{aligned} \tag{19}$$

Continuing in this way, we find:

$$\begin{aligned}
F_n^3 &= \frac{F_{3n} - 3(-1)^n F_n}{5} \\
L_n^3 &= L_{3n} + 3(-1)^n L_n \\
F_n^4 &= \frac{L_{4n} - 4(-1)^n L_{2n} + 6}{25} \\
L_n^4 &= L_{4n} + 4(-1)^n L_{2n} + 6 \\
F_n^5 &= \frac{F_{5n} - 5(-1)^n F_{3n} + 10F_n}{25} \\
L_n^5 &= L_{5n} + 5(-1)^n L_{3n} + 10L_n \\
F_n^6 &= \frac{L_{6n} - 6(-1)^n L_{4n} + 15L_{2n} - 20(-1)^n}{125} \\
L_n^6 &= L_{6n} + 6(-1)^n L_{4n} + 15L_{2n} + 20(-1)^n
\end{aligned} \tag{20}$$

To remove higher powers, the expansion formulas (18) can be used repeatedly, expanding out the results after each step.

CHANGE OF BASIS (Shift Formulas)

ALGORITHM "FibShift" TO TRANSFORM AN EXPRESSION INVOLVING F_n, L_n INTO ONE INVOLVING F_{n+a}, L_{n+b}:
Use the identities:

$$\begin{pmatrix} F_n \\ L_n \end{pmatrix} = \frac{2}{L_a L_b - 5F_a F_b} \begin{pmatrix} L_b & -F_a \\ -5F_b & L_a \end{pmatrix} \begin{pmatrix} F_{n+a} \\ L_{n+b} \end{pmatrix}. \tag{21}$$

Proof. Solve the linear equations

$$\begin{aligned}
F_{n+a} &= \frac{1}{2}(F_a L_n + L_a F_n) \\
L_{n+b} &= \frac{1}{2}(5F_b F_n + L_b L_n)
\end{aligned}$$

for F_n and L_n.

In a similar manner, we find

$$\begin{pmatrix} F_n \\ L_n \end{pmatrix} = \frac{2}{F_a L_b - L_a F_b} \begin{pmatrix} -F_b & F_a \\ L_b & -L_a \end{pmatrix} \begin{pmatrix} F_{n+a} \\ F_{n+b} \end{pmatrix} \tag{22}$$

$$\begin{pmatrix} F_n \\ L_n \end{pmatrix} = \frac{2}{5(L_a F_b - F_a L_b)} \begin{pmatrix} -L_b & L_a \\ 5F_b & -5F_a \end{pmatrix} \begin{pmatrix} L_{n+a} \\ L_{n+b} \end{pmatrix}. \tag{23}$$

To change from an arbitrary basis to another, apply algorithm "FibReduce" to transform the given expression to the basis (F_n, L_n). Then use one of the above shift formulas.

ALGORITHM "ConvertToAlphaBeta" TO EXPRESS F_n AND L_n IN TERMS OF α AND β:

Use the well-known Binet forms:

$$\begin{aligned} F_n &= \frac{\alpha^n - \beta^n}{\sqrt{5}} \\ L_n &= \alpha^n + \beta^n \end{aligned} \tag{24}$$

to express F_n and L_n in terms of α and β, the roots of the characteristic equation $x^2 = x+1$.

ALGORITHM "RemoveAlphaBeta" TO REMOVE α's AND β's:

Sometimes a formula involves the quantities α and β. To transform such expressions into "canonical" form, apply the identites:

$$\begin{aligned} \alpha^n &= \frac{L_n + F_n\sqrt{5}}{2} \\ \beta^n &= \frac{L_n - F_n\sqrt{5}}{2}. \end{aligned} \tag{25}$$

Expand out the resulting polynomial and express it in the form $x + y\sqrt{5}$. If y is not identically 0, then the resulting expression cannot be expressed in terms of Fibonacci and Lucas numbers alone. The quantities α and β can be reintroduced into the resulting expression by means of the formula

$$\sqrt{5} = \alpha - \beta \tag{26}$$

and either α or β may be removed by use of one of the identities

$$\alpha + \beta = 1 \qquad \text{or} \qquad \alpha\beta = -1 \tag{27}$$

whichever is preferred.

TO REMOVE POWERS OF α AND β:

The following identities are frequently useful:

$$\begin{aligned} \alpha^n &= \alpha F_n + F_{n-1} \\ \beta^n &= \beta F_n + F_{n-1}. \end{aligned} \tag{28}$$

THE SUBTRACTION FORMULAS:

Combining the reduction and negation formulas gives us the following:

$$\begin{aligned} F_{m-n} &= (-1)^n \cdot \frac{F_m L_n - F_n L_m}{2} \\ L_{m-n} &= (-1)^n \cdot \frac{L_m L_n - 5F_m F_n}{2}. \end{aligned}$$

6. Arbitrary Starting Conditions.

Let $\{H_n\}$ be a sequence that satisfies

$$H_{n+2} = H_{n+1} + H_n \tag{29}$$

with initial conditions $H_0 = a$ and $H_1 = b$.

We can formally manipulate this sequence by first converting to Fibonacci numbers and then converting back.

ALGORITHM "ConvertHToF" TO EXPRESS H_n IN TERMS OF F_n AND F_{n-1}:

Use the identity:

$$H_n = aF_{n-1} + bF_n. \tag{30}$$

To prove an identity involving H_n, use equation (30) to convert H's to F's, and then apply algorithm "FibSimplify".

After applying algorithms "ConvertHToF" and "FibSimplify" to obtain a canonical form, use the following algorithm to transform the result (expressed with F's and L's) back to an expression involving H's.

ALGORITHM "ConvertToH" TO EXPRESS F_n and L_n IN TERMS OF H_n AND H_{n+1}:

Use the identities:

$$\begin{aligned} F_n &= \frac{1}{e}(bH_n - aH_{n+1}) \\ L_n &= \frac{1}{e}\left[(2b-a)H_n - (2a-b)H_{n+1}\right] \end{aligned} \tag{31}$$

where $e = b^2 - a^2 - ab$.

ALGORITHM "HExpand" TO EXPRESS H_{n+m} IN TERMS OF H_n AND H_{n+1}:

$$H_{n+m} = F_{m-1}H_n + F_mH_{n+1}. \tag{32}$$

A more symmetrical form is:

$$H_{n+m} = \frac{1}{2}(F_mG_n + L_mH_n)$$

where $G_n = H_{n-1} + H_{n+1}$.

CHANGE OF BASIS

ALGORITHM "HShift" TO TRANSFORM AN EXPRESSION INVOLVING H_n, H_{n+1} INTO ONE INVOLVING H_{n+c}, H_{n+d}:

Use the identities:

$$\begin{pmatrix} H_n \\ H_{n+1} \end{pmatrix} = \frac{1}{F_{c-1}F_d - F_cF_{d-1}} \begin{pmatrix} F_d & -F_c \\ -F_{d-1} & F_{c-1} \end{pmatrix} \begin{pmatrix} H_{n+c} \\ H_{n+d} \end{pmatrix}. \tag{33}$$

To convert expressions involving arbitrary subscripts of H to expressions involving H_{n+c} and H_{n+d}, apply algorithm "ConvertHToF" to put everything in terms of F. Then apply algorithm "FibReduce" to get everything in terms of F_n and L_n. Use algorithm "ConvertToH" to get the expression in terms of H_n and H_{n+1}; and then perform an "HShift" to transform to the desired basis.

7. Generalized Fibonacci Numbers.

We may consider the sequences $\{u_n\}$ and $\{v_n\}$ defined by the recurrences:

$$\begin{aligned} u_0 = 0, \quad u_1 = 1, \quad u_{n+2} = Pu_{n+1} - Qu_n \\ v_0 = 2, \quad v_1 = P, \quad v_{n+2} = Pv_{n+1} - Qv_n \end{aligned} \tag{34}$$

where P and Q are given constants (usually integers) with $Q \neq 0$. Let r_1 and r_2 be the roots of the characteristic equation

$$x^2 = Px - Q \tag{35}$$

so that

$$r_1 + r_2 = P \qquad \text{and} \qquad r_1 r_2 = Q, \tag{36}$$

and let

$$D = P^2 - 4Q \tag{37}$$

so that

$$r_1 = \frac{P + \sqrt{D}}{2} \qquad \text{and} \qquad r_2 = \frac{P - \sqrt{D}}{2}. \tag{38}$$

We will also assume that P and Q are chosen so that $D \neq 0$. This assures that the roots are distinct. The Binet form for u_n and v_n is

$$u_n = \frac{r_1^n - r_2^n}{r_1 - r_2}, \qquad v_n = r_1^n + r_2^n. \tag{39}$$

Powers of r_1 and r_2 can be removed from an expression by means of the formulas:

$$\begin{aligned} r_1^n &= \frac{v_n + u_n\sqrt{D}}{2} \\ r_2^n &= \frac{v_n - u_n\sqrt{D}}{2}. \end{aligned} \tag{40}$$

If a term of the form $\sqrt{D}$ is present, r_1 and r_2 can be reintroduced, if desired, by the formula

$$r_1 - r_2 = \sqrt{D}\,. \tag{41}$$

Also of interest are the identities:

$$\begin{aligned} r_1^2 &= Pr_1 - Q \\ r_2^2 &= Pr_2 - Q \end{aligned} \tag{42}$$

and

$$\begin{aligned} r_1^n &= r_1 u_n - Q u_{n-1} \\ r_2^n &= r_2 u_n - Q u_{n-1}. \end{aligned}$$

For these sequences, we have the following algorithms (from [6]):

ALGORITHM "LucasNegate" TO REMOVE NEGATIVE SUBSCRIPTS:
Use the identities:

$$\begin{aligned} u_{-n} &= -\frac{u_n}{Q^n} \\ v_{-n} &= \frac{v_n}{Q^n}. \end{aligned} \tag{43}$$

ALGORITHM "LucasReduce" TO REMOVE SUMS IN SUBSCRIPTS:
Use the identities:

$$\begin{aligned} u_{m+n} &= \frac{u_m v_n + u_n v_m}{2} \\ v_{m+n} &= \frac{v_m v_n + D u_m u_n}{2}. \end{aligned} \tag{44}$$

We also have the "subtraction formulas":

$$\begin{aligned} u_{m-n} &= \frac{u_m v_n - u_n v_m}{2Q^n} \\ v_{m-n} &= \frac{v_m v_n - D u_m u_n}{2Q^n}. \end{aligned} \tag{45}$$

THE FUNDAMENTAL IDENTITY:
The fundamental identity that connects u_n and v_n is:

$$v_n^2 - D u_n^2 = 4Q^n. \tag{46}$$

This can be used to give us the following three algorithms.

ALGORITHM "RemovePowersOfV":
Apply the substitution

$$v_n^2 = D u_n^2 + 4Q^n \tag{47}$$

repeatedly until the exponent of all v_n terms is less than 2.

ALGORITHM "RemovePowersOfU":

Apply the substitution

$$u_n^2 = \frac{v_n^2 - 4Q^n}{D} \tag{48}$$

repeatedly until the exponent of all u_n terms is less than 2.

ALGORITHM "RemovePowersOfQ":

Apply the substitution

$$Q^n = \frac{v_n^2 - Du_n^2}{4}. \tag{49}$$

ALGORITHM "LucasExpand" TO TURN PRODUCTS INTO SUMS:

Use the identities:

$$\begin{aligned} u_m u_n &= \frac{v_{m+n} - Q^n v_{m-n}}{D} \\ v_m v_n &= v_{m+n} + Q^n v_{m-n} \\ u_m v_n &= u_{m+n} + Q^n u_{m-n}. \end{aligned} \tag{50}$$

This algorithm can be repeated to turn products of any number of u and v terms into simple sums of such terms.

ALGORITHM "ConvertToU" TO REMOVE ALL v's:

Use the identity:

$$v_n = u_{n+1} - Qu_{n-1}. \tag{51}$$

ALGORITHM "ConvertToV" TO REMOVE ALL u's:

Use the identity:

$$u_n = \frac{v_{n+1} - Qv_{n-1}}{D}. \tag{52}$$

RELATIONSHIP TO TRIGONOMETRIC EXPRESSIONS:

To convert to trigonometric form, use the identities:

$$\begin{aligned} u_n &= 2Q^{n/2} \sin\left(\frac{ni}{2} \log \frac{r_1}{r_2}\right) / \sqrt{-D} \\ v_n &= 2Q^{n/2} \cos\left(\frac{ni}{2} \log \frac{r_1}{r_2}\right). \end{aligned} \tag{53}$$

TO REMOVE SCALAR MULTIPLES IN SUBSCRIPTS:

Repeatedly apply algorithm "LucasReduce" or use the identities:

$$\begin{aligned} u_{kn} &= u_n \sum_{i=0}^{\lfloor \frac{k-1}{2} \rfloor} \binom{k-1-i}{i} (-1)^i Q^{in} v_n^{k-1-2i} \\ v_{kn} &= \sum_{i=0}^{\lfloor k/2 \rfloor} \frac{k}{k-i} \binom{k-i}{i} (-1)^i Q^{in} v_n^{k-2i}. \end{aligned} \tag{54}$$

These formulas convert directly to our canonical form.

In particular, the double argument formulas are:

$$\begin{aligned} u_{2n} &= u_n v_n \\ v_{2n} &= v_n^2 - 2Q^n. \end{aligned} \tag{55}$$

Other formulas of interest:

$$u_{kn} = \frac{1}{2^{k-1}} \sum_{i=0}^{\lfloor \frac{k-1}{2} \rfloor} \binom{k}{2i+1} D^i u_n^{2i+1} v_n^{k-1-2i}$$

$$v_{kn} = \frac{1}{2^{k-1}} \sum_{i=0}^{\lfloor k/2 \rfloor} \binom{k}{2i} D^i u_n^{2i} v_n^{k-2i}$$

$$u_{kn+s} = \sum_{i=0}^{k} \binom{k}{i} u_n^i (-Q u_{n-1})^{k-i} u_{i+s}$$

$$v_{kn+s} = \sum_{i=0}^{k} \binom{k}{i} u_n^i (-Q u_{n-1})^{k-i} v_{i+s}.$$

TO REMOVE POWERS:

Repeatedly apply algorithm "LucasExpand" or use the identities:

$$\begin{aligned} u_n^k &= \frac{1}{2} \frac{1}{D^{\lfloor k/2 \rfloor}} \sum_{i=0}^{k} \binom{k}{i} (-1)^i Q^{in} \times \begin{cases} u_{(k-2i)n}, & \text{if } k \text{ is odd,} \\ v_{(k-2i)n}, & \text{if } k \text{ is even;} \end{cases} \\ v_n^k &= \frac{1}{2} \sum_{i=0}^{k} \binom{k}{i} Q^{in} v_{(k-2i)n}. \end{aligned} \tag{56}$$

Any negative subscripts introduced can be removed by using algorithm "LucasNegate", if desired.

CHANGE OF BASIS (Shift Formulas)

ALGORITHM "LucasShift" TO TRANSFORM AN EXPRESSION INVOLVING u_n, v_n INTO ONE INVOLVING u_{n+a}, v_{n+b}:

Use the identities:

$$\begin{pmatrix} u_n \\ v_n \end{pmatrix} = \frac{2}{v_a v_b - D u_a u_b} \begin{pmatrix} v_b & -u_a \\ -D u_b & v_a \end{pmatrix} \begin{pmatrix} u_{n+a} \\ v_{n+b} \end{pmatrix}. \tag{57}$$

PROVING IDENTITIES.

To effectively prove an identity involving u's and v's, perform the following algorithm which is analogous to algorithm "FibSimplify".

ALGORITHM "LucasSimplify":

Use algorithm "LucasReduce" to remove any sums in subscripts. Make all the multipliers in subscripts positive by using algorithm "LucasNegate". Use "LucasReduce" again to remove any constant multipliers in the subscripts. Evaluate any numerical terms using the recurrence (34) and its initial values (applying algorithm "LucasNegate" first for negative subscripts). Finally, use algorithm "RemovePowersOfU" and collect terms to arrive at a canonical form.

Note that if the variables P, Q, and D all occur in the final expression, you should replace D by its equivalent value, $P^2 - 4Q$, from formula (37).

The expression is identically 0 if and only if this canonical form is 0.

8. Arbitrary Second-Order Linear Recurrences.

Let w_n be an arbitrary second-order linear recurrence with constant coefficients. That is,

$$w_n = P w_{n-1} - Q w_{n-2}, \qquad n \geq 2 \tag{58}$$

with arbitrary initial values w_0 and w_1, not both 0. Again we require that $Q \neq 0$ and $P^2 \neq 4Q$. Since u_n and v_n form a basis among all such sequences, w_n can be expressed in terms of u_n and v_n. This can be accomplished by the following formula:

$$w_n = (w_1 - P\frac{w_0}{2})u_n + \frac{w_0}{2} v_n. \tag{59}$$

Thus, to prove any identity involving w's, first convert the w's to u's and v's and then apply algorithm "LucasSimplify". An equivalent formula is

$$w_n = (w_1 - P w_0)u_n + w_0 u_{n+1} = -Q w_{-1} u_n + w_0 u_{n+1}.$$

The reduction formula takes the form

$$w_{n+m} = -Q w_{m-1} u_n + w_m u_{n+1} = -Q w_m u_{n-1} + w_{m+1} u_n.$$

9. Summations.

We can perform indefinite summations of expressions involving u_n and v_n any time we can perform such summations with x^n instead, since from the Binet forms, these terms are actually exponentials with bases r_1 and r_2.

ALGORITHM "LucasSum":

First, the expression is converted to exponential form using equation (39). Then it is summed. The result is converted back to u's and v's by using equation (40). Then r_1 and r_2 are converted to expressions involving P and Q using equations (38) and (36).

The reader who has not kept abreast of developments in the area of symbolic algebra may be surprised at the large class of functions that can now be automatically summed. Gosper's algorithm [3] can be used to sum complicated expressions involving hypergeometric series. This includes most of the common elementary functions. See [4] pp. 224–230 for an elementary exposition.

Example. Let us see how to sum $\sum_{k=1}^{n} L_k \cos kx$ algorithmically. Expanding L_k by the Binet form (24), and expanding $\cos kx$ by the formula $\cos x = (e^{ix} + e^{-ix})/2$, turns the sum into four terms each of which is a simple geometric progression. These are easily summed. The result is turned back into a nice form by using formula (25) and the formula $e^{ix} = \cos x + i \sin x$. The result is the following, which we believe to be new.

$$\sum_{k=1}^{n} L_k \cos kx = \frac{1}{2\cos 2x - 3}\Big[5 + 2\cos x - 4\cos^2 x - F_n(2\cos x - 5)\cos(n+1)x + F_{n+1}((2\cos x - 3)\cos nx + 2\cos(n+2)x) - 4F_{n+2}\cos x \cos nx\Big]. \tag{60}$$

10. Discovering Identities.

For the most part, the algorithms described in this paper can only be used to prove an identity once the identity is known or suspected. It still requires human ingenuity to discover new and interesting identities.

There are, however, several ways these algorithms can be used to aid in discovering new identities.

Algorithm "LucasSum" and its Fibonacci counterpart, algorithm "FibSum", are effective algorithms for summing expressions. Thus, they can be used, as in the previous example, to discover new results.

If the form of an identity is suspected, algorithm "FibSimplify" may frequently be used to advantage to discover a new result. For example, suppose that you suspect that F_{n+3}^2 can be written as a linear combination of F_{n+2}^2, F_{n+1}^2, and F_n^2. In that case, you can use algorithm "FibSimplify" to convert $F_{n+3}^2 - aF_{n+2}^2 - bF_{n+1}^2 - cF_n^2$ into canonical form. The result is

$$(-1)^n(-16 + 9a + b + 4c)/5 + L_n^2(18 - 7a - 3b - 2c)/10 + F_nL_n(8 - 3a - b)/2.$$

Since the original expression is an identity if and only if this canonical form is identically 0, we must have the three equations

$$\begin{aligned} 9a + b + 4c &= 16 \\ 7a + 3b + 2c &= 18 \\ 3a + b &= 8. \end{aligned}$$

Solving these equations yields $a = 2$, $b = 2$, and $c = -1$. This reveals the identity

$$F_{n+3}^2 - 2F_{n+2}^2 - 2F_{n+1}^2 + F_n^2 = 0. \tag{61}$$

11. Implementation of the Algorithms.

All the algorithms described in this paper have been implemented in Mathematica™. They are available from the author by email. Similar algorithms exist for third-order linear recurrences [8] and higher-order recurrences.

12. Philosophical Implications.

Now that these effective algorithms for proving Fibonacci identities are known, the philosophical question arises as to whether one should study and/or publish newly discovered identities. The answer is "yes". Although formula (1) can now be proven by computer, the computer proof gives no insight into how the formula came about. What is the significance of the binomial coefficients appearing in the formula? Does the formula generalize? Can a similar proof be used to discover new results?

Should one stop submitting pretty new identities to problem columns? The answer is "no". There is always the challenge involved in finding an elegant proof. For example, an algorithmic proof of the trigonometric identity $\sin^2 17x + \cos^2 17x = 1$ would not recognize that it follows from the Pythagorean Theorem. Instead, it would expand out by the multiple angle formula and get a horrendous mess which it would then attempt to show is divisible by $(\sin^2 x + \cos^2 x - 1)$.

Elegant new identities will always be welcome to journals such as *The Fibonacci Quarterly*. Discovering new identities is still important. Ingenious proofs are always appreciated. What these algorithms imply, is that the next time you need a complicated identity as a lemma in the middle of a research paper, you can safely state the identity and say that "This identity is straightforward to prove."

References

[1] David E. Dobbs, "Proving Trig. Identities to Freshpersons", *The MATYC Journal*. **14** (1980):39–42.

[2] David E. Dobbs and Robert Hanks, *A Modern Course on the Theory of Equations*. Passaic, NJ: Polygonal Publishing House, 1980.

[3] R. William Gosper, Jr., "Decision Procedure for Indefinite Hypergeometric Summation", *Proceedings of the National Academy of Sciences of the United States of America*. **75** (1978):40–42.

[4] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik, *Concrete Mathematics*. Reading, MA: Addison-Wesley Publishing Company, 1989.
[5] M. S. Klamkin, "On Proving Trigonometric Identities", *Mathematics Magazine*. **56** (1983):215–220.
[6] Edouard Lucas, "Théorie des Fonctions Numériques Simplement Périodiques", *American Journal of Mathematics*. **1** (1878):184–240, 289–321; reprinted as "The Theory of Simply Periodic Numerical Functions", Santa Clara, CA: The Fibonacci Association, 1969.
[7] Roman Maeder, *The Mathematica Programmer*. Boston, MA: Academic Press, 1994.
[8] Stanley Rabinowitz, "Algorithmic Manipulation of Third-Order Linear Recurrences", *The Fibonacci Quarterly*. **34** (1996):447–464.
[9] S. Vajda, *Fibonacci & Lucas Numbers, and the Golden Section*. West Sussex, England: Ellis Horwood Limited, 1989.
[10] Gregory Wulczyn, "Problem H-324", *The Fibonacci Quarterly*. **19** (1981):93.
[11] Gregory Wulczyn, "Problem B-464", *The Fibonacci Quarterly*. **20** (1982):370.

AMS Classification Numbers: 11Y16, 11B39, 11B37