A translation method for finding combinatorial bijections

Philip Matchett Wood
Department of Mathematics, Rutgers University (New Brunswick), Hill Center-Busch Campus, 110 Frelinghuysen Rd, Piscataway, NJ 08854-8019, USA
matchett@math.rutgers.edu

Doron Zeilberger
Department of Mathematics, Rutgers University (New Brunswick), Hill Center-Busch Campus, 110 Frelinghuysen Rd, Piscataway, NJ 08854-8019, USA
zeilberg@math.rutgers.edu

February 2, 2007

Abstract

Consider a combinatorial identity that can be proved by induction. In this paper, we describe a general method for translating the inductive proof into a recursive bijection. Furthermore, we will demonstrate that the resulting recursive bijection can often be defined in a direct, non-recursive way. Thus, the translation method often results in a bijective proof of the identity that helps illuminate the underlying combinatorial structures.

This paper has two main parts: first, we describe the translation method and accompanying Maple code; and second, we give a few examples of how the method has been used to discover new bijections.

1 Introduction

In 1981 Garsia and Milne [2, 3] broke new ground by translating an analytic proof of an identity into a bijective proof (see [8] for a brief exposition). Inspired by their work, we will describe a method in this paper for translating an inductive proof into a bijective proof.

In this paper we will consider families of integer identities parameterized by an integer n. For example, one can think of the following identity for concreteness (see Identity 7 from [1]):

$$3f_n = f_{n+2} + f_{n-2} \qquad \text{for } n \geq 1, \tag{1}$$

where f_k is the k-th Fibonacci number, i.e. $f_k = f_{k-1} + f_{k-2}$ for $k \geq 2$ and $f_0 = f_1 = 1$, so $(f_0, f_1, f_2, f_3, f_4, f_5, \ldots) = (1, 1, 2, 3, 5, 8, \ldots)$ (our definition of f_k follows the combinatorial convention and should not to be confused with the number-theorical and more common definition F_k, which takes $F_1 = 1$ and $F_2 = 1$, and so $f_k = F_{k+1}$). Given an integer identity, there is sometimes a combinatorial interpretation for each side of the equation; that is, each side of the equation is the cardinality of a set of combinatorial objects. For example, a combinatorial interpretation for f_k is the cardinality of the set $\mathcal{F}_k$, which consists of all finite sequences of 1's and 2's that sum to k. Thus, one can interpret the left hand side of Equation (1) as the cardinality of the set $\{1, 2, 3\} \times \mathcal{F}_n$

and one can interpret the right hand side as the cardinality of the set $\mathcal{F}_{n+2} \uplus \mathcal{F}_{n-2}$, where $\uplus$ is disjoint union.

Given a combinatorial interpretation for an identity, it is often desirable to find a bijection between the sets of objects counted by each side of the equation. For example, in Equation (1) we would like to find a bijection

$$\{1,2,3\} \times \mathcal{F}_n) \longrightarrow^{\sim} \mathcal{F}_{n+2} \uplus \mathcal{F}_{n-2} \ ,$$

and we say that such a bijection provides a *combinatorial proof* (or *bijective proof*) of the identity. Of course, we are considering a family of identities parameterized by n, and so a bijective proof actually consists of a family of bijections also parameterized by n. Throughout this paper, we will use "identity" to refer to a family of identities and "bijection" to refer to a corresponding family of bijections.

In Section 2, we describe a method that, given a combinatorial identity with an inductive proof, produces a recursive bijection for the identity. By *recursive bijection*, we mean that the definition of the k-th bijection in the family depends on the definitions of the i-th bijections for some values of $i < k$. On the other hand, in a *direct bijection*, the k-th bijection in the family does *not* depend on the bijection for smaller cases, but instead is defined purely by the underlying structures of the relevant sets. When seeking a bijective proof of an identity, we always want to find a direct bijection, not a recursive one.

One should note that we did not attempt to define "direct bijection" and "recursive bijection" in a completely formal, logically rigorous way. Rather, the purpose of these terms used to informally distinguish between two sorts of bijections: recursively defined bijections, which can be cumbersome and hard to understand; and directly defined bijections, which are natural, elegant, and clever. Of course, our goal is to discover bijections of the latter sort. The purpose of this paper is to show that our translation method often leads to natural direct bijections, with a recursively defined bijection being created as an intermediate step.

In Section 3, we will demonstrate through a number of examples that often the recursive bijection translated from an inductive proof can be defined as a natural direct bijection; thus providing a bijective proof for the combinatorial identity. Finding the non-recursive definition (if one exists) of a recursive bijection usually requires studying the behavior of the bijection on small cases. It is interesting that recursive bijections constructed via the translation method are often easy to define as direct bijections, and heuristically this seems to indicate that an inductive proof of a combinatorial identity may depend on the underlying combinatorial structure in a subtle way.

Our translation method for constructing a recursive bijection from an inductive proof is algorithmic. In particular, each step in the inductive proof—addition, multiplication, subtraction, and applying the inductive hypothesis—is translated into a corresponding bijection; and then these bijections are composed to complete the combinatorial proof. Addition corresponds to disjoint union, multiplication to Cartesian set product, and induction to recursion. Subtraction is perhaps the trickiest operation to deal with; however, the alternating paths algorithm gives a natural corresponding bijection. Using the alternating paths algorithm in this way can be traced to an "involution principle" underlying the work of Garsia and Milne (see [2] and [3]).

Our paper is accompanied by four Maple packages `BijTools`, `Examples`, `Fibonacci`, and `TransMethodZeck` (see [6]). `BijTools` provides general Maple functions for constructing bijections via the translation method, and `Fibonacci` provides some basic functions for the Fibonacci numbers including the standard combinatorial interpretation. `Examples` implements the recursive bijections constructed in Sections 2 and 3 along with the corresponding natural direct bijections. Finally,

`TransMethodZeck` implements the recursive bijections described in Subsection 3.7, for which the corresponding natural direct bijections are described in [5] and implemented in `ZeckFibBijections`. The Maple code may be found online at

`http://www.math.rutgers.edu/~matchett/Publications/Maple.html`

In Subsection 2.1, we will demonstrate the translation method on Equation (1), eventually arriving at a natural bijective proof. In Subsection 2.2, we will describe the translation method in general, and in Subsection 2.3, we will give a primer on the Maple code written for this paper. In Section 3, we discuss three applications of the translation method to previously open problems: Cassini's Fibonacci Identity (see Subsection 3.1) which was first proven bijectively in [7]; a Fibonacci identity for which finding a bijection was stated as an open problem in [1] (see Subsection 3.2) and which was first proven bijectively in [4]; and certain Zeckendorf Family Identities (see Sections 3.3, 3.4, 3.5, 3.6, and 3.7), also from [1], for which the first bijective proofs were found in [5] (note Equation (1) is a simple example of a Zeckendorf Family Identity). Constructing recursive bijections that lead to the most natural direct bijections for a Zeckendorf Family Identity requires a combinatorial interpretation of the Fibonacci numbers with negative indices (see Subsections 3.3, 3.4, and 3.5). The basic idea is that if we start with smaller base cases (which have negative indices), then there will be fewer arbitrary choices, resulting in a more natural bijection. In Subsection 3.6 we revisit the motivating example and Subsection 2.1 (which is a simple example of a Zeckendorf Family Identity) and construct a unique recursive bijection using negative indices, and in Subsection 3.7, we discuss how to construct bijective proofs for general Zeckendorf Family Identities.

2 The translation method

2.1 A motivating example

We will begin by demonstrating the translation method on Equation (1), which has the following easy inductive proof:

$$\begin{aligned} 3f_n &= 3f_{n-1} + 3f_{n-2} && \text{(definition of Fibonacci numbers)} && (2)\\ &= f_{n+1} + f_{n-3} + f_n + f_{n-4} && \text{(induction hypothesis, twice)} && (3)\\ &= f_{n+2} + f_{n-2} && \text{(definition of Fibonacci numbers, twice).} && (4) \end{aligned}$$

The base cases of $n = 1$ and $n = 2$ are also easy: $3 \cdot 1 = 3f_1 = f_{-1} + f_3 = 0 + 3$ (note that defining $f_{-1} = 0$ is consistent with the other Fibonacci numbers, and interpreting $\mathcal{F}_{-1}$ as the empty set is then natural), and $3 \cdot 2 = 3f_2 = f_0 + f_4 = 1 + 5$.

Notice that each of Equations (2), (3), and (4) is a combinatorial identity, for example, Equation (4) is the identity $f_{n+1} + f_{n-3} + f_n + f_{n-4} = f_{n+2} + f_{n-2}$. Thus, if we can find bijections for Equations (2), (3), and (4), we can compose them to get a bijection $\Phi_n : [3] \times \mathcal{F}_n \to \mathcal{F}_{n+2} \uplus \mathcal{F}_{n-2}$ for Equation (1). The resulting bijection will, of course, be recursive, since the bijection for Equation (3) will simply consist of applying recursion twice. We will now construct bijections for Equations (2), (3), and (4) explicitly, using the notation from the introduction.

For Equation (2), the translation of $3f_n = 3f_{n-1} + 3f_{n-2}$ into sets is:

$$\phi_{(2),n} : \{1, 2, 3\} \times \mathcal{F}_n \longrightarrow \{1, 2, 3\} \times \mathcal{F}_{n-1} \uplus \{1, 2, 3\} \times \mathcal{F}_{n-2}.$$

The bijection $\phi_{(2),n}$ can easily be thought of as the "multiplication" (via Cartesian product) of the identity bijection on $\{1, 2, 3\}$ with the defining bijection for the Fibonacci numbers (which is $\mathfrak{d}_n$:

$\mathcal{F}_n \to \mathcal{F}_{n-1} \uplus \mathcal{F}_{n-2}$ defined by deleting the last element in the list, i.e. $[\![\ell_1, \ldots, \ell_r]\!] \mapsto [\![\ell_1, \ldots, \ell_{r-1}]\!]$). Thus, for $i \in \{1,2,3\}$ and $L = [\![\ell_1, \ldots, \ell_r]\!] \in \mathcal{F}_n$, we define

$$\phi_{(2),n} : (i, L) \mapsto (i, [\![\ell_1, \ldots, \ell_{r-1}]\!]).$$

For Equation (3), the translation of $3f_{n-1} + 3f_{n-2} = f_{n+1} + f_{n-3} + f_n + f_{n-4}$ into sets is

$$\phi_{(3),n} : \{1,2,3\} \times \mathcal{F}_{n-1} \uplus \{1,2,3\} \times \mathcal{F}_{n-2} \longrightarrow \mathcal{F}_{n+1} \uplus \mathcal{F}_{n-3} \uplus \mathcal{F}_n \uplus \mathcal{F}_{n-4},$$

which may easily be defined by "adding" (via disjoint union) the two recursively defined bijections

$$\begin{aligned} \Phi_{n-1} :& \{1,2,3\} \times \mathcal{F}_{n-1} \longrightarrow \mathcal{F}_{n+1} \uplus \mathcal{F}_{n-3} \qquad \text{and} \\ \Phi_{n-2} :& \{1,2,3\} \times \mathcal{F}_{n-2} \longrightarrow \mathcal{F}_n \uplus \mathcal{F}_{n-4}. \end{aligned}$$

Thus, for (i, L) in $\{1,2,3\} \times \mathcal{F}_{n-1} \uplus \{1,2,3\} \times \mathcal{F}_{n-2}$, we define

$$\phi_{(3),n} : (i, L) \mapsto \begin{cases} \Phi_{n-1}((i, L)) & \text{if } L \in \mathcal{F}_{n-1}, \text{ and} \\ \Phi_{n-2}((i, L)) & \text{if } L \in \mathcal{F}_{n-2}. \end{cases}$$

For Equation (4), the translation of $f_{n+1} + f_{n-3} + f_n + f_{n-4} = f_{n+2} + f_{n-2}$ into sets is:

$$\phi_{(4),n} : \mathcal{F}_{n+1} \uplus \mathcal{F}_{n-3} \uplus \mathcal{F}_n \uplus \mathcal{F}_{n-4} \longrightarrow \mathcal{F}_{n+2} \uplus \mathcal{F}_{n-2},$$

which is naturally defined by "adding" (via disjoint union) the inverse of the defining bijection for the Fibonacci numbers for $n+2$ and $n-2$. That is, we "add" the two bijections:

$$\begin{aligned} \mathfrak{d}_{n+2}^{-1} :& \mathcal{F}_n \uplus \mathcal{F}_{n+1} \longrightarrow \mathcal{F}_{n+2} \qquad \text{and} \\ \mathfrak{d}_{n-2}^{-1} :& \mathcal{F}_{n-4} \uplus \mathcal{F}_{n-3} \longrightarrow \mathcal{F}_{n-2}. \end{aligned}$$

Thus, for L in $\mathcal{F}_{n+1} \uplus \mathcal{F}_{n-3} \uplus \mathcal{F}_n \uplus \mathcal{F}_{n-4}$, we define

$$\phi_{(4),n} : L \mapsto \begin{cases} \mathfrak{d}_{n+2}^{-1}(L) & \text{if } L \in \mathcal{F}_n \uplus \mathcal{F}_{n+1}, \text{ and} \\ \mathfrak{d}_{n-2}^{-1}(L) & \text{if } L \in \mathcal{F}_{n-4} \uplus \mathcal{F}_{n-3}. \end{cases}$$

Finally, we can compose to get a bijection for Equation (1), namely $\Phi_n := \phi_{(4),n} \circ \phi_{(3),n} \circ \phi_{(2),n}$.

Of course, since Φ_n is a recursive bijection, we must also define the base cases Φ_1 and Φ_2. Define:

$$\Phi_1 : \begin{array}{l} (1, [\![1]\!]) \mapsto [\![1,1,1]\!] \\ (2, [\![1]\!]) \mapsto [\![2,1]\!] \\ (3, [\![1]\!]) \mapsto [\![1,2]\!] \end{array} \qquad \text{and} \qquad \Phi_2 : \begin{array}{lcl} (1, [\![1,1]\!]) & \mapsto & [\![1,1,1,1]\!] \\ (1, [\![2]\!]) & \mapsto & [\![1,1,2]\!] \\ (2, [\![1,1]\!]) & \mapsto & [\![2,1,1]\!] \\ (2, [\![2]\!]) & \mapsto & [\![2,2]\!] \\ (3, [\![1,1]\!]) & \mapsto & [\![1,2,1]\!] \\ (3, [\![2]\!]) & \mapsto & [\![]\!]. \end{array}$$

Note that the choices for these base cases may seem arbitrary (out of $3! \cdot 6!$ possibilities); however, in Subsection 3.3, we will show that these base cases are the only "natural" choices in a certain sense.

At this point, having defined a recursive bijection Φ_n for Equation (1), we should study the behavior of Φ_n for small n in hopes of finding a non-recursive definition for Φ_n. While of course

Figure 1

```
# The code for the base cases (n = 1 and n = 2, respectively):

Id7egBase1:=proc() local B:
B:=table([
    [1,[1]] = [1,1,1],
    [2,[1]] = [2,1],
    [3,[1]] = [1,2]
]);
B:
end:

Id7egBase2:=proc() local B:
B:= table([ [1,[1,1]] = [1,1,1,1],
            [1,[2]]   = [1,1,2],
            [2,[1,1]] = [2,1,1],
            [2,[2]]   = [2,2],
            [3,[1,1]] = [1,2,1],
            [3,[2]]   = []
]);
B:
end:

# The code for the recursive cases (n ≥ 2):

Id7eg:=proc(n) local B;
if n=1 then
    return Id7egBase1():
elif n=2 then
    return Id7egBase2():
elif n>2 then
## Induction Step ##
    # 3 f_n = 3 f_{n-1} + 3 f_{n-2}
    B[1]:=MultBij(IdBij({1,2,3}), FibDef(n));

    # 3 f_{n-1} + 3 f_{n-2} = f_{n+1} + f_{n-3} + f_n + f_{n-4}
    B[3]:=AddBij(Id7eg(n-1), Id7eg(n-2)):

    # f_{n-3} + f_{n-4} + f_n + f_{n+1} = f_{n-2} + f_{n+2}
    B[4]:=InvBij(AddBij(FibDef(n-2),FibDef(n+2))):

    #       3 f_n = 3 f_{n-1} + 3 f_{n-2}               \\
    #             = f_{n+1} + f_{n-3} + f_n + f_{n-4}   \\
    #             = f_{n-2} + f_{n+2}
    B[5]:=ComposeBij(B[4],ComposeBij(B[3],B[1])):
    return B[5]:
fi:
# function should never reach this point.
return FAIL:
end:
```

Figure 1: Above is the Maple code used to generate a recursive bijection for Equation (1). Notice how closely the inductive step (starting below "`## Induction Step ##`" near the middle of this page) mirrors the inductive proof in Equations (2), (3), and (4). The bijection, called `Id7eg`, is stored as a table in Maple, thus a line of the form "`[1,[1]] = [1,1,1]`" means that in the bijection, $(1, [\![1]\!]) \mapsto [\![1, 1, 1]\!]$. Also, `Id7eg` calls the two base cases as sub-functions. See [6, `Examples`] for the full code.

this could be done by hand, in practice it is easier to generate the data with a computer. The code in Figure 1 was used to generate the output below for the case $n = 3$ (see [6, `Examples`] for the full code):

$$\Phi_3 : \begin{array}{lcl} (1, [\![1,1,1]\!]) & \mapsto & [\![1,1,1,1,1]\!] \\ (1, [\![1,2]\!]) & \mapsto & [\![1,1,1,2]\!] \\ (1, [\![2,1]\!]) & \mapsto & [\![1,1,2,1]\!] \\ (2, [\![1,1,1]\!]) & \mapsto & [\![2,1,1,1]\!] \\ (2, [\![1,2]\!]) & \mapsto & [\![2,1,2]\!] \\ (2, [\![2,1]\!]) & \mapsto & [\![2,2,1]\!] \\ (3, [\![1,1,1]\!]) & \mapsto & [\![1,2,1,1]\!] \\ (3, [\![1,2]\!]) & \mapsto & [\![1,2,2]\!] \\ (3, [\![2,1]\!]) & \mapsto & [\![1]\!] \end{array}$$

From the output for Φ_3 and a few other small cases, it is not hard to guess that Φ_n can be defined as a direct bijection as follows:

$$\Phi_n : (i, [\![\ell_1, \ldots, \ell_r]\!]) \mapsto \begin{cases} [\![1, 1, \ell_1, \ldots, \ell_r]\!] & \text{if } i = 1 \\ [\![2, \ell_1, \ldots, \ell_r]\!] & \text{if } i = 2 \\ [\![1, 2, \ell_2, \ldots, \ell_r]\!] & \text{if } \ell_1 = 1 \text{ and } i = 3 \\ [\![\ell_2, \ldots, \ell_r]\!] & \text{if } \ell_1 = 2 \text{ and } i = 3. \end{cases} \tag{5}$$

It is easy to see that Display (5) does indeed define a direct bijection, and thus provides a bijective proof of Equation (1). In fact, the direct bijection in Display (5) is the same as that described in [1, page 6] (except in [1], the output lists are all reversed compared to our notation). The Maple code [6, `Examples`] includes an implementation of the directly-defined bijection Φ_n, and one can verify for any specific n that the two definitions lead to the same bijection using the function ts`BijEqual` in [6, `BijTools`] (of course, the algorithm for constructing the recursive bijection is exponential, so it is impractical to use `BijEqual` for n much larger than 15).

2.2 A general description of the translation method

Recall that the translation method converts each step of an inductive proof into a set-theoretic analog. Consider two bijections $\alpha : A \to B$ and $\gamma : C \to D$ where the cardinalities of A, B, C, and D are, respectively, a, b, c, and d. Below we will describe how to translate statements in an inductive proof into bijections.

Addition becomes disjoint union. Consider the statement "$a = b$ and $c = d$ implies $a+c = b+d$". In set-theoretic terms, this can be phrased

$$A \xrightarrow[\alpha]{\sim} B \text{ and } C \xrightarrow[\gamma]{\sim} D \quad \text{implies} \quad A \uplus C \xrightarrow[\sigma]{\sim} B \uplus D\,,$$

where we define $\sigma(x) := \begin{cases} \alpha(x) & \text{if } x \in A \\ \gamma(x) & \text{if } x \in C \end{cases}$.

Multiplication becomes Cartesian product. Consider the statement "$a = b$ and $c = d$ implies $ac = bd$". In set-theoretic terms, this becomes

$$A \xrightarrow[\alpha]{\sim} B \text{ and } C \xrightarrow[\gamma]{\sim} D \quad \text{implies} \quad A \times C \xrightarrow[\sigma]{\sim} B \times D\,,$$

where we define $\sigma(x, y) = (\alpha(x), \gamma(y))$ for $x \in A$ and $y \in C$.

Induction becomes recursion. At the step in the inductive proof that uses the induction hypothesis, we simply have the bijection that we are trying to define, say Φ_n, call itself recursively at values $i_1, \ldots, i_r$, each smaller than n.

Subtraction becomes combinatorial inversion via the alternating paths algorithm. Assume that $c \leq a$ and $d \leq b$ and consider the statement "$a = b$ and $c = d$ implies $a - c = b - d$". In set-theoretic terms, this becomes:

$$\begin{aligned} &\text{Assume } C \subset A \text{ and } D \subset B. \\ &\text{Then } A \xrightarrow[\alpha]{\sim} B \text{ and } C \xrightarrow[\gamma]{\sim} D \text{ implies } A \setminus C \xrightarrow[\sigma]{\sim} B \setminus D\,, \end{aligned} \tag{6}$$

where σ is defined using the alternating paths algorithm (see next paragraph). The idea of using the alternating paths algorithm in this way may be traced to an implicit "involution principle" that underlies the work of Garsia and Milne [2, 3].

We now define the bijection σ in Display (6). Specifically, let $a \in A \setminus C$. Then we define $\sigma(a) := (\alpha\gamma^{-1})^k \alpha(a)$, where

$$(\alpha\gamma^{-1})^k = \underbrace{\alpha\gamma^{-1} \circ \alpha\gamma^{-1} \circ \cdots \circ \alpha\gamma^{-1}}_{k \text{ times}}$$

and k is the smallest non-negative integer such that $(\alpha\gamma^{-1})^k \alpha(a) \in B \setminus D$. It is easy to see that σ is well-defined and a bijection, and we give a pictorial representation of σ in Figure 2. This process for defining σ is known as the alternating paths algorithm.

The recursive definition of a bijection derived via the translation method can often be quite ungainly. The goal of the translation method is to study the output of such a recursive bijection (often with the help of a computer) and find an elegant and direct definition for the same bijection. In Section 3 we will demonstrate a number of examples were the translation method does, in fact, lead to a natural direct bijection.

2.3 Primer on accompanying Maple code

In this subsection, we will describe the main parts of the Maple code that accompanies this paper. We will focus on the general tools in `BijTools`, and a few functions relating to the Fibonacci numbers in `Fibonacci` (see [6]).

For the purposes of the Maple implementation, a bijection is simply a table, with the indices forming the domain and the entries forming the range. For example,

```
B:= table( [ a = 1,
             b = 2,
             c = 3 ] );
```

defines a bijection B with domain $\{a, b, c\}$ and range $\{1, 2, 3\}$, where $B[a]$ returns 1, $B[b]$ returns 2, and $B[c]$ returns 3.

Below we list the basic Maple functions for building bijections (all are found in [6, `BijTools`] except for `FibDef` which is in [6, `Fibonacci`]).

`IdBij(S)`	Given a set S, returns the identity bijection on S.
`FibDef(n)`	Given n an integer, returns $\mathfrak{d}_n : \mathcal{F}_n \to \mathcal{F}_{n-1} \uplus \mathcal{F}_{n-2}$. (See Subsection 2.1 for the definition of $\mathfrak{d}_n$.)

Figure 2: the Alternating Paths Algorithm

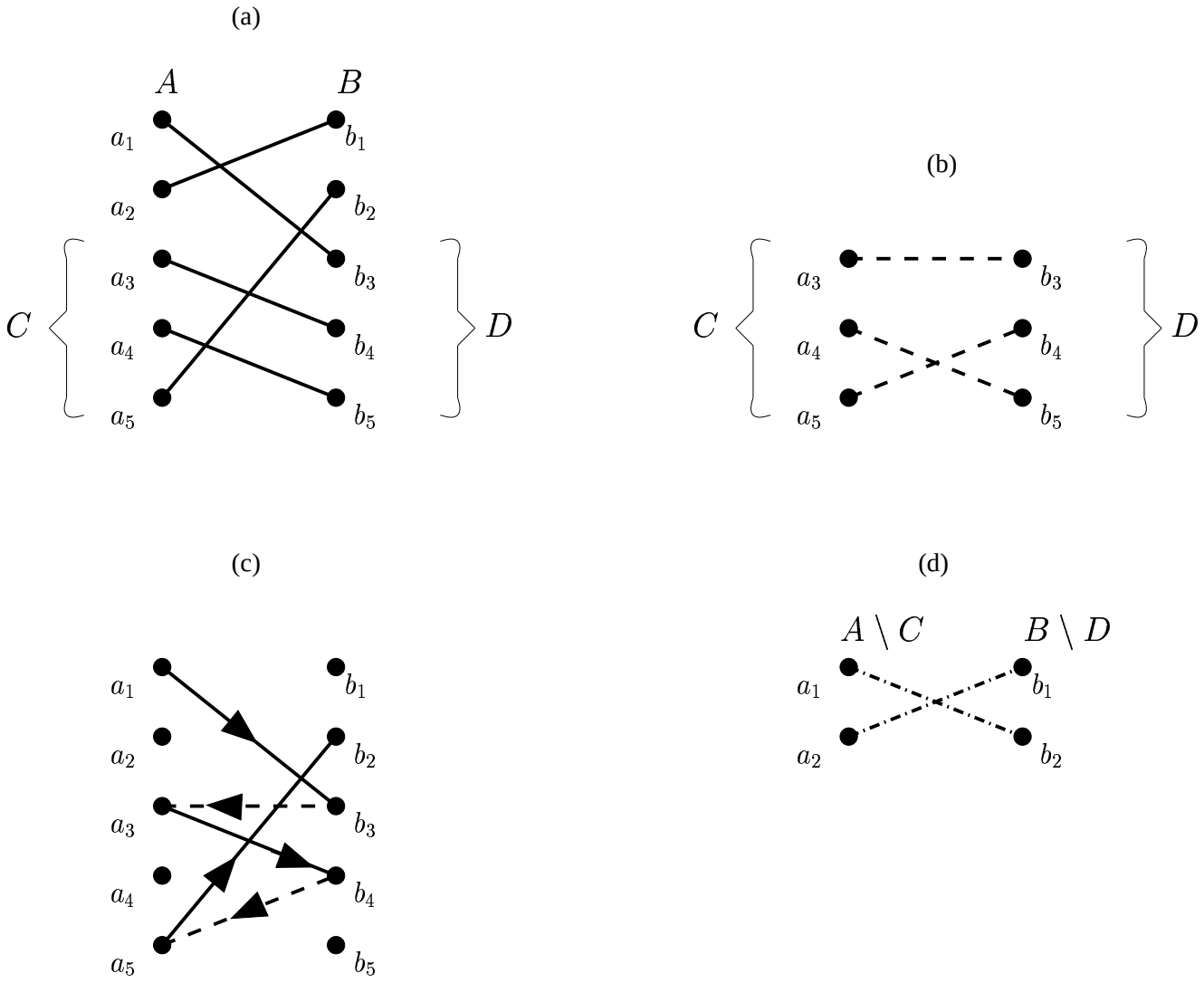

Figure 2: Given $C \subset A$ and $D \subset B$ and given bijections $\alpha : A \to B$ (shown with solid lines in (a) above) and $\gamma : C \to D$ (shown with dashed lines in (b) above), this figure describes how to derive a bijection $\delta : A \setminus C \to B \setminus D$ (shown with dot-dashed lines in (d) above). In (c), we show how to determine the image of a_1 using the alternating paths algorithm, and (c) also demonstrates where this algorithm its its name. The alternating paths algorithm is implemented as `SubtBij` in [6, `BijTools`] as a way of subtracting bijections. Note that the alternating paths algorithm is essential if the bijection α does not map c onto D (as above). See Subsection 3.6 for a practical example where using the alternating paths algorithm (as implemented in the function `SubtBij`) is essential.

`AddBij`(α, γ)	Given $\alpha : A \to B$ and $\gamma : C \to D$ bijections with $A \cap C = \emptyset$ and $B \cap D = \emptyset$, returns a bijection $\sigma : A \uplus B \to C \uplus D$ as described above.
`SumBij`$([\alpha_1, \ldots, \alpha_r])$	Given a list of bijections $[\alpha_1, \ldots, \alpha_r]$, iteratively applies `AddBij`. E.g., if $r = 3$, returns `AddBij(AddBij(`α_1, α_2`),`α_3`)`.
`MultBij`(α, γ)	Given $\alpha : A \to B$ and $\gamma : C \to D$ bijections, returns $\sigma : A \times C \to B \times D$ as described above.
`SubtBij`$(\alpha, \gamma\ )$	Given $\alpha : A \to B$ and $\gamma : C \to D$ bijections where $C \subset A$ and $D \subset B$, returns a bijection $\sigma : A \setminus C \to B \setminus D$ using the alternating paths algorithm described above (also see Figure 2).
`InvBij`(α)	Returns the inverse bijection α^{-1}.
`ComposBij`(β, α)	Given $\alpha : A \to B$ and $\beta : B \to C$, returns the bijection $\beta \circ \alpha : A \to C$.

See Figure 1 and Figure 3 for examples of how to use the above Maple functions.

3 Applications of the translation method to Fibonacci identities

3.1 Cassini's Fibonacci Identity

Cassini's Fibonacci Identity

$$f_n^2 = f_{n+1} f_{n-1} + (-1)^n, \tag{7}$$

which is also known as Simson's Formula, was first proven bijectively in 1986 [7]. Cassini's Identity is Identity 8 of [1, page 8], and a good exposition of the bijection may also be found there. The original proof in [7] was, in fact, discovered by the translation method described in Section 2. In this subsection, we will demonstrate how the translation method easily leads to a direct bijection for Equation (7).

Consider the following inductive proof of Equation (7):

$$\begin{aligned}
f_n^2 &= f_n(f_{n-1} + f_{n-2}) && \text{(defn of Fibonacci numbers)} \\
&= f_n f_{n-1} + f_n f_{n-2} \\
&= f_n f_{n-1} + f_{n-1}^2 - (-1)^{n-1} && \text{(inverse of Cassini for } n-1\text{)} \\
&= (f_n + f_{n-1}) f_{n-1} + (-1)^n \\
&= f_{n+1} f_{n-1} + (-1)^n && \text{(inverse defn of Fibonacci numbers).}
\end{aligned}$$

When finding a bijection for an identity, it is best to have all quantities in the equation be positive; thus, we will construct a recursive bijection in two cases, showing that

$$\begin{aligned}
\mathcal{F}_n^2 \uplus \{1\} &\xrightarrow{\sim} \mathcal{F}_{n+1} \times \mathcal{F}_{n-1} && \text{when } n \text{ is odd, and} \\
\mathcal{F}_n^2 &\xrightarrow{\sim} (\mathcal{F}_{n+1} \times \mathcal{F}_{n-1}) \uplus \{1\} && \text{when } n \text{ is even.}
\end{aligned}$$

We must also consider what to do with the base case (note that only one base case is necessary, since the odd-n bijection will recursively call the even-n bijection and vice versa). Thus, the natural choice for the base case—that is, the value of n where both sides of Equation (7) are as small as possible—is $n = 0$, giving us the formula $1^2 = f_0^2 = f_1 f_{-1} + 1 = 1 \cdot 0 + 1$.

To translate the base case into a bijection, we need to represent f_{-1} as the cardinality of a set, and since $f_{-1} = 0$, the natural choice is the empty set; hence $\mathcal{F}_{-1} := \emptyset$. (In Section 3.3, we will

discuss a case where it is useful to find sets to represent f_n for other negative values of n.) Note that for the base case of $n = 0$, there is only one choice for the bijection for Equation (7), and thus only one possible bijection may be derived from the given inductive proof via the translation method.

A recursive bijection for Equation (7) may now be readily constructed. The Maple code in Figure 3 for the bijection mirrors that of the inductive proof given at the start of this section (see [6] for complete code and related files).

Studying the output for some small cases leads us to the directly-defined "tail-swapping" bijection described in [1, page 8] and [7]. (see Figure 4). An implementation for the "tail-swapping" bijection is given in [6, `Examples`], and for a given n, it can be verified to be the same as the recursive-bijection using the `BijEqual` function. We should emphasize here that the "tail-swapping" bijection is very elegant—so much so that it is one of the first bijections described in [1]—which shows that while the translation method may seem complicated, it often produces bijections that are simple and natural.

3.2 A recently solved problem from *Proofs that really count* [1]

The Fibonacci Identity

$$f_{n+4} + f_1 + 2f_2 + \cdots + nf_n = (n+1)f_{n+2} + 3 \tag{8}$$

was stated in 2003 [1, page 14] as an identity with no known bijective proof, and in 2006 [4] a direct bijective proof was discovered using the translation method of Section 2. In [4], the base case of $n = 0$ was used; however, it is more natural to use the base case of $n = -1$ and interpret Equation (8) as $f_{n+4} + -1f_{-1} + 0f_0 + f_1 + 2f_2 + \cdots + nf_n = (n+1)f_{n+2} + 3$. With this interpretation the $n = -1$ case of the formula becomes $3 = f_3 = (-1+1)f_1 + 3 = 0 + 3$ and the corresponding bijection, say

$$\begin{array}{rcl} [\![1,1,1]\!] & \mapsto & 1 \\ [\![1,2]\!] & \mapsto & 2 \\ [\![2,1]\!] & \mapsto & 3, \end{array}$$

is unique up to renaming the symbols "1", "2", and "3".

A recursive bijection from an inductive proof for Equation (8) is implemented in [6, Examples], as is a directly defined analog (see functions `IdRec` and `IdDirect`, respectively). For an exposition of the direct bijection, we refer readers to [4], and in the remainder of this section we will discuss the slight changes one needs to make to the bijection in [4] so that it matches the bijections defined by `IdRec` and `IdDirect` (which are of course the same) [6, Examples]. The bijection defined in `IdDirect` will be called $\widetilde{\phi}$, and that defined in [4] will be called ϕ.

In [4], the bijection

$$\phi : \mathcal{F}_{n+4} \cup \bigcup_{k=1}^{n} ([k] \times \mathcal{F}_k) \longrightarrow \{1,2,3\} \cup ([n+1] \times \mathcal{F}_{n+2})$$

Figure 3

```
# The code for the base case of n = 0:

Id8Base0:=proc() local B:
B:=table([ [[],[]] = 1 ]);
B:
end:

# The code for the base inductive step if n is odd:

Id8:=proc(n) local B:
if n = 0 then
    return Id8Base0():
elif (n mod 2 = 1) then #  n  is odd
    # f_n^2 +1 = f_n(f_{n-1} + f_{n-2}) + 1
    B[1]:= AddBij( MultBij(IdBij(Fn(n)),FibDef(n)) , IdBij({1}) ):

    # # the below is an identity of sets, so no bijection needed.
    # f_n(f_{n-1} + f_{n-2}) + 1 = f_n f_{n-1} + f_n f_{n-2} + 1

    # f_n f_{n-1} + f_n f_{n-2} + 1 = f_n f_{n-1} + f_{n-1}^2
    B[2]:= AddBij( MultBij(IdBij(Fn(n)),IdBij(Fn(n-1))),
                   InvBij(Id8(n-1)) ):

    # # the below is an identity of sets, so no bijection needed.
    # f_n f_{n-1} + f_{n-1}^2 = (f_n + f_{n-1}) f_{n-1}

    # (f_n + f_{n-1}) f_{n-1} = f_{n+1} f_{n-1}
    B[3]:= MultBij( InvBij(FibDef(n+1)), IdBij(Fn(n-1)) ):

    # f_n^2 +1 = f_n(f_{n-1} + f_{n-2}) + 1
    #          = f_n f_{n-1} + f_{n-1}^2
    #          = f_{n+1} f_{n-1}
    B[4]:= ComposeBij(B[3],ComposeBij(B[2],B[1])):

    return(B[4]):

elif (n mod 2 = 0) then #  n  is even

# Rest of even-n case omitted.
```

Figure 3: Above is the Maple code used to generate a recursive bijection for Equation (7). Again, note how closely the code above mirrors the inductive proof given in Subsection 3.1. The bijection, called `Id8`, is stored as a table in Maple, thus a line of the form "`[[],[]] = 1`" means that in the bijection, $([\![], [\![]) \mapsto 1$. See [6, `Examples`] for the full code.

Figure 4

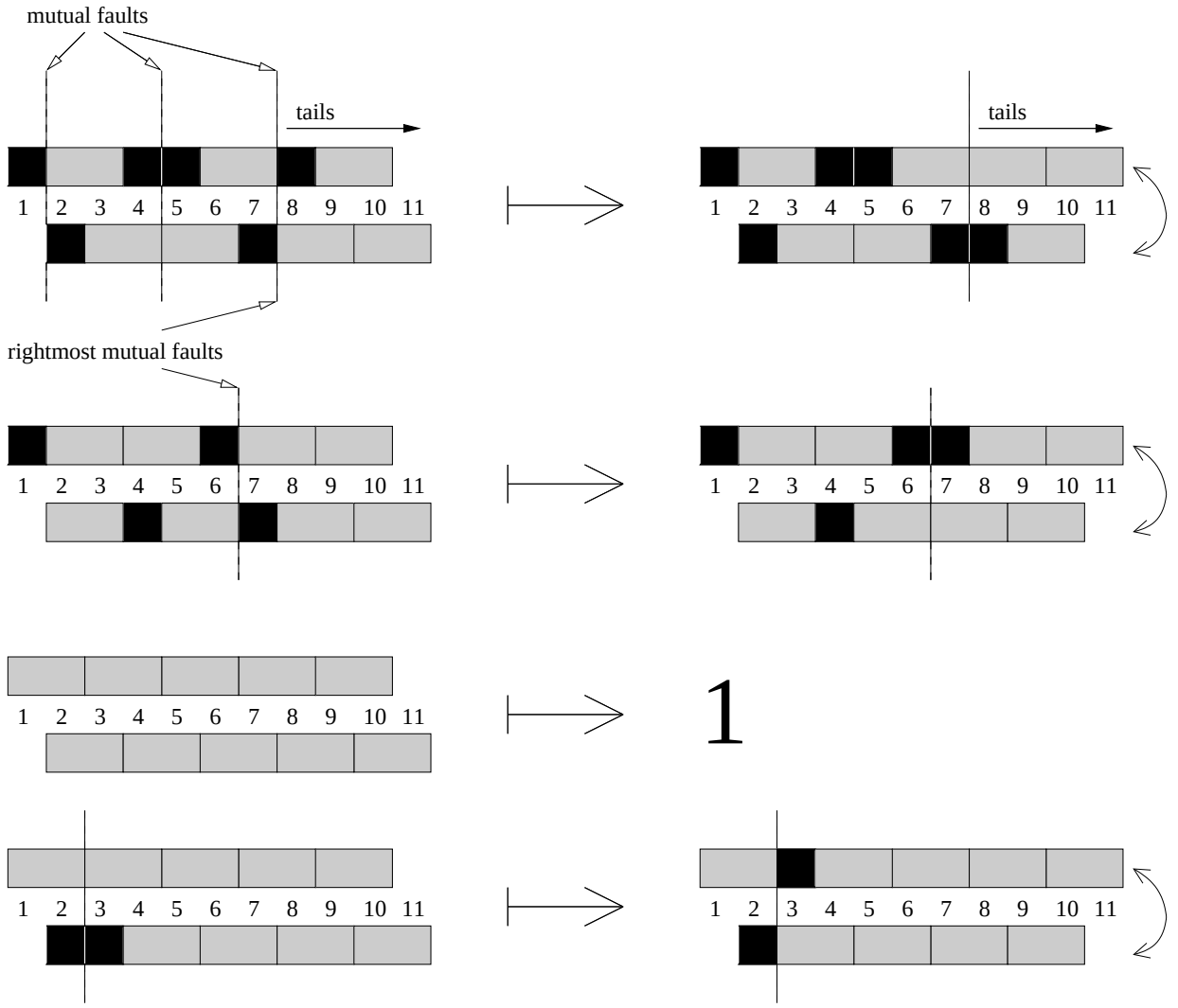

Figure 4: Above are a few examples describing the "tail-swapping" bijection for Equation (7). Each element of $\mathcal{F}_n$ is represented by a sequence of dominoes and monomios of total length n. For example, that the pair of tilings of length 10 in the upper left corner of this figure represent the element $([\![1,2,1,1,2,1,2]\!],[\![1,2,2,1,2,2]\!]) \in \mathcal{F}_{10} \times \mathcal{F}_{10}$. The tails of a pair tilings are defined to be the portion of tilings to the right of the rightmost mutual fault. The bijection works by simply taking the tail from one tiling and swapping it with the tail from the other tiling in the pair. If there is no mutual fault, then the pair of tilings corresponds to the symbol "1". For a complete description of the bijection, see [1, page 8].

was defined using five special cases, namely

$$\begin{aligned}
\phi: \quad & [\![1,1,1,1,\overbrace{1,1,\ldots,1}^{n}]\!] && \mapsto && (1,[\![\overbrace{1,1,\ldots,1}^{n+2}]\!]) \\
\phi: \quad & [\![2,1,1,\overbrace{1,1,\ldots,1}^{n}]\!] && \mapsto && 1 \\
\phi: \quad & [\![1,2,1,\overbrace{1,1,\ldots,1}^{n}]\!] && \mapsto && 2 \\
\phi: \quad & [\![1,1,2,\overbrace{1,1,\ldots,1}^{n}]\!] && \mapsto && 3 \\
\phi: \quad & [\![2,2,\overbrace{1,1,\ldots,1}^{n}]\!] && \mapsto && (1,[\![2,\overbrace{1,1,\ldots,1}^{n}]\!])
\end{aligned}$$

These five special cases were a byproduct of having used $n = 0$ as the base case, since for $n = 0$ Equation (8) becomes $5 = f_4 = 1 \cdot f_2 + 3 = 2 + 3$. In writing the current paper, it was noticed that using the base case $n = -1$ is more natural, and so for $\widetilde{\phi}$ the five special cases above become

$$\begin{aligned}
\widetilde{\phi}: \quad & [\![1,1,1,1,\overbrace{1,1,\ldots,1}^{n}]\!] && \mapsto && 1 \\
\widetilde{\phi}: \quad & [\![1,2,1,\overbrace{1,1,\ldots,1}^{n}]\!] && \mapsto && 2 \\
\widetilde{\phi}: \quad & [\![2,1,1,\overbrace{1,1,\ldots,1}^{n}]\!] && \mapsto && 3
\end{aligned}$$

$$\boxed{\begin{aligned}
\widetilde{\phi}: \quad & [\![1,1,2,\overbrace{1,1,\ldots,1}^{n}]\!] && \mapsto && (1,[\![\overbrace{1,1,\ldots,1}^{n+2}]\!]) \\
\widetilde{\phi}: \quad & [\![2,2,\overbrace{1,1,\ldots,1}^{n}]\!] && \mapsto && (1,[\![2,\overbrace{1,1,\ldots,1}^{n}]\!]).
\end{aligned}} \tag{9}$$

The only difference between the bijections ϕ and $\widetilde{\phi}$ is on the five cases displayed above. Thus, the remainder of the definition of $\widetilde{\phi}$ is given by Case 1 and Case 2 on page two of [4]. We argue that $\widetilde{\phi}$ is more natural, since the two boxed cases in Display (9) are no longer special—they are now covered by the general case of how $\widetilde{\phi}$ behaves on other elements of $\mathcal{F}_{n+4}$ (see Case 1 of [4, page 2]).

In this way, $\widetilde{\phi}$ also gives a combinatorial interpretation for the 3 in Equation (8), namely the three elements $\mathcal{F}_{n+4}$ ending in at least $(n+1)$ 1's. Thus, Equation (8) might be more natural if the 3 were replaced by f_3 and the elements $1, 2, 3$ in the range of $\widetilde{\phi}$ were replaced by $[\![1,1,1]\!], [\![1,2]\!], [\![2,1]\!]$, respectively.

3.3 Zeckendorf Family Identities

The ℓ-th Zeckendorf Family Identity is defined to be an equation of the form

$$\ell f_n = \sum_{t \in S_\ell} f_{n+t}$$

that holds for all integers n, where S_ℓ is a finite set of integers depending only on ℓ containing no two adjacent integers. Note that the ℓ-th Zeckendorf Family Identity exists and is unique for every $\ell \geq 1$ (see [5, Section 3] for more discussion of Zeckendorf Family Identities). The validity of a given Zeckendorf Family Identity may easily be proven by induction; for example Equation (1) is the 3rd

Zeckendorf Family Identity and is proven by induction in Equations (2), (3), and (4). In [1], the 5th through 12th Zeckendorf Family Identities were stated as identities in need of bijective proofs. Using the translation method in the same way as in Subsection 2.1, it is not hard to construct direct bijections for any specific Zeckendorf Family Identity, and such direct bijections for the 5th through 12th Zeckendorf Family Identities are described in [5].

The rest of this section is devoted to discussing an issue not mentioned in [5], namely how to choose the base case so that the recursive bijection derived from the translation method may be defined in a natural, direct way. For example, when defining the base case ϕ_2 in Subsection 2.1, there were apparently 6! suitable choices. We will demonstrate below that, in fact, there is only one natural candidate. The general approach is to take advantage of the fact that the Fibonacci numbers can be defined for negative indices. To define the Fibonacci numbers with negative indices, we use the relation $f_{n-2} = f_n - f_{n-1}$ and the base case $f_0 = f_1 = 1$, and thus we have the following table as n decreases:

n	1	0	-1	-2	-3	-4	-5	-6	-7	$\cdots$
f_n	1	1	0	1	-1	2	-3	5	-8	$\cdots$

With this definition, the ℓ-th Zeckendorf Family Identity holds for all integers n. We then make the heuristic assumption that any "natural" recursive bijection for the ℓ-th Zeckendorf Family Identity must be built up from the base cases where both sides of the identity are as small as possible. In particular, if we consider the case where $n = -1$, the ℓ-th Zeckendorf Family Identity becomes:

$$0 = \ell f_{-1} = \sum_{t \in S_\ell} f_{-1+t} \tag{10}$$

where the Fibonacci numbers on the right-hand side sum to zero since some of them are negative. In Subsection 3.4, we will discuss a way to interpret Fibonacci numbers with negative indices as sets, thus making Equation (10) a combinatorial identity for any ℓ. It is then straightforward to use the translation method to construct a recursive bijection for all n, building up from an arbitrarily chosen base case bijection for $n = -1$. Of course, we actually need to define two base cases (as in Subsection 2.1); however, the different base cases possible for $n = 0$ amount to simply re-naming the symbols $1, 2, \ldots, \ell$ (note that the domain for the $n = 0$ case is $[\ell] \times \mathcal{F}_0 = \{(1, [\![]\!]), (2, [\![]\!]), \ldots, (\ell, [\![]\!])\}$.). Thus, we consider recursive bijections differing only in the choice of a base case for $n = 0$ to be formally equivalent (see [5, Section 3] for more discussion of formal equivalence).

3.4 Interpreting Fibonacci numbers with negative indices as sets

For $n > 0$, we will interpret the Fibonacci number f_{-n} as the set $\mathcal{F}_{-n}$ of all lists $[\![-2, a_1, \ldots, a_k]\!]$ such that $-2 + \sum_{j=1}^{k} a_j = -n$ and each $a_j \in \{-1, -2\}$. We will call f_{-n} for $n \geq 3$ and n odd a *negative Fibonacci number* since $f_{-n} < 0$. If $n \geq 2$ is even, then f_{-n} is a positive integer and is equal to the cardinality of $\mathcal{F}_{-n}$. If $n \geq 3$ is odd, then $-1 \cdot f_{-n}$ is a positive number and is equal to the cardinality of $\mathcal{F}_{-n}$. For example:

$$\begin{aligned}
f_{-1} &= 0 \quad \text{and} \quad \mathcal{F}_{-1} = \{\} \\
f_{-2} &= 1 \quad \text{and} \quad \mathcal{F}_{-2} = \{[\![-2]\!]\} \\
f_{-3} &= -1 \quad \text{and} \quad \mathcal{F}_{-3} = \{[\![-2, -1]\!]\} \\
f_{-4} &= 2 \quad \text{and} \quad \mathcal{F}_{-4} = \{[\![-2, -1, -1]\!], [\![-2, -2]\!]\} \\
f_{-5} &= -3 \quad \text{and} \quad \mathcal{F}_{-5} = \{[\![-2, -1, -1, -1]\!], [\![-2, -1, -2]\!], [\![-2, -2, -1]\!]\} \\
&\vdots \qquad\qquad\quad \vdots
\end{aligned}$$

Because the cardinality of $\mathcal{F}_{-n}$ is always equal to $(-1)^n f_{-n}$ for $n > 0$, we can always rearrange terms in a Zeckendorf Family Identity so that any negative Fibonacci number is multiplied by -1. For example, the 3rd Zeckendorf Family Identity at $n = -1$ becomes

$$-f_{-1-2} + 3f_{-1} = f_{-1+2} \tag{11}$$

(instead of $3f_{-1} = f_{-1+2} + f_{-1-2}$ since f_{-3} is negative); and the 5th Zeckendorf Family Identity (namely $5f_n = f_{n+3} + f_{n-1} + f_{n-4}$) at $n = -1$ becomes

$$-f_{-1-4} + 5f_{-1} = f_{-1+3} + f_{-1-1}. \tag{12}$$

Equation (11) may be interpreted (uniquely) as the bijection $[\![-2, -1]\!] \mapsto [\![1]\!]$, since the sets corresponding to each side of Equation (11) each have cardinality 1. Equation (12), on the other hand, has 3! equally valid interpretations as a bijection, one of which is

$$\begin{aligned} [\![-2, -1, -1, -1]\!] &\mapsto [\![1, 1]\!] \\ [\![-2, -1, -2]\!] &\mapsto [\![2]\!] \\ [\![-2, -2, -1]\!] &\mapsto [\![-2]\!]. \end{aligned}$$

Note that $3f_{-1}$ and $5f_{-1}$ are both interpreted as the empty set.

In Subsection 3.5 below, we will discuss the defining bijection for the Fibonacci numbers when negative indices are involved; and in Subsection 3.6, we will discuss how to apply the translation method when the base cases involve negative Fibonacci numbers (for example, if Equation (11) or (12) were a base case).

3.5 The defining bijection $\mathfrak{d}_{-n}$ for Fibonacci numbers when $n \geq 0$

In this section we will describe the defining bijection $\mathfrak{d}_{-n}$ for the Fibonacci numbers when $n > 0$. Our definitions below are motived by two things: first by interpreting the formula $f_{-n} = f_{-n-1} + f_{-n-2}$ for $n \geq 0$ (see Subsection 3.4), and second by the definition of $\mathfrak{d}_n$ when $n \geq 1$.

For $n = 0$ we have $f_0 = f_{-1} + f_{-2}$ which we interpret as the bijection

$$\begin{aligned} \mathfrak{d}_0 : \{[\![]\!]\} &\to \{\} \uplus \{[\![-2]\!]\} \\ [\![]\!] &\mapsto [\![-2]\!]. \end{aligned}$$

For $n = 1$ we have $f_{-1} = f_{-2} + f_{-3}$ which we re-write as $-f_{-3} + f_{-1} = f_{-2}$ (since $-f_{-3}, f_{-1}, f_{-2} \geq 0$) and interpret as the bijection

$$\begin{aligned} \mathfrak{d}_{-1} : \{[\![-2, -1]\!]\} \uplus \{\} &\to \{[\![-2]\!]\} \\ [\![-2, -1]\!] &\mapsto [\![-2]\!]. \end{aligned}$$

In general, for $n \geq 2$ and n even, we have $f_{-n} = f_{-n-1} + f_{-n-2}$ which we re-write as $-f_{-n-1} + f_{-n} = f_{-n-2}$ (since $-f_{-2-1}, f_{-n}, f_{-n-2} \geq 0$) and interpret as the bijection

$$\begin{aligned} &\mathfrak{d}_{-n} : \mathcal{F}_{-n-1} \uplus \mathcal{F}_{-n} \to \mathcal{F}_{-n-2} \\ &\quad [\![-2, a_1, \ldots, a_k]\!] \mapsto [\![-2, a_1, \ldots, a_k, -1]\!] \qquad \text{if } [\![-2, a_1, \ldots, a_k]\!] \in \mathcal{F}_{-n-1} \\ &\quad [\![-2, a_1, \ldots, a_k]\!] \mapsto [\![-2, a_1, \ldots, a_k, -2]\!] \qquad \text{if } [\![-2, a_1, \ldots, a_k]\!] \in \mathcal{F}_{-n}. \end{aligned}$$

Finally, for general $n \geq 3$ with n odd, we have $f_{-n} = f_{-n-1} + f_{-n-2}$ which we re-write as $-f_{-n-2} = f_{-n-1} - f_{-n}$ (since $-f_{-n-2}, f_{-n-1}, -f_{-n} \geq 0$) and interpret as the bijection

$$\begin{aligned} \mathfrak{d}_{-n} : \mathcal{F}_{-n-2} &\to \mathcal{F}_{-n-1} \uplus \mathcal{F}_{-n} \\ [\![-2, a_1, \ldots, a_k]\!] &\mapsto [\![-2, a_1, \ldots, a_{k-1}]\!]. \end{aligned}$$

The definition given above for $\mathfrak{d}_{-n}$ (for $n \geq 0$) is natural given our interpretation as sets of Fibonacci numbers with negative indices. Also, as we will see in the Subsection 3.6 below, this definition of $\mathfrak{d}_{-n}$ works well for applying the translation method to Zeckendorf Family Identities.

3.6 The translation method with negative Fibonacci numbers: An example

In this subsection we will revisit the motivating example of Subsection 2.1, and we will demonstrate how to use the translation method to construct a bijection for $3f_n = f_{n+2} + f_{n-2}$ (the 3rd Zeckendorf Family Identity, and also Equation (1)) using the base cases of $n = -1$ and $n = 0$. By extending the 3rd Zeckendorf Family Identity to Fibonacci numbers with negative indices, we will be able to show that the direct bijection derived in Subsection 2.1 is (in some sense) the only natural bijection for the 3rd Zeckendorf Family Identity derivable via the translation method. Also, the alternating paths algorithm for "subtracting bijections" is an essential tool for constructing recursive bijections for Zeckendorf Family Identities, as we will see below.

We will define the base cases for $n = -1$ and $n = 0$ first and then discuss how to build the bijection recursively. For $n = -1$, we re-write $3f_n = f_{n+2} + f_{n-2}$ as $-f_{-3} + 3f_{-1} = f_1$ and define

$$\begin{aligned} \Phi_{-1} : \quad & \mathcal{F}_{-3} \to \mathcal{F}_1 \\ & [\![-2, -1]\!] \mapsto [\![1]\!]. \end{aligned}$$

For $n = 0$, we write $3f_n = f_{n+2} + f_{n-2}$ as $3f_0 = f_2 + f_{-2}$ and define

$$\begin{aligned} \Phi_0 : \quad & \{1, 2, 3\} \times \mathcal{F}_0 \to \mathcal{F}_2 \uplus \mathcal{F}_{-2} \\ & (1, [\![]\!]) \mapsto [\![1, 1]\!] \\ & (2, [\![]\!]) \mapsto [\![2]\!] \\ & (3, [\![]\!]) \mapsto [\![-2]\!]. \end{aligned}$$

Note that there is only one possible bijection for the case $n = -1$, and for the case $n = 0$, choosing another of the 3! possible bijections amounts to re-naming the symbols 1, 2, and 3. Thus, there is only one natural way to choose the base cases for the 3rd Zeckendorf Family Identity, and since the base cases determine the rest of the bijection by the translation method, we see that there is only one natural bijection derivable via the translation method.

For the induction step, we will have to consider the case of $n = 1$ separately because we must always re-arrange the terms so that all quantities are non-negative. In particular, if we attempt to proceed as in Equations (2), (3), and (4), we would have

$$\begin{aligned} 3f_1 &= 3f_0 + 3f_{-1} && \text{(definition of Fibonacci numbers)} \\ &= f_2 + f_{-2} + f_1 + f_{-3} && \text{(recursion)}, \end{aligned}$$

which is not useful since f_{-3} is negative. Instead we will use the following chain of equalities in

which all terms are positive.

$$-f_{-3} + 3f_1 = 3f_0 + 3f_{-1} - f_{-3} \qquad \text{(definition of Fibonacci numbers)} \tag{13}$$

$$= f_2 + f_{-2} + f_1 \qquad \text{(recursion, using } 3f_0 = f_2 + f_{-2} \text{ and } -f_{-3} + 3f_0 = f_1) \tag{14}$$

$$= f_3 + f_{-1} - f_{-3} \qquad \text{(definition of Fibonacci numbers, using } f_2 + f_1 = f_3 \text{ and } f_{-2} = f_{-1} - f_{-3}) \tag{15}$$

If we now subtract the positive quantity $-f_{-3}$ from both sides of $-f_{-3} + 3f_1 = f_3 + f_{-1} - f_{-3}$, we get the desired identity:

$$3f_1 = f_3 + f_{-1}.$$

If the bijection corresponding to $-f_{-3} + 3f_1 = f_3 + f_{-1} - f_{-3}$ mapped the set $\mathcal{F}_{-3}$ to itself, then subtracting $-f_{-3}$ would be trivial—we could simply restrict the domain of the bijection to $\{1, 2, 3\} \times \mathcal{F}_1$. However, this is *not* the case, and so we must use the alternating paths algorithm to subtract the bijections. Since all of the terms in this derivation were non-negative, we can easily interpret each equality as a bijection of sets. The maple code for the cases $n = -1, 0$, and 1 is given in Figure 5.

Finally, for $n \geq 2$, we may recursively construct a bijection in exactly the same way as in Subsection 2.1, since $f_{n-4} \geq 0$ for all $n \geq 2$ and $n-4$ is the smallest index to appear when constructing the bijection recursively (see Equations (2)–(4) or Figure 1 below the line "`## Induction Step ##`"). The resulting recursive bijection is the same as the one defined in Subsection 2.1, and so it can be defined in the same direct, non-recursive way.

3.7 The ℓ-th Zeckendorf Family Identity in general

The approach described in Subsection 3.6 above can be carried out in general for the ℓ-th Zeckendorf Family Identity. Recursive bijections for the ℓ-th Zeckendorf Family Identity for $\ell = 5, 6, \ldots, 12$ are implemented in Maple in [6, TransMethodZeck], and their directly defined counterparts are described in [5] with Maple implementation give in [6, ZeckFibBijections]. As we saw in Subsection 3.6, a recursive bijection for the ℓ-th Zeckendorf Family Identity is determined (up to formal equivalence—see [5]) by the definition of the bijection in the $n = -1$ base case. For example, the 5th Zeckendorf Family Identity is

$$5f_n = f_{n+3} + f_{n-1} + f_{n-4}$$

which has the base case of $-f_{-1-4} = f_{-1+3} + f_{-1-1} = 3$, and hence there are 3! distinct choices for the base case bijection mapping $\mathcal{F}_{-5} \to \mathcal{F}_2 \uplus \mathcal{F}_{-2}$. Each of the 3! choices leads to a distinct (in the sense of formal equivalence) bijection that may be directly defined.

The bijections given in [5] and implemented in [6, TransMethodZeck,ZeckFibBijections] represent an arbitrary selection among the many possibilities. Finding a canonical choice for a bijection for the ℓ-th Zeckendorf Family Identity remains open (see [5] for other possible further directions). We close this section with a chart showing how many distinct bijections (up to formal equivalence) may be generated for the ℓ-th Zeckendorf Family Identity by the translation method for $\ell = 6, 7, \ldots, 12$.

Figure 5

```
# The code for the base cases (n = -1 and n = 0, respectively):

Id7Base_n1:=proc() local B:
B:=table([ [-2,-1] = [1] ]);
B:
end:

Id7Base0:=proc() local B:
B:= table([ [1,[]] = [1,1],
            [2,[]] = [2],
            [3,[]] = [-2]
]);
B:
end:

# The code for the n = 1 case and the recursive cases (n ≥ 2):

Id7:=proc(n) local B;
if n=-1 then
    return Id7Base_n1():
elif n=0 then
    return Id7Base0():
elif n=1 then
    #  3 f_1 = 3 f_{0} + 3 f_{-1}
    B[1]:=MultBij(IdBij({1,2,3}), FibDef(n));

    #  3 f_1 - f_{-3}= 3 f_{0} + 3 f_{-1} - f_{-3}  ## note f_{-3} < 0.
    B[2]:=AddBij(B[1],IdBij(Fn(-3))):

    #  3 f_{0} + 3 f_{-1} - f_{-3} = f_{2} + f_{-2} + f_1
    B[3]:=AddBij(Id7(n-1), Id7(n-2)):

    #  f_2 + f_1 + f_{-2} = f_3 + f_{-1} - f_{-3}
    B[4]:=InvBij(AddBij(FibDef(n-2),FibDef(n+2))):

    #  3 f_1 - f_{-3} = 3 f_{0} + 3 f_{-1} - f_{-3}   \\
    #                 = f_{2} + f_{-2} + f_1          \\
    #                 = f_3 + f_{-1} - f_{-3}
    B[5]:=ComposeBij(B[4],ComposeBij(B[3],B[2])):

    # $ 3 f_1 =  f_3 + f_{-1}$
    B[6]:=SubtBij(B[5],IdBij(Fn(-3))):
    return B[6]:
elif n>1 then
    ## Induction Step ##
            ⋮
```

Figure 5: Above is the Maple code used to generate a recursive bijection for Equation (1), starting with the base cases $n = -1$ and $n = 0$. Notice how closely the code above mirrors the inductive proof described Equations (13), (14), and (15) and the sentence following Equation (15). This figure should be compared with Figure 1. The primary difference between the code here and that in Figure 1 is that here we added the positive quantity $-f_{-3}$ to both sides, and then later subtracted using `SubtBij` in the line starting with `B[6]`. Using the function `SubtBij` here is essential, since the bijection `B[5]` does **not** map $\mathcal{F}_{-3}$ to itself. Also, note that the omitted code following "`## Induction Step ##`" above is *identical* to the code following "`## Induction Step ##`" shown in Figure 1. See [6, `Examples`] for the full code.

ℓ	ℓ-th Zeckendorf Family Identity	Base Case ($n = -1$)	Number of Bijections
6	$6f_n = f_{n+3} + f_{n+1} + f_{n-4}$	$-f_{-1-4} = f_{-1+3} + f_{-1+1}$ $= 3$	3!
7	$7f_n = f_{n+4} + f_{n-4}$	$-f_{-1-4} = f_{-1+4}$ $= 3$	3!
8	$8f_n = f_{n+4} + f_n + f_{n-4}$	$-f_{-1-4} = f_{-1+4} + f_{-1}$ $= 3$	3!
9	$9f_n = f_{n+4} + f_{n+1} + f_{n-2} + f_{n-4}$	$-f_{-1-4} - f_{-1-2} = f_{-1+4} + f_{-1+1}$ $= 4$	4!
10	$10f_n = f_{n+4} + f_{n+2} + f_{n-2} + f_{n-4}$	$-f_{-1-4} - f_{-1-2} = f_{-1+4} + f_{-1+2}$ $= 4$	4!
11	$11f_n = f_{n+4} + f_{n+2} + f_n + f_{n-2} + f_{n-4}$	$-f_{-1-4} - f_{-1-2} = f_{-1+4} + f_{-1+2} + f_n$ $= 4$	4!
12	$12f_n = f_{n+5} + f_{n-1} + f_{n-3} + f_{n-6}$	$-f_{-1-6} = f_{-1+5} + f_{-1-1} + f_{-1-3}$ $= 8$	8!

References

[1] Benjamin, Arthur T.; Quinn, Jennifer J. *Proofs that really count. The art of combinatorial proof.* The Dolciani Mathematical Expositions, 27. Mathematical Association of America, Washington, DC, 2003.

[2] Garsia, A. M.; Milne, S. C. A Rogers-Ramanujan bijection. *J. Combin. Theory Ser. A.* **31** (1981), no. 3, 289–339.

[3] Garsia, A. M.; Milne, S. C. Method for constructing bijections for classical partition identities. *Proc. Nat. Acad. Sci. U.S.A.* **78** (1981), no. 4, part 1, 2026–2028.

[4] Wood, Philip Matchett. A bijective proof of $f_{n+4} + f_1 + 2f_2 + \cdots + nf_n = (n+1)f_{n+2} + 3$. *Integers* **6** (2006), A2, 4 pp.

[5] Wood, Philip Matchett. Bijective proofs for Fibonacci identities related to Zeckendorf's Theorem. (submitted).

[6] Wood, Philip Matchett. Five Maple packages: `BijTools`, `Examples`, `Fibonacci`, `TransMethodZeck`, `ZeckFibBijections`, online at `http://www.math.rutgers.edu/~matchett/Publications/Maple.html`

[7] Werman, M.; Zeilberger, D. A bijective proof of Cassini's Fibonacci identity. *Discrete Math.* **58** (1986), no. 1, 109.

[8] Zeilberger, D. Enumerative and Algebraic Combinatorics, to appear in *Princeton Companion of Mathematics* (T. Gowers, ed.), Princeton University Press, Princeton, NJ.