Dynamic One and Two Pile Nim Games Using Generalized Bases

Arthur Holshouser

3600 Bullard St.

Charlotte, NC, USA

Harold Reiter

Department of Mathematics,

University of North Carolina Charlotte,

Charlotte, NC 28223, USA

hbreiter@email.uncc.edu

Abstract

Several authors have written papers dealing with a class of combinatorial games consisting of one-pile and two-pile counter pickup games for which the maximum number of counters that can be removed on each move changes during the play of the game. See Holshouser [8],[7], and Flanigan [4]. The maximum number of counters that can be removed can depend upon a number of factors including the number of counters removed on the last move, the sizes of the piles, and the move number in the game. In this paper we consider a two-pile game in which the maximum number of counters that can be removed depends on both the size of the last move and difference between the two pile sizes of the previous position. Consider the following game where $f : N_0 \times N \to N_0$ is a function. Two players alternate removing positive numbers of counters from two piles of counters. On each move the moving player chooses a pile and removes counters from the chosen pile. Initially, the player moving first can remove from one pile at most k counters. On each subsequent move, a player can remove from one pile a maximum of $f(x - y, t)$ counters, where x and y, $x \geq y$, are the pile sizes in

the preceding position and t is the number of counters removed by his opponent on the preceding move. The game is over when the moving player cannot move. The winner is the last player to make a legal move in the game. When $f(n,t) = 2t$, the strategy uses the Fibonacci base $(1, 2, 3, 5, 8, 13, \cdots)$, and when $f(n,t) = t$, the strategy uses the binary base $(1, 2, 4, 8, 16, \cdots)$. The most efficient way to study these $f(n,t) = 2t$, $f(n,t) = t$ games is to first develop a basic theory. In this paper, we extend this basic theory as far as we can, and we also find all functions $f(n,t)$ for which this theory is effective. We also explain why the applications are endless. These applications can include increasingly complex single and two pile games in which counters can be removed or added back using arbitrarily complex rules which we discuss in the appendix. We have studied more than a dozen such games.

Introduction. In sections 1-3, we develop all the basic machinery and in section 4, we apply this machinery to study two relatively simple games. In the appendix, we explain how the rules can be made arbitrarily complex. It is helpful to point out that once theorem 1 is proved at the end of section 3, the reader can then forget much of the previous details.

Section 1. Throughout the paper we use the following notation. Let N denote the set of positive integers, and $N_0 = \{0\} \cup N$.

Definition 1. *A base* $B = (b_1 = 1, b_2, b_3, \cdots)$ *is an infinite strictly increasing sequence of positive integers satisfying* $b_1 = 1$ *and* $\forall k \in N, b_{k+1} - b_k \in \{b_1, b_2, \cdots, b_k\}$.

Definition 2. $\forall k \in N, \overline{g}(b_k) = b_{k+1} - b_k$. *That is,* $b_{k+1} = b_k + \overline{g}(b_k)$. *Thus,* $\overline{g}(1) = 1$.

For example, $B = (1, 2, 3, 6, 7, 10, 17, 34, \cdots)$. For this B, $\overline{g}(17) = 34 - 17 = 17, \overline{g}(7) = 10 - 7 = 3$.

Definition 3. $\forall k \in N$, *we say that* b_k *is binary if* $\overline{g}(b_k) = b_k$. *Also,* b_k *is non-binary if* $\overline{g}(b_k) < b_k$. *Thus* $b_1 = 1$ *is always binary.*

In the above example, 1, 3, 17 are binary, and 2, 6, 7,10 are non-binary.

Algorithm 1. *Let* $B = (b_1 = 1, b_2, b_3, \cdots)$ *satisfy definition 1.* $\forall n \in N$, *we can write* $n = b_{i_1} + b_{i_2} + \cdots + b_{i_t}$, *where* $b_{i_1} < b_{i_2} < \cdots < b_{i_t}$ *and each* $b_{i_j} \in B$, *by the following inductive algorithm. First,* $1 = b_1 \in B$. *Therefore, suppose* $1, 2, 3, \cdots, n-1$ *have been represented by the algorithm. Then* n *can be represented as follows. Let* $b_k \leq n < b_{k+1}$. *Then* $n = (n - b_k) + b_k$ *where* $0 \leq n - b_k < b_{k+1} - b_k = \overline{g}(b_k) \leq b_k$. *If* $0 = n - b_k$, *the algorithm*

is completed. If $1 \leq n - b_k$, *by induction we can write* $n - b_k = b_{i_1} + b_{i_2} + \cdots + b_{i_{t-1}}$, *where* $b_{i_1} < b_{i_2} < \cdots < b_{i_{t-1}} < b_k$ *and each* $b_{i_j} \in B$. *Letting* $b_{i_t} = b_k$, *we have* $n = b_{i_1} + b_{i_2} + \cdots + b_{i_t}$ *where* $b_{i_1} < b_{i_2} < \cdots < b_{i_t}$ *and each* $b_{i_j} \in B$.

As an example, for the $B = (1, 2, 3, 6, 7, 10, 17, 34, \cdots), 31 = 1 + 3 + 10 + 17$.

Definition 4. *Let* $n = b_{i_1} + b_{i_2} + \cdots + b_{i_t}, b_{i_1} \leq b_{i_2} \leq \cdots \leq b_{i_t}$ *and each* $b_{i_j} \in B$, *where* B *satisfies definition 1. We say that this representation of* n *in* B *is stable if* $\forall 2 \leq j \leq t, b_{i_{j-1}} < \overline{g}\left(b_{i_j}\right)$. *This implies* $b_{i_1} < b_{i_2} < \cdots < b_{i_t}$.

Observation 1. In definition 4 the statement $\forall\ 2 \leq j \leq t, b_{i_{j-1}} < \overline{g}\left(b_{i_j}\right)$ is equivalent to $\forall\ 2 \leq j \leq t, b_{i_{j-1}+1} \leq \overline{g}\left(b_{i_j}\right)$ since $\overline{g}\left(b_{i_j}\right) \in B$ and $b_{i_{j-1}+1}$ is the member of B coming right after $b_{i_{j-1}}$.

Lemma 1. *Let* $n = b_{i_1} + b_{i_2} + \cdots + b_{i_t}, b_{i_1} \leq b_{i_2} \leq \cdots \leq b_{i_t}$ *and each* $b_{i_j} \in B$, *where* B *satisfies definition 1. Then this representation of* n *in* B *is stable if and only if this representation of* n *in* B *is computed by algorithm 1.*

Proof. It is easy to see that algorithm 1 gives a stable representation of n in B. Suppose $n = b_{i_1} + b_{i_2} + \cdots + b_{i_t}, b_{i_1} \leq b_{i_2} \leq \cdots \leq b_{i_t}$ and each $b_{i_j} \in B$, is stable. We show that this representation of n in B is computed by algorithm 1. Suppose $t \geq 2$. Now $\forall\ 2 \leq j \leq t, b_{i_{j-1}} < \overline{g}\left(b_{i_j}\right)$ is equivalent to $b_{i_{j-1}+1} \leq \overline{g}\left(b_{i_j}\right)$. Now $b_{i_1} < \overline{g}\left(b_{i_2}\right)$ implies $b_{i_1} + b_{i_2} < b_{i_2+1} \leq \overline{g}\left(b_{i_3}\right)$. Also, $\left(b_{i_1} + b_{i_2}\right) + b_{i_3} < b_{i_3+1} \leq \overline{g}\left(b_{i_4}\right)$. Continuing in this pattern, we end up with $b_{i_t} < b_{i_1} + b_{i_2} + \cdots + b_{i_t} < b_{i_t+1}$. This means that b_{i_t} must have the same value that is computed by algorithm 1. Since $b_{i_1} + b_{i_2} + \cdots + b_{i_{t-1}}$ is also in stable form, this also means that $b_{i_{t-1}}$ must have the same value that is computed by algorithm 1. Continuing in this pattern, we see that $b_{i_1}, b_{i_2}, \cdots, b_{i_t}$ must all have the values computed by algorithm 1. □

Definition 5. $\forall n \in N$, *let* $n = b_{i_1} + b_{i_2} + \cdots + b_{i_t}, b_{i_1} < b_{i_2} < \cdots < b_{i_t}$ *and each* $b_{i_j} \in B$, *be computed by algorithm 1, where* B *satisfies definition 1. We say that* n *is* semi-stable *if* $t \geq 2$ *and* $b_{i_1+1} = \overline{g}\left(b_{i_2}\right)$.

By observation 1, $b_{i_1+1} \leq \overline{g}\left(b_{i_2}\right)$ is always true. Note that the representation of n in B is stable but the term semi-stable refers to n itself.

Definition 6. B *satisfies definition 1 and* $\forall n \in N$, *let* $n = b_{i_1} + b_{i_2} + \cdots + b_{i_t}, b_{i_1} < b_{i_2} < \cdots < b_{i_t}$ *and each* $b_{i_j} \in B$, *be computed by algorithm 1. Then we define* $g(n) = b_{i_1}$ *and*

$\overline{g}(n) = \overline{g}(g(n)) = \overline{g}(b_{i_1})$. *Also, we define* $\overline{g}(0) = g(0) = \infty$. *Of course, when* $g(n) = b_{i_1} = 1, \overline{g}(n) = 1$.

Note that n is semi-stable if and only if $\overline{g}(n) + g(n) = \overline{g}(n - g(n))$.

Definition 7. *$\forall n \in N$, let $n = b_{i_1} + b_{i_2} + \cdots + b_{i_t}, b_{i_1} < b_{i_2} < \cdots < b_{i_t}$ and each $b_{i_j} \in B$, be computed by algorithm 1, where B satisfies definition 1. We say that n is* binary *if $g(n) = b_{i_1}$, is binary. That is, $\overline{g}(b_{i_1}) = b_{i_1}$. We say that n is* non-binary *if $g(n) = b_{i_1}$ is non-binary. That is, $\overline{g}(b_{i_1}) < b_{i_1}$. Thus n is binary if $\overline{g}(n) = g(n)$, and n is non-binary if $\overline{g}(n) < g(n)$.*

Lemma 2. *Suppose $n = b_{i_1} + b_{i_2} + \cdots + b_{i_t}, b_{i_1} < b_{i_2} < \cdots < b_{i_t}$ and each $b_{i_j} \in B$, is computed by algorithm 1, where B satisfies definition 1. Suppose n is semi-stable. That is, $t \geq 2$ and $b_{i_1+1} = \overline{g}(b_{i_2})$. Then $\exists r, 2 \leq r \leq t$, such that $\overline{g}(n) + n = b_{i_r+1} + b_{i_{r+1}} + \cdots b_{i_t}, b_{i_r+1} < b_{i_{r+1}} < \cdots < b_{i_t}$, is in stable form. From lemma 1, this stable representation of $\overline{g}(n) + n$ in B must be the same form as using algorithm 1 to represent $\overline{g}(n) + n$ in B.*

Proof. Since n is semi-stable, $\overline{g}(n) + n = \overline{g}(b_{i_1}) + b_{i_1} + \cdots + b_{i_t} = b_{i_1+1} + b_{i_2} + \cdots + b_{i_t} = b_{i_2+1} + b_{i_3} + \cdots + b_{i_t}$. Suppose this is not in stable form. Then since $b_{i_2+1} \leq \overline{g}(b_{i_3})$, this means $b_{i_2+1} = \overline{g}(b_{i_3})$ and we must replace $b_{i_2+1} + b_{i_3}$ by b_{i_3+1}. If $b_{i_3+1} + b_{i_4} + \cdots + b_{i_t}$ is not in stable form, we must have $b_{i_3+1} = \overline{g}(b_{i_4})$, and we must replace $b_{i_3+1} + b_{i_4}$ by b_{i_4+1}. We keep doing this over and over until $b_{i_2+1} + b_{i_3} + \cdots + b_{i_t}$ has been written in stable form, and this proves lemma 2. □

Section 2. When we first researched this paper, we started with the two functions $f(n,t) = 2t$ and $f(n,t) = t$. The strategies for $f(n,t) = 2t$ and $f(n,t) = t$ led to the Fibonacci and binary bases respectively. We then unified our proofs and developed a basic theory that used a consolidated Fibonacci-Binary base where $b_1 = 1$ and $b_{k+1} - b_k \in \{b_k, b_{k-1}\}$ and consisted of a series of lemmas.

We used [7] to help us do this. However, we observed that the two-variable functions $f(n,t)$ can be defined so easily that it is more convenient to reverse ourselves and also start with an arbitrary base B that satisfies the much more general definition 1. We then consider the lemmas of the basic theory to just be the properties that the two-variable functions $f(n,t)$ must satisfy. An added benefit is that we find all $f(n,t)$ for which our basic theory is effective since $f(n,t)$ is two variable instead of one variable and can easily be defined directly

from the lemmas. We cannot do this with one variable functions $f(t)$. This basic theory can then be applied to study increasingly complex single and 2-pile games exactly the same as before. Also the reader does not have to go through the proofs for a list of lemmas.

We will now list in table form the properties that $f(n,t)$ must satisfy in the basic theory. We will then discuss the compatibility of these properties.

In the following, $B = (b_1 = 1, b_2, b_3, \cdots)$ satisfies definition 1, and $f : N_0 \times N \to N_0$ is a function. Also, $\forall n \in N$, $n = b_{i_1} + b_{i_2} + \cdots + b_{i_t}, b_{i_1} < b_{i_2} < \cdots < b_{i_t}$ and each $b_{i_j} \in B$, is computed by algorithm 1. From algorithm 1 and lemma 1, we note the following. In P-5 if $1 \leq x < g(n)$, then $g(g(n) - x) = g(n - x)$. In P-6 if $1 \leq x < \overline{g}(n)$, then $g(n + x) = g(x)$. The value of these 9 properties will be apparent later in the paper. For example, see theorem 1 and remark 2. We will also explain why the applications are endless. Until then the reader will have to rely on faith.

	$\forall n \in N$ if	then
P-1	n is semi-stable	$f(n, \overline{g}(n)) < \overline{g}(\overline{g}(n) + n)$.
P-2	n is not semi-stable, n is non-binary, $\overline{g}(n) + n$ is binary	$f(n, \overline{g}(n)) < \overline{g}(\overline{g}(n) + n) = g(\overline{g}(n) + n)$.
P-3	n is not semi-stable, n is non-binary $\overline{g}(n) + n$ is non-binary	$\overline{g}(\overline{g}(n) + n) \leq f(n, \overline{g}(n)) < g(\overline{g}(n) + n)$.
P-4	n is not semi-stable	$f(n, g(n)) < \overline{g}(n - g(n))$.
P-5	$x \in N, 1 \leq x < g(n)$	$f(n,x) \geq g(n - x) = g(g(n) - x)$. see above note.
P-6	$x \in N, 1 \leq x < \overline{g}(n)$	$f(n,x) \geq g(n + x) = g(x)$. see above note.
P-7	n is non-binary $x \in N, \overline{g}(n) < x < g(n)$	$f(n,x) \geq g(n + x)$.
P-8	$n \in N$	$f(n, g(n)) < g(n - g(n))$.
P-9	$x \in N$	$f(0,x) \geq g(x)$.

There is just one compatibility condition that the base B must satisfy in order for $f : N_0 \times N \to N_0$ to exist that satisfies P-1, P-2, $\cdots$, P-9.

Compatibility Condition. $\forall n \in N$, if n is both non-binary and semi-stable, then $g(n - \overline{g}(n)) < \overline{g}(\overline{g}(n) + n)$ must be true in the base B.

Note. We will show that if $\overline{g} : B \to B$ is non-decreasing, then this compatibility condition is automatically met.

The reader might like to show that $f(n,t) = 2t, f(n,t) = t$ satisfy P-1, P-2, $\cdots$, P-9 for the Fibonacci, binary base respectively. See [7] for more about this. We will now show that the above compatibility condition is the only one that B must satisfy in order for $f : N_0 \times N \to N_0$ to exist that satisfies P-1, P-2, $\cdots$, P-9.

Since P-9 is independent of P-1, $\cdots$, P-8, there is no compatibility requirements for P-9. So we consider only P-1, P-2, $\cdots$, P-8.

1. Let us first assume that $n \in N$ is binary. When n is binary, $\overline{g}(n) = g(n)$, Also, only P-1, P-4, P-5, P-6, P-8 apply. Note that $f : N_0 \times N \to N_0$ is a function into N_0 and $0 \in N_0$. Also, $\forall n, x \in N, f(n,x)$ can be made as large as we wish. So when n is binary, it is trivial to see that there are no compatibility requirements on B in order for $f(n,x)$ to exist.

2. Next, suppose $n \in N$ is non-binary which means $1 \leq \overline{g}(n) < g(n)$. The only compatibility problem that might arise concerns the value of $f(n, \overline{g}(n))$. Now $f(n, \overline{g}(n))$ occurs only in P-1, P-2, P-3 and P-5. We will consider three subcases of (2).

   A. n is non-binary and semi-stable. P-1 and P-5 apply which means $g(n - \overline{g}(n)) \leq f(n, \overline{g}(n)) < \overline{g}(\overline{g}(n) + n)$. Since $1 \leq \overline{g}(n) < g(n) = b_{i_1}$, it is easy to see that $g(n - \overline{g}(n)) = g(g(n) - \overline{g}(n)) \leq b_{i_1 - 1}$. Also, since n is semi-stable, from lemma 2 we know that $\exists 2 \leq r \leq t$ such that $\overline{g}(n) + n = b_{i_r + 1} + b_{i_{r+1}} + \cdots + b_{i_t}$ is in stable form. This means $\overline{g}(\overline{g}(n) + n) = \overline{g}(b_{i_r + 1})$. However, there is no way that we can prove $g(n - \overline{g}(n)) < \overline{g}(b_{i_r + 1})$. Now if $\overline{g} : B \to B$ is a non-decreasing function, then $g(n - \overline{g}(n)) \leq b_{i_1 - 1} < b_{i_1 + 1} = \overline{g}(b_{i_2}) \leq \overline{g}(b_{i_r + 1}) = \overline{g}(\overline{g}(n) + n)$. This implies $g(n - \overline{g}(n)) \leq f(n, \overline{g}(n)) < \overline{g}(\overline{g}(n) + n)$ is compatible for $f(n, \overline{g}(n))$ to exist. However, in general we must assume that if n is non-binary and semi-stable then $g(n - \overline{g}(n)) < \overline{g}(\overline{g}(n) + n)$ is true in the base B.

   B. n is non-binary and not semi-stable, and $\overline{g}(n) + n$ is binary. Only P-2 and P-5 apply and we must have $g(n - \overline{g}(n)) \leq f(n, \overline{g}(n)) < \overline{g}(\overline{g}(n) + n) = g(\overline{g}(n) + n)$. Now since n is not semi-stable, $\overline{g}(n) + n = \overline{g}(b_{i_1}) + b_{i_1} + b_{i_2} + \cdots + b_{i_t} =$

$b_{i_1+1}+b_{i_2}+\cdots+b_{i_t}$ is in stable form since if $t \geq 2$ then $b_{i_1+1} < \overline{g}(b_{i_2})$. Therefore, $g(\overline{g}(n)+n) = b_{i_1+1}$ and $g(n-\overline{g}(n)) \leq b_{i_1-1} < g(\overline{g}(n)+n)$ implies $g(n-\overline{g}(n)) \leq f(n,\overline{g}(n)) < g(\overline{g}(n)+n)$ is compatible for $f(n,\overline{g}(n))$ to exist.

C. n is non-binary and not semi-stable, and $\overline{g}(n)+n$ is non-binary. Only P-3 and P-5 apply and we must have $\overline{g}(\overline{g}(n)+n) \leq f(n,\overline{g}(n)) < g(\overline{g}(n)+n)$ and $g(n-\overline{g}(n)) \leq f(n,\overline{g}(n))$. Now $\overline{g}(\overline{g}(n)+n) < g(\overline{g}(n)+n)$ since $\overline{g}(n)+n$ is non-binary. So the only compatibility requirement for $f(n,\overline{g}(n))$ to exist is $g(n-\overline{g}(n)) < g(\overline{g}(n)+n)$, and this is obviously true since $g(n-\overline{g}(n)) \leq b_{i_1-1} < g(\overline{g}(n)+n) = b_{i_1+1}$.

Section 3. We will now develop the machinery for the endless applications of the basic theory.

Definition 8. *Let B satisfy definition 1 and the compatibility condition. If B is fixed, we define $h: N_0 \to \{\infty\} \cup N$ to be any arbitrary but fixed function that satisfies the conditions*

A. $h(0) = \infty$

B. $\forall n \in N, h(n) \in \{\overline{g}(n), g(n)\}$.

Since $\overline{g}(0) = g(0) = \infty$ and $\forall n \in N, \overline{g}(n) \leq g(n)$, we know that $\forall n \in N_0, \overline{g}(n) \leq h(n) \leq g(n)$.

Definition 9. *Let B satisfy definition 1 and the compatibility condition. Also, $f: N_0 \times N \to N_0$ satisfies P-1, P-2, $\cdots$,P-9, and $h: N_0 \to \{\infty\} \cup N$ satisfies definition 8. Then $g': N_0 \to \{\infty\} \cup N$ is defined as follows.*

A. $g'(0) = \infty$

B. $\forall n \in N, g'(n)$ *is the smallest* $x \in N$ *such that (1) or (2) is true for* x.

1. $f(n,x) < h(n-x)$ *or*
2. $f(n,x) < h(n+x)$.

Note. $\forall n \in N, h(n-n) = h(0) = \infty$ implies $1 \leq g'(n) \leq n$.

Definition 9′ (generalized). Let B satisfy definition 1 and the compatibility condition. Also, $f : N_0 \times N \to N_0$ satisfies P-1, P-2, $\cdots$, P-9, and $\forall i \in \{1, 2, \cdots, m\}, h_i : N_0 \to \{\infty\} \cup N$ satisfies definition 8.

Then $g' : N_0 \to \{\infty\} \cup N$ is defined as follows.

A′. $g'(0) = \infty$.

B′. $\forall n \in N, g'(n)$ is the smallest $x \in N$ such that (1′) or (2′) is true for x.

1′. $\exists i \in \{1, 2, \cdots, m\}$ such that $f(n, x) < h_i(n - x)$ or

2′. $\exists i \in \{1, 2, \cdots, m\}$ such that $f(n, x) < h_i(n + x)$.

Note. Definition 9′ is used in complicated two-pile games in which counters can be removed and/or added back using arbitrarily complex rules as we explain in the appendix. However, 9′ is not used in this paper.

Theorem 1 *Suppose B satisfies definition 1 and the compatibility condition. Also $f : N_0 \times N \to N_0$ satisfies P-1, P-2, $\cdots$, P-9, and $h : N_0 \to \{\infty\} \cup N$ satisfies definition 8. $\forall n \in N$, let $n = b_{i_1} + b_{i_2} + \cdots + b_{i_t}, b_{i_1} < b_{i_2} < \cdots < b_{i_t}$ and each $b_{i_j} \in B$, be computed by algorithm 1. Also, g' is computed from definition 9. For all $n \in N$, conclusions A, B, C follow from P-1, P-2, $\cdots$, P-9, and the fact that $[\overline{g}(x) \leq g(x)$ and $h(x) \in \{\overline{g}(x), g(x)\}]$ implies $\overline{g}(x) \leq h(x) \leq g(x)$.*

A. Suppose n is semi-stable. Then $g'(n) = \overline{g}(n) = \overline{g}(g(n)) = \overline{g}(b_{i_1})$, and $f(n, g'(n)) < h(n + g'(n))$.

B. Suppose n is binary. Then $g'(n) = g(n) = \overline{g}(n) = b_{i_1}$.

If n is binary and semi-stable, then from conclusion A, $f(n, g'(n)) < h(n + g'(n))$. If n is binary and not semi-stable, then $f(n, g'(n)) < h(n - g'(n))$.

C. Suppose n is non-binary and not semi-stable. Then (a) or (b) is true.

(a) Suppose $\overline{g}(n) + n$ is binary. Then $g'(n) = \overline{g}(n) = \overline{g}(b_{i_1}) < b_{i_1}$ and $f(n, g'(n)) < h(n + g'(n))$.

(b) Suppose $\overline{g}(n) + n$ is non-binary.

1. If $h(\overline{g}(n)+n) = g(\overline{g}(n)+n)$, then $g'(n) = \overline{g}(n) = \overline{g}(b_{i_1}) < b_{i_1}$ and $f(n, g'(n)) < h(n+g'(n))$.
2. If $h(\overline{g}(n)+n) = \overline{g}(\overline{g}(n)+n)$, then $g'(n) = g(n) = b_{i_1}$ and $f(n, g'(n)) < h(n-g'(n))$.

Since theorem 1 is the main tool that we will now use, the reader can forget many of the previous details.

Remark 2 *Note that since $g'(0) = \infty$, theorem 1 means that g' satisfies definition 8. This means that if g' is used in the place of h, then g'' also satisfies definition 8. Also, $g''', g'''', \cdots$ all satisfy definition 8.*

The closure of $g', g'', g''', \ldots$ is one reason why the applications of the basic theory are endless. It is very important to note that $g'(n)$ depends *only* on the function h and the base B and is independent of $f(n,t)$. Of course, $f(n,t)$ must satisfy P-1, P-2, $\cdots$, P-9, and h must satisfy definition 8. But much more than this, $\forall n \in N, g'(n)$ depends on h in only one small way. $\forall n \in N, g'(n)$ depends on h if and only if (a), n is non-binary and (b) n is not semi-stable, and (c) $\overline{g}(n)+n$ is non-binary. When all three of (a), (b), (c) are true, then $g'(n)$ depends only on which of the two values $\overline{g}(\overline{g}(n)+n), g(\overline{g}(n)+n)$ is taken by $h(\overline{g}(n)+n)$. This is another reason why the applications are endless. We discuss this more in the appendix.

Remark 3 Let $h_i : N_0 \to \{\infty\} \cup N, i = 1, 2, \cdots, m$, satisfy definition 8 and define $g' : N_0 \to \{\infty\} \cup N$ as in definition 9′. Then theorem 1 remains valid with an obvious modification of conclusions c-a and c-b. This is another reason way the applications are endless.

Remark 4 (Important). Let B satisfy definition 1 and the compatibility condition. Also, $f : N_0 \times N \to N_0$ satisfies P-1, P-2, $\cdots$, P-9. Let us suppose that the domain of the h-function defined in definition 8 has been restricted to $\{0, 1, 2, \cdots, M\}$ and suppose $n \in \{0, 1, 2, \cdots, M\}$. As always, $g'(0) = \infty$. Modifying definition 9, $\forall n \in N$, let us define $g'(n)$ to be the smallest $x \in N$ such that (1^*) or (2^*) is true for x.

(1^*) $f(n,x) < h(n-x)$ or

(2^*) $f(n,x) < h(n+x), 1 \leq x \leq M-n$.

The machinery of theorem 1 will compute $g'(n)$ exactly the same as before as long as $\overline{g}(n)+n \leq M$.

In particular suppose the domain of the h-function is restricted to $\{0, 1, 2, 3, \cdots, b_{k+1}\}$.

We now show that if $1 \leq n < b_{k+1}$, then $\overline{g}(n)+n \leq b_{k+1}$ which means that theorem 1 remains effective in computing $g'(n)$. (1). Suppose $n = b_t \leq b_k$. Then $\overline{g}(b_t) + b_t = b_{t+1} \leq b_{k+1}$. (2). Suppose $b_t < n < b_{t+1} \leq b_{k+1}$. (a). If n is semi-stable, from lemma 2 we see that $\overline{g}(n) + n \leq b_{t+1} \leq b_{k+1}$. (b) If n is not semi-stable, from algorithm 1 and lemma 1, it is easy to see that $\overline{g}(n) + n < b_{t+1} < b_{k+1}$.

When $\overline{g}(n) + n > M$, the machinery of theorem 1 breaks down in computing $g'(n)$, and we must rely totally on properties P-4, P-5, P-6 and P-8. We will use remark 4 when we analyze game 2, the two-pile game described in the abstract.

Section 4. We now use our basic machinery to study two games. We will first analyze game 1, a very simple single-pile game.

Game 1. Let B satisfy definition 1. B need not satisfy the compatibility condition. Also let $f : N_0 \times N \rightarrow N_0$ satisfy P-5 and P-8.

Two players alternate removing positive numbers of counters from a single pile of counters. On the first move of the game, the player moving first can remove from the pile at most k counters, k being fixed at the start. On each subsequent move, a player can remove at most $f(n,t)$ counters where n was the pile size of the preceding position and t was the number of counters removed by his opponent on the preceding move. The winner is the last player to make a legal move. Of course, the moving player cannot move if $\min(f(n,t), n) = 0$.

As an example, suppose the moving player is facing a pile size of 10 counters and the preceding pile size was 15 counters. This means his opponent removed 5 counters on the preceding move. Also, suppose $f(15,5) = 7$. This means the moving player can remove from the 10 counter pile $1, 2, 3, 4, 5, 6$, or 7 counters. If $f(15,5) \geq 10$, the moving player could remove the entire 10 counters and immediately win. Also, if $f(15,5) = 0$, the moving player cannot make a legal move, and the other player won the game.

Analysis of Game 1. $\forall n \in N$, define $g_0(n)$ to be the smallest winning move size for a pile of n counters. Also, define $g_0(0) = \infty$. This means the removal of $g_0(n)$ counters from a pile of n counters is a winning move, but $\forall 1 \leq t < g_0(n)$ the removal of t counters from a pile of n counters is a losing move. Of course, $\forall n \in N, 1 \leq g_0(n) \leq n$. Since $g_0(0) = \infty$, it is easy to see that $\forall n \in N, g_0(n)$ is the smallest $x \in \{1, 2, 3, \cdots, n\}$ such that $f(n,x) < g_0(n-x)$.

From P-5 and P-8 it is easy to see by induction that $\forall n \in N, g_0(n) = g(n)$ where $g(n)$ is defined in definition 6.

Game 2. Two players alternate removing a positive number of counters from two piles.

On each turn the moving player chooses a pile and removes counters from this chosen pile. On the first move of the game, the player moving first can remove from one pile at most k counters, k being specified at the start. On each subsequent move, a player can remove from one pile at most $f(x-y,t)$ counters where t was the number of counters removed by his opponent on the preceding move, and x and $y, x \geq y$, are the two pile sizes in the preceding position. The winner is the last player to make a legal move. Of course, the moving player cannot move when $f(x-y,t)=0$ or $x=y=0$. B is a base that satisfies definition 1 and the compatibility condition, and $f: N_0 \times N \rightarrow N_0$ satisfies all of P-1, P-2, $\cdots$, P-9.

As an example, suppose the moving player is facing pile sizes $(15,10)$ and the pile sizes of the preceding position were $(30,10)$. This means his opponent removed 15 counters on the preceding move. Also, suppose $f(30-10,15)=f(20,15)=6$. This means the moving player can remove $1,2,3,\cdots,6$ from the 10 counter pile or he can remove $1,2,3,\cdots,6$ from the 15 counter pile. If $f(20,15)=0$, he cannot move at all, and he loses the game.

In the rest of this paper, we are studying Game 2 using a given B and f.

Notation 1. *Let the pile sizes of a position be x and $y, x \geq y$. Then the position can be denoted (x,y). However, it is more convenient to denote a position by (a,b) where $(a,b)=(x-y,x)$. This means that b is the larger pile size and a is the difference between the larger and smaller pile sizes. Note that the total number of counters is $2b-a$. Also, note that $0 \leq a \leq b$,*

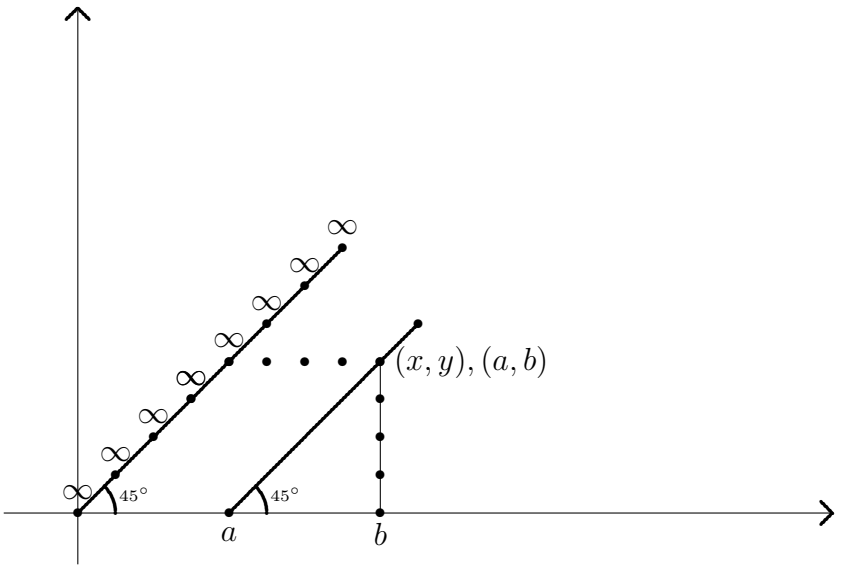

Fig. 1 $(a,b) = (x-y, x)$.

Definition 10. $\forall$ *position* $(a,b), 0 \leq a \leq b$, *we define* $g'(a,b)$ *to be the smallest winning move size, where* $g'(0,0) = \infty$. *This means a winning move is to remove* $g'(a,b)$ *counters from one of the two piles. Of course,* $g'(a,b)$ *by itself does not necessarily tell the player from which pile* $g'(a,b)$ *can be removed. Also,* $\forall 1 \leq t < g'(a,b)$, *the removal of* t *counters is a losing move no matter from which of the two piles* t *is removed. Also,* $g'(a,b) = \infty$ *means that all moves are losing moves.*

Algorithm 2. *In theorem 2, we prove that* $\forall b \geq 0, g'(0,b) = \infty$. *Using this fact, we see that* $\forall 1 \leq a \leq b, g'(a,b)$ *must be the smallest* $t \in \{1, 2, \cdots, a\}$ *such that (1) or (2) is true for* t.

(1) $f(a,t) < g'(a-t, b-t)$, where $1 \leq t \leq a$, or (2) $f(a,t) < g'(a+t, b)$, where $1 \leq t \leq b-a$. In (2) we could write $1 \leq t \leq min\{a, b-a\}$ but $1 \leq t \leq b-a$ is more convenient.

To understand (1) and (2), note that if t is removed from the larger pile in the position (a,b), the new position becomes $(a-t, b-t)$, and if t is removed from the smaller pile, the new position becomes $(a+t, b)$. The reader can visualize this using figure 1. Since $a = x-y$, algorithm 2 should now be easy to see.

In theorem 2 in addition to $\forall b \geq 0, g'(0,b) = \infty$, we also show by induction that $\forall a, b$ such that $1 \leq a \leq b, g'(a,b) \in \{\overline{g}(a), g(a)\}$. This means that $g'(a-t, b-t) \in$

$\{\overline{g}(a-t), g(a-t)\}$ when $1 \leq t < a$, and $g'(a+t,b) \in \{\overline{g}(a+t), g(a+t)\}$, when $1 \leq t \leq b-a$. From this we see in (1) and (2) of algorithm 2 that $g'(a-t,b-t)$ and $g'(a+t,b)$ are playing the roles of $h(n-t)$ and $h(n+t)$ respectively, and $g'(a,b)$ is playing the role of $g'(n)$ in definition 9. Of course, in $g'(a+t,b)$, t is restricted to $1 \leq t \leq b-a$. However, from remark 4, this will not cause us a problem in using theorem 1 when we need it to prove theorem 2. The reader should be able to quickly interpret (1). $f(a,t) < g'(a-t,b-t)$, (2) $f(a,t) < g'(a+t,b)$ in light of definition 9.

Definition 11. *$\forall b_k \in B$, let b_j be the smallest member of $\{b_k, b_{k+1}, b_{k+2}, \cdots\}$ such that b_j is binary if such a b_j exists. We define the binary degree of b_k as $d_b(b_k) = j-k$. If no such b_j exists, we define $d_b(b_k) = \infty$. Of course, $d_b(b_k) = 0$ if b_k itself is binary.*

Definition 12. *Suppose (b_k, b) is a position where $b_k \in B, b_k \leq b$. Also, suppose $b_h \leq b < b_{h+1}$. Then the degree of (b_k, b), denoted $d(b_k, b)$, is $d(b_k, b) = \min\{d_b(b_k), h-k\}$, where min denotes the minimum and $d_b(b_k)$ is the binary degree of b_k. Of course, $d(b_k, b)$ is always finite and non-negative. Note that $d(b_k, b_k) = 0$*

In the following theorem 2, note that the function $g' : N_0 \times N_0 \to \{\infty\} \cup N$ depends only on the base B. We conjecture that if B satisfies definition 1 and $f : N_0 \times N \to N_0$ is an arbitrary function, then theorem 2 is true for (B, f) if and only if f satisfies P-1, P-2, $\cdots$, P-9.

Theorem 2 *Suppose $(a,b), 0 \leq a \leq b$, is an arbitrary, position in Game 2. Then the following is true.*

1. *If $a = b, g'(a,a) = g(a)$.*
2. *If $a = 0, g'(0,b) = \infty$.*
3. *If $1 \leq a \leq b, g'(a,b) \in \{\overline{g}(a), g(a)\}$.*
4. *If $a = b_k \in B$, then*
   (a) *$g'(b_k, b) = b_k$ if $d(b_k, b)$ is even, and*
   (b) *$g'(b_k, b) = \overline{g}(b_k)$ and $\overline{g}(b_k) < b_k$ if $d(b_k, b)$ is odd.*
5. *If $b_k < a < b_{k+1}$, then*
   (a) *$g'(a,b) = g'(a, b_{k+1})$ if $b_{k+1} < b$, and*

(b) $g'(a,b) = g'(a-b_k, b-b_k)$, *if* $b_k < b \leq b_{k+1}$.

Remark 5 A rapid calculation of $g'(a,b)$ based on Theorem 2, and the winning moves of Game 2 are included at the end of the paper.

Proof. Conclusion 1 is obvious from Game 1 since the position (a,a) is a single pile game and $g'(a,a) = g_0(a) = g(a)$. We will now deal with conclusions 2-5 by induction on the quantity $2b-a$, which is the total number of counters in both piles. Now when $2b-a \in \{0,1\}$, we have no counters or one counter. This gives the two positions $(0,0)$ and $(1,1)$, and the conclusions that apply are obviously true since $g'(0,0) = \infty, g'(1,1) = 1$. Note that $1 \in B$ and 1 is always binary. So the induction is started, and we can just focus our attention on an arbitrary position $(a,b), 0 \leq a \leq b$.

Let us first prove conclusion 4 for (a,b), which means $(a,b) = (b_k, b)$.

First, suppose $d(b_k, b) = 0$. From definition 12, this means $d_b(b_k) = 0$ or $b_k \leq b < b_{k+1}$. Since $d(b_k, b)$ is even, we need to show $g'(b_k, b) = b_k$. From induction and algorithm 2, $g'(b_k, b)$ is the smallest $t \in \{1, 2, \cdots, b_k\}$ such that (1) or (2) is true for t.

1. $f(b_k, t) < g'(b_k - t, b-t), 1 \leq t \leq b_k$, or
2. $f(b_k, t) < g'(b_k + t, b), 1 \leq t \leq b - b_k$.

By induction $g'(b_k - b_k, b-b_k) = g'(0, b-b_k) = \infty, \forall 1 \leq t < b_k, g'(b_k - t, b-t) \in \{\overline{g}(b_k - t), g(b_k - t)\}$, and $\forall 1 \leq t \leq b - b_k, g'(b_k + t, b) \in \{\overline{g}(b_k + t), g(b_k + t)\}$.

A. First, suppose $d_b(b_k) = 0$, which means b_k is binary. Therefore, $\overline{g}(b_k) = b_k$ and $b_{k+1} = b_k + b_k = 2b_k$.

From property P-6 and the last few sentences, we see that if $1 \leq t \leq b - b_k$ and $t < \overline{g}(b_k) = b_k$, then $f(b_k, t) \geq g(b_k + t) \geq g'(b_k + t, b)$. Therefore, condition (2) of algorithm 2 is impossible when $1 \leq t < b_k$. Also, from property P-5, and the last few sentences, when $1 \leq t < b_k = g(b_k), f(b_k, t) \geq g(b_k - t) \geq g'(b_k - t, b-t)$.

Therefore, condition 1 of algorithm 2 is also impossible when $1 \leq t < b_k$. However, $f(b_k, b_k) < g'(b_k - b_k, b-b_k) = g'(0, b-b_k) = \infty$. This means condition (1) is satisfied when $t = b_k$, which means $g'(b_k, b) = b_k$.

B. Still supposing $d(b_k, b) = 0$, let $d_b(b_k) \geq 1$. This means $b_k \leq b < b_{k+1}$. Since $d_b(b_k) \geq 1$, this means b_k is non-binary. That is, $b_{k+1} - b_k = \overline{g}(b_k) < b_k$. In the above condition (2), algorithm 2, we know that $1 \leq t \leq b - b_k < b_{k+1} - b_k = \overline{g}(b_k)$. That is, $1 \leq t < \overline{g}(b_k)$. Now when $1 \leq t < \overline{g}(b_k)$, we can use property P-6 to see that $f(b_k, t) \geq g(b_k + t) \geq g'(b_k + t, b)$. Therefore, condition (2), algorithm 2 is impossible when $1 \leq t < \overline{g}(b_k)$. Also, repeating the argument in A, condition (1), algorithm 2 is also impossible when $1 \leq t < b_k$. Of course, condition (1), algorithm 2 holds when $t = b_k$ since $f(b_k, b_k) < g'(b_k - b_k, b - b_k) = g'(0, b - b_k) = \infty$. Combining A, B means $g'(b_k, b) = b_k$ when $d(b_k, b) = 0$.

Next, suppose $d(b_k, b) \geq 1$. This means that $b_{k+1} \leq b$ and $d_b(b_k) \geq 1$. This means that b_k is non-binary, which means $\overline{g}(b_k) < b_k$. Let us now observe that $\overline{g}(b_k) + b_k = b_{k+1} \leq b$. From the second paragraph of remark 4 (using $n = b_k, M = b$) this means that we will be able to apply theorem 1 directly to algorithm 2 since the domain of the h-function in theorem 1 that is used in algorithm 2 is big enough to allow this.

A. First, suppose b_{k+1} is binary. From definition 11, $d_b(b_k) = 1$ and from definition 12, $d(b_k, b) = 1$, which is odd. Therefore, we must show that $g'(b_k, b) = \overline{g}(b_k) < b_k$. Since b_k is non-binary and, of course, not semi-stable and since $\overline{g}(b_k) + b_k = b_{k+1}$ is binary, we can apply c-a, theorem 1 directly to algorithm 2 to see that $g'(b_k, b) = \overline{g}(b_k) < b_k$.

B. Next, suppose $\overline{g}(b_k) + b_k = b_{k+1}$ is non-binary. Since b_k is non-binary and not semi-stable, from c-b, theorem 1 applied to algorithm 2, we know that $g'(b_k, b)$ is determined by the value of $g'(\overline{g}(b_k) + b_k, b) = g'(b_{k+1}, b)$. Now if $d(b_k, b)$ is odd, we know from definitions 11, 12 that $d(b_{k+1}, b) = d(b_k, b) - 1$ is even, and by induction $g'(b_{k+1}, b) = b_{k+1}$. From c-b-1, theorem 1, this means that $g'(b_k, b) = \overline{g}(b_k) < b_k$. If $d(b_k, b)$ is even, then $d(b_{k+1}, b)$ is odd, and by induction $g'(b_{k+1}, b) = \overline{g}(b_{k+1})$. From c-b-2, theorem 1, this means that $g'(b_k, b) = g(b_k) = b_k$. This completes the proof of conclusion 4, theorem 2.

Next we prove conclusion 2 of theorem 2. This means $(a, b) = (0, b)$, and the two piles are equal. By symmetry, $g'(0, b)$ is the smallest $1 \leq t \leq b$ such that $f(0, t) < g'(t, b)$. Now by induction with conclusion 3, $g'(t, b) \in \{\overline{g}(t), g(t)\}$. Also, from property P-9, we know that $\forall t \in N, f(0, t) \geq g(t) \geq \overline{g}(t)$. This means $f(0, t) < g'(t, b)$ is impossible which means $g'(0, b) = \infty$.

We now prove conclusion 5 for (a,b). We will prove 5-b first, which means $b_k < a \leq b \leq b_{k+1}$ and $b_k < a < b_{k+1}$.

We will consider two subcases of this. Subcase 2 will be largely a repeat of the argument given for subcase 1.

Subcase 1. $b < b_{k+1}$.

Subcase 2. $b = b_{k+1}$.

Subcase 1. When $a = b$, from conclusion 1 (which we have already proved), algorithm 1, definition 6 (also see lemma 1) and the fact that $b_k < a < b_{k+1}$, we see that $g'(a,a) = g(a), g'(a-b_k, a-b_k) = g(a-b_k) = g(a)$. This implies $g'(a,a) = g'(a-b_k, a-b_k)$. So we can now assume that $b_k < a < b < b_{k+1}$ and we must prove that $g'(a,b) = g'(a-b_k, b-b_k)$. As always, $g'(a,b)$ is the smallest $t \in \{1,2,3,\cdots,a\}$ such that

1. $f(a,t) < g'(a-t, b-t), 1 \leq t \leq a$ or

2. $f(a,t) < g'(a+t, b), 1 \leq t \leq b-a$.

Let us now show how (1) and (2) compare with the computation of $g'(a-b_k, b-b_k)$. Now, $g'(a-b_k, b-b_k)$ is the smallest $t \in \{1,2,\cdots,a-b_k\}$ such that

1′. $f(a-b_k, t) < g'(a-b_k-t, b-b_k-t), 1 \leq t \leq a-b_k$ or

2′. $f(a-b_k, t) < g'(a-b_k+t, b-b_k), 1 \leq t \leq b-a$.

In (2′) we note that $1 \leq t \leq (b-b_k)-(a-b_k) = b-a$. This means that in both (2) and (2′) we have $1 \leq t \leq b-a$.

In (1), (1′) we know by induction with conclusion 2 that $g'(a-a, b-a) = g'(0, b-a) = \infty$, and $g'(a-b_k-(a-b_k), b-b_k-(a-b_k)) = g'(0, b-a) = \infty$. Also, in (1), (2), (1′), (2′) for the other values of t, we know by induction with conclusion 3 that
(1) $g'(a-t, b-t) \in \{\overline{g}(a-t), g(a-t)\}$,
(2) $g'(a+t, b) \in \{\overline{g}(a+t), g(a+t)\}$,
(3) $g'(a-b_k-t, b-b_k-t) \in \{\overline{g}(a-b_k-t), g(a-b_k-t)\}$ and
(4) $g'(a-b_k+t, b-b_k) \in \{\overline{g}(a-b_k+t), g(a-b_k+t)\}$.
As we previously explained, this means that both (1), (2) and (1′), (2′) are in the exact same form as definition 9 with the h-function having a restricted domain. Therefore, from remark 4, we must consider 2 cases, namely the case where theorem 1 is effective and where it is not. We observe that $g(a) = g(a-b_k), \overline{g}(a) = \overline{g}(a-b_k)$.

Case A. $\overline{g}(a) + a \leq b$, and equivalently $\overline{g}(a - b_k) + (a - b_k) \leq b - b_k$.

In case A, theorem 1 computes both $g'(a, b)$ and $g'(a - b_k, b - b_k)$.

Case B. $\overline{g}(a) + a > b$, and equivalently $\overline{g}(a - b_k) + (a - b_k) > b - b_k$.

In case B, theorem 1 computes neither $g'(a, b)$ nor $g'(a - b_k, b - b_k)$.

Case A. From remark 4, we know that theorem 1 remains effective in computing both $g'(a, b)$ and $g'(a - b_k, b - b_k)$. Using algorithm 1, let us first write $a = b_{i_1} + b_{i_2} + \cdots + b_{i_{t-1}} + b_k, b_{i_1} < b_{i_2} < \cdots < b_k$, each $b_{i_j} \in B$. Of course, $t \geq 2$ since $b_k < a < b_{k+1}$. We will now go through the conclusions of theorem 1 for a and $a - b_k$ in the order B, A, C.

B. First, we note that a is binary if and only if $a - b_k$ is binary. Now if a and $a - b_k$ are binary, from B, theorem 1, $g'(a, b) = g(a)$, and $g'(a - b_k, b - b_k) = g(a - b_k) = g(a)$, which means $g'(a, b) = g'(a - b_k, b - b_k)$. Therefore, we can now assume that both a and $a - b_k$ are non-binary which means $\overline{g}(a) = \overline{g}(a - b_k) = \overline{g}(b_{i_1}) < b_{i_1}$.

A. Suppose a is non-binary and semi-stable. From conclusion A, theorem 1, $g'(a, b) = \overline{g}(a) = \overline{g}(b_{i_1}) < b_{i_1}$. Now if $t \geq 3$, we know that $a - b_k$ is also semi-stable. From conclusion A, $g'(a - b_k, b - b_k) = \overline{g}(a - b_k) = \overline{g}(b_{i_1}) < b_{i_1}$. This implies $g'(a, b) = g'(a - b_k, b - b_k)$. So let us suppose a is semi-stable and $t = 2$. We will show that this is impossible. Now if $t = 2$ and a is semi-stable, then $a = b_{i_1} + b_k$ and $b_{i_1+1} = \overline{g}(b_k)$. This means $\overline{g}(a) + a = \overline{g}(b_{i_1}) + b_{i_1} + b_k = b_{i_1+1} + b_k = b_{k+1}$. However, since we are in case A, we know that $\overline{g}(a) + a \leq b$. This means $b_{k+1} \leq b$. However, since we are also in subcase 1, we know that $b < b_{k+1}$. So we have a contradiction.

C. Suppose a is non-binary and not semi-stable. This means $a - b_k$ is also non-binary and not semi-stable. Therefore, both a and $a - b_k$ come under conclusion c, theorem 1. We will consider cases c-a, c-b, theorem 1.

C-a. Suppose $\overline{g}(a) + a$ is binary. This means $\overline{g}(a - b_k) + (a - b_k) = (\overline{g}(a) + a) - b_k$ is binary since a is not semi-stable. Therefore, from c-a, theorem 1, $g'(a, b) = \overline{g}(a) = \overline{g}(b_{i_1}) < b_{i_1}$, and $g'(a - b_k, b - b_k) = \overline{g}(a - b_k) = \overline{g}(b_{i_1}) < b_{i_1}$. Therefore, $g'(a, b) = g'(a - b_k, b - b_k)$.

C-b. Suppose $\overline{g}(a) + a$ is non-binary which means $\overline{g}(a - b_k) + (a - b_k) = (\overline{g}(a) + a) - b_k$ is non-binary since a is not semi-stable. From c-b-1, c-b-2, theorem 1, we see that $g'(a, b)$ is determined completely by the value of $g'(\overline{g}(a) + a, b)$. Also, $g'(a - b_k, b - b_k)$ is determined completely and in exactly the same way by the value of $g'(\overline{g}(a - b_k) +$

$(a-b_k), b-b_k) = g'(\overline{g}(a)+a-b_k, b-b_k)$. Now from the definitions of case A and subcase 1, $b_k < a < \overline{g}(a)+a \leq b < b_{k+1}$. Therefore by induction with conclusion 5-b, $g'(\overline{g}(a)+a, b) = g'(\overline{g}(a)+a-b_k, b-b_k) = g'(\overline{g}(a-b_k)+(a-b_k), b-b_k)$. Now since a is not semi-stable, it is easy to see that $g(\overline{g}(a)+a) = g(\overline{g}(a-b_k)+a-b_k)$ and $\overline{g}(\overline{g}(a)+a) = \overline{g}(\overline{g}(a-b_k)+a-b_k)$. Of course, we already know $g(a) = g(a-b_k)$ and $\overline{g}(a) = \overline{g}(a-b_k)$. Using this with c-b-1 and c-b-2, theorem 1, we see that $g'(a,b) = g'(a-b_k, b-b_k)$.

Case B. $\overline{g}(a)+a > b$ and equivalently $\overline{g}(a-b_k)+(a-b_k) > b-b_k$. From remark 4, we know that theorem 1 is not effective in computing $g'(a,b)$ and $g'(a-b_k, b-b_k)$. From property P-6, we know that $\forall 1 \leq x < \overline{g}(n), f(n,x) \geq g(n+x)$ which implies $f(n,x) \geq h(n+x)$ when h satisfies definition 8. Let us apply this to algorithm 2. Since $\overline{g}(a)+a > b$ and since $\overline{g}(a-b_k)+(a-b_k) > b-b_k$, it follows that $g'(a,b)$ and $g'(a-b_k, b-b_k)$ are computed by the following simplified algorithms: $g'(a,b)$ is the smallest $t \in \{1,2,\cdots,a\}$ such that (1) $f(a,t) < g'(a-t, b-t), 1 \leq t \leq a$. Also, $g'(a-b_k, b-b_k)$ is the smallest $t \in \{1,2,\cdots,a-b_k\}$ such that (1′) $f(a-b_k, t) < g'(a-b_k-t, b-b_k-t), 1 \leq t \leq a-b_k$.

As stated before, in (1), we know by induction that $g'(a-a, b-a) = g'(0, b-a) = \infty$, and for $1 \leq t < a, g'(a-t, b-t) \in \{\overline{g}(a-t), g(a-t)\}$. The same is true for (1′).

Using algorithm 1, let $a = b_{i_1} + b_{i_2} + \cdots + b_{i_{t-1}} + b_k, b_{i_1} < b_{i_2} < \cdots < b_k$ and each $b_{i_j} \in B$. Of course, $t \geq 2$ since $b_k < a < b_{k+1}$. From property P-5, we know that $\forall 1 \leq t < g(a), f(a,t) \geq g(a-t)$ which implies $f(a,t) \geq g'(a-t, b-t)$. Also, from P-5, $\forall 1 \leq t < g(a-b_k) = g(a), f(a-b_k, t) \geq g'(a-b_k-t, b-b_k-t)$. Since by induction with conclusion 3, $g'(a-b_k, b-b_k) \in \{\overline{g}(a-b_k), g(a-b_k)\} = \{\overline{g}(a), g(a)\}$ and since $\overline{g}(a) \leq g(a)$, we know from the preceding sentence that $g'(a-b_k, b-b_k) = g(a-b_k) = g(a)$ must be true. This means that we must show that $g'(a,b) = g(a)$ which means we must show that $f(a, g(a)) < g'(a-g(a), b-g(a))$. If we can show that $g'(a-g(a), b-g(a)) = g(a-g(a))$, then from property P-8 we see that $f(a, g(a)) < g(a-g(a))$. And this means $g'(a,b) = g(a) = g'(a-b_k, b-b_k)$.

For the position $(a-g(a), b-g(a))$, let us show that $\overline{g}(a-g(a))+a-g(a) > b-g(a)$. Now $\overline{g}(a-g(a)) = \overline{g}(b_{i_2})$. Therefore, this inequality is true if and only if $\overline{g}(b_{i_2})+a > b$. Since we are in case B, we know that $\overline{g}(a)+a > b$. That is, $\overline{g}(b_{i_1})+a > b$. But since $a = b_{i_1} + b_{i_2} + \cdots + b_k$ is in stable form, we know that $\overline{g}(b_{i_1}) \leq b_{i_1} < b_{i_1+1} \leq \overline{g}(b_{i_2})$. Therefore, $\overline{g}(b_{i_2})+a > b$ is true. Therefore, we now know that $\overline{g}(a-g(a))+a-g(a) > b-g(a)$. Now by induction we know that $g'(a-g(a), b-g(a)) \in \{\overline{g}(a-g(a)), g(a-g(a))\}$. Now

if $\overline{g}(a-g(a)) = g(a-g(a))$, then $g'(a-g(a), b-g(a)) = g(a-g(a))$. So suppose $\overline{g}(a-g(a)) < g(a-g(a))$. From the general algorithm 2 for computing $g'(a-g(a), b-g(a))$, from the fact that $g'(a-g(a), b-g(a)) \in \{\overline{g}(a-g(a)), g(a-g(a))\}$ and the fact that $\overline{g}(a-g(a))+a-g(a) > b-g(a)$ and from property P-5, we know that $g'(a-g(a), b-g(a)) \neq \overline{g}(a-g(a))$. Therefore, $g'(a-g(a), b-g(a)) = g(a-g(a))$, which completes case B.

Subcase 2. $b = b_{k+1}$. From remark 4, we see that $\overline{g}(a)+a \leq b_{k+1} = b$. Equivalently we see that $\overline{g}(a-b_k)+a-b_k \leq b-b_k = b_{k+1}-b_k$. From remark 4, we know that theorem 1 is effective in computing both $g'(a,b) = g'(a,b_{k+1})$ and $g'(a-b_k, b-b_k) = g'(a-b_k, b_{k+1}-b_k)$. If we handle subcase 2 by copying nearly word for word the proof given for case A of subcase 1, we will discover just one flaw. This flaw occurs at the point in the proof where all of the following are true: (1) $a, a-b_k$ are non-binary; (2) a is semi-stable; and (3) $t = 2$ which means $a = b_{i_1} + b_k$. We will correct this flaw.

Since a is semi-stable, $b_{i_1+1} = \overline{g}(b_k)$. Since a is semi-stable and non-binary, from A, theorem 1, $g'(a,b) = g'(a,b_{k+1}) = \overline{g}(a) = \overline{g}(b_{i_1}) < b_{i_1}$.

We must show that $g'(a-b_k, b-b_k) = g'(b_{i_1}, b_{k+1}-b_k) = g'(b_{i_1}, \overline{g}(b_k)) = g'(b_{i_1}, b_{i_1+1}) = \overline{g}(b_{i_1}) < b_{i_1}$.

Now since b_{i_1} is non-binary, $d_b(b_{i_1}) \geq 1$. Therefore, from definition 12, $d(b_{i_1}, b_{i_1+1}) = \min(d_b(b_{i_1}), (i_1+1)-i_1) = 1$, an odd number. Therefore, from induction with conclusion 4, we know that $g'(b_{i_1}, b_{i_1+1}) = \overline{g}(b_{i_1}) < b_{i_1}$. Therefore, $g'(a, b_{k+1}) = g'(a-b_k, b_{k+1}-b_k)$.

Last, we prove 5-a of conclusion 5. This means $b_k < a < b_{k+1} < b$, and we need to show $g'(a,b) = g'(a,b_{k+1})$. Since $b_k < a < b_{k+1} < b$ we again see from remark 4 that $\overline{g}(a)+a \leq b_{k+1} < b$ which means that theorem 1 is effective in computing both $g'(a,b)$ and $g'(a,b_{k+1})$.

As always, we use algorithm 1 to write $a = b_{i_1} + b_{i_2} + \cdots + b_{i_t-1} + b_k, b_{i_1} < b_{i_2} < \cdots < b_k$, each $b_{i_j} \in B$. Again, $t \geq 2$ since $b_k < a < b_{k+1}$. We go through the conclusions of theorem 1 for a in the order B, A, C.

B. First, if a is binary, from conclusion B, $g'(a,b) = g'(a,b_{k+1}) = g(a) = \overline{g}(a)$. So let us now assume that a is non-binary.

A. If a is semi-stable, then from conclusion A, $g'(a,b) = g'(a,b_{k+1}) = \overline{g}(a)$. So we can now assume that a is non-binary and not semi-stable, which means a comes under C, theorem 1.

C. First, suppose $\overline{g}(a)+a$ is binary. From c-a, theorem 1, $g'(a,b) = g'(a,b_{k+1}) = \overline{g}(a)$.

Next, suppose $\overline{g}(a)+a$ is non-binary. From C-b, theorem 1, this means that $g'(a,b)$ is determined by the value of $g'(\overline{g}(a)+a,b) \in \{\overline{g}(\overline{g}(a)+a), g(\overline{g}(a)+a)\}$. Of course, from remark 4, if a is not semi-stable, then $b_k < a < \overline{g}(a)+a < b_{k+1} < b$. From C-b, theorem 1, we know that $g'(a,b_{k+1})$ is determined (in exactly the same way as $g'(a,b)$) by the value of $g'(\overline{g}(a)+a,b_{k+1}) \in \{\overline{g}(\overline{g}(a)+a), g(\overline{g}(a)+a)\}$.

Since $b_k < \overline{g}(a)+a < b_{k+1} < b$, we see by induction with conclusion 5-a, theorem 2, that $g'(\overline{g}(a)+a,b) = g'(\overline{g}(a)+a,b_{k+1})$.

Therefore, from C-b, theorem 1, we see that $g'(a,b) = g'(a,b_{k+1})$.

Last, we prove conclusion 3. Conclusion 3 for the position (a,b) follows readily from conclusions 1, 4, 5 for (a,b) and the induction of those positions that we have already taken care of. In conclusion 5-b, note that algorithm1, lemma 1, definition 6, and the fact that $b_k < a < b_{k+1}$ imply $g(a) = g(a-b_k)$ and $\overline{g}(a) = \overline{g}(a-b_k)$. This completes the proof of theorem 2. □

Theorem 3 shows how to compute $g'(a,b)$ efficiently.

Theorem 3 *Consider an arbitrary position (a,b) in Game 2 where $1 \le a < b$. Let $a = b_{i_1} + b_{i_2} + \cdots + b_{i_t}, b_{i_1} < b_{i_2} < \cdots < b_{i_t}$, each $b_{i_j} \in B$, be computed by algorithm 1. If $t = 1$, then $g'(a,b)$ is computed directly from conclusion 4, theorem 2. Therefore, suppose $t \ge 2$, and let $b = x + b_{i_2} + b_{i_3} + \cdots + b_{i_t}$. Of course, $b_{i_1} < x$ since $a < b$.*

1. *If $b_{i_1} < x < \overline{g}(b_{i_2})$, then $g'(a,b) = g'(b_{i_1},x)$.*
2. *If $\overline{g}(b_{i_2}) \le x$, then $g'(a,b) = g'(b_{i_1},\overline{g}(b_{i_2}))$. Of course, $g'(b_{i_1},x)$ and $g'(b_{i_1},\overline{g}(b_{i_2}))$ can be computed from conclusion 4, and $g'(a,a)$ can be computed from conclusion 1, theorem 2.*

Proof. The proof, which is based on theorem 2, is left as a challenge to the reader. □

Strategy for Game 2. First, for a position (a,b), we compute $g'(a,b)$. Theorem 1, properties P-1, P-2, $\cdots$,P-8 and simple logic are sufficient to tell the winning player how to move.

Therefore, suppose the winning player is confronted with a position (a,b). Since $g'(0,b) = \infty$, we must assume that $1 \le a \le b$.

1. Suppose $g'(a,b) = \overline{g}(a) < g(a)$. A winning move is to remove $\overline{g}(a)$ from the smaller pile and by property P-5 we know that the smaller pile must have $\overline{g}(a)$ counters in it.

2. Suppose $g'(a,b) = g(a)$.

   (a) If $\overline{g}(a) < g(a)$, a winning move is to remove $g(a)$ from the larger pile. Of course, if the smaller pile does not have $g(a)$ counters, we could not remove $g(a)$ from the smaller pile anyway. Therefore, if the smaller pile has $g(a)$ counters, we use c-b-1, theorem 1 to see this.

   (b) Suppose $\overline{g}(a) = g(a)$, that is, a is binary.

      1. If the smaller pile does not have $g(a)$, a winning move is to remove $g(a)$ from the larger pile.
      2. Suppose the smaller pile has $g(a)$. Then from A, B, theorem 1, we know the following.

         A. If a is semi-stable, a winning move is to remove $g(a)$ from the smaller pile.

         B. If a is not semi-stable, a winning move is to remove $g(a)$ from the larger pile.

Appendix We have studied over a dozen applications of the basic theory, and all of these are special cases of the following general model.

B is a base that satisfies both definition 1 and the compatibility condition. Also, $f : N_0 \times N \to N_0$ satisfies P-1, P-2, ..., P-9. Two players are facing one or two piles of counters, and they alternate moving. Using arbitrarily complex rules, each player on his turn does the following where $x \geq y$ are the two pile sizes in the preceding position and $t > 0$ was the move size of his opponent on the preceding move.

1. The moving player chooses $\overline{t} \in \{1, 2, 3, \ldots, f(x-y, t)\}$.

2. The moving player adds $\overline{t}$ counters to one of the piles or removes $\overline{t}$ counters from one of the piles in a way that depends on both his choice and the exact rules of the game. The number $\overline{t}$ is called the move size.

From our experience, we believe that an unlimited number of these games can be analyzed using the basic theory. In some cases we have restricted B to be the Fibonacci base, for example.

As an example, for the first k moves, where $k \geq 0$ and fixed at the start of the game, the moving player has the option of removing up to $f(x-y,t)$ counters from the pile of his choice or adding back up to $f(x-y,t)$ counters to the pile of his choice, where an empty pile is still considered to be a pile. But after k moves, the moving player can only *remove* up to $f(x-y,t)$ counters from the pile of his choice.

As another example, for the first k moves of the game, the moving player might be required to add back up to $f(x-y,t)$ counters to the pile of his choice. But after k moves, the moving player must remove up to $f(x-y,t)$ counters from the pile of his choice.

As a final example, for the first k, the moving player might be required to remove up to $f(x-y,t)$ counters from the larger pile. But after k moves, the moving player has the option of removing up to $f(x-y,t)$ counters from either pile as he chooses.

References

[1] Elwyn Berlekamp, John H. Conway, and Richard Guy, *Winning Ways*, Academic Press, Vol. 1, A. K. Peters, 2001.

[2] John H. Conway, *On Games and Numbers*, A. K. Peters, 2001.

[3] Epp, Richard and Thomas Ferguson, "A Note on Takeaway Games", *The Fibonacci Quarterly*, 18:4 (1980):300-303.

[4] J. Flanigan, "Generalized two-pile Fibonacci nim", Fibonacci Quart. 16, (1978): 459-469.

[5] J. Flanigan, "One-pile time and size dependent take-away games", Fibonacci Quart. 20, (1982): 51-59.

[6] Richard K. Guy, Fair Game, 2nd. ed., COMAP, New York, 1989.

[7] Holshouser, A., J. Rudzinski, and H. Reiter, "Dynamic One-Pile Nim", Fibonacci Quart. 41, (2003): 253-262.

[8] Holshouser, A., Achim Flammenkamp, and H. Reiter, Dynamic One-Pile Blocking Nim, *Electronic Journal of Combinatorics*, Volume 10(1), 2003.

[9] Holshouser, A., and H. Reiter, One Pile Nim with Arbitrary Move Function, *Electronic Journal of Combinatorics*, Volume 10(1), 2003. .

[10] A. J. Schwenk, "Take-Away Games", *The Fibonacci Quarterly*, 8:3 (1970):225-234.

ams classification: 91A46