Dean Thomas Allemang
Trinity College

Machine Computation with Finite Games

submitted for M. *Sc.*
on 7 September, 1984
revised February, 1985

DECLARATION BY AUTHOR OF THE DISSERTATION

Name of author in full DEAN THOMAS ALLEMANG
(block letters)

Title of dissertation as approved by the Board of Graduate Studies Machine computation with Finite games

1 I understand that I am the owner of the copyright of this dissertation and of the summary of the dissertation and that the copyright rests with me unless I specifically transfer it to another person.

2 I understand that the University requires that I shall deposit one copy of my dissertation and one copy of the summary in the University Library where they shall be available for consultation, and that photocopies of them shall be made available by the University Library to those who wish to consult them elsewhere. I understand that the Library, before allowing the dissertation or the summary to be consulted either in the original or in a photocopy, shall require each person wishing to consult them to sign a declaration that he recognises that the copyright of the dissertation and of the summary belongs to me and that no quotation from them and that no information derived from them may be published without my prior written consent.

3 I agree that, subject to any conditions decided upon by the Board of Graduate Studies under Regulation 16 of the Regulations for the Ph.D., M.Sc., and M.Litt. Degrees, my dissertation and summary shall be available for consultation in accordance with paragraph 2 above.

4 I agree that, subject to any conditions decided upon by the Board of Graduate Studies under Regulation 16 of the Regulations for the Ph.D., M.Sc., and M.Litt. Degrees, the summary of my dissertation shall be available for copying and publication at the discretion of the Board of Graduate Studies.

BOARD OF GRADUATE STUDIES
4 MILL LANE
CAMBRIDGE
CB2 1RZ

Signed Dean Thomas Allemang

Date 2 Sept 1984

G50

Thane -
with my compliments,
Dean

MACHINE COMPUTATION WITH FINITE GAMES

In this **work** I will discuss the application of a digital computer to the theory of games as .presented in Conway, ***On Numbers and Games*** [3] (henceforth, I will refer to this as **ONAG),** and Berlekamp, Conway and Guy, ***Winning Ways*** [1] (henceforth, **WW).** Often I have found that slight variants of the definitions given in **ONAG** and WW are more convenient for the computer's calculations. In these cases, I will describe the modified notation and definitions rather than the originals.

I will describe two projects, which treat widely different topics from **ONAG.** The first treats the theory of misère play of impartial games, the second deals with the theory of normal play of partizan games.

Misère play of impartial games

With the aid of the computer, I was able to obtain results much more extensive than those presented in WW and **ONAG.** For many of the octal games introduced in WW, I have been able to use these results to obtain complete solutions, which are catalogued in the final chapter. For many other games, I provide extensive incomplete solutions in appendix 11.

Normal play of partizan games

The theory from **ONAG** and WW has eliminated much of the combinatorial complexity from this domain, so the computer's impact is much less dramatic; it serves as a tabletop calculator to assist the games theorist in his calculations. In this work I describe how this system appears to the user, and a sample **of** its performance.

Except as stated explicitly in the text, this dissertation is the result of work of my own, and includes nothing which is the outcome of work done in collaboration.

[illegible]

MISÈRE PLAY OF IMPARTIAL GAMES

The games (or interchangeably, the positions) whose outcomes the computer has calculated are finite in nature. A game G is constructed as a finite set of games (called the *options* of G) which have already been constructed, and this is the only way in which games arise. G, its options, their options, and so forth, are the *positions* of G. This construction allows us, in any non-empty set of games, to find a (not necessarily unique) *simplest* game in that set, that is, a game s in that set, such that no other position of s is in that set. We write G' for a typical option of G, either to indicate a particular option, or all the options collectively, so that the set of options of G is $\{G'\}$. We identify a position with the set of its options, so we can write $G=\{G'\}$. These games are called *impartial* because the legal moves from any position are the same for both players. *Misère play* means that a player who has no legal move is declared the winner, and that this is the only way in which victory is decided. (c.f. Grundy & Smith, [9])

The sum of two positions G and H is defined in the usual Conway sense (**ONAG,** p. **74)** as the position in which a legal move is to choose a summand and make a move there. Thus $G + H = \{G' + H, G + H'\}$. Since these games are impartial, the same moves are available to both players, hence the players can only be distinguished by knowing whose turn it is to play. We have the *Nest* player, whose turn it is to play, and the *Previous* player, who we may as well suppose has just played. In a finite game in which every end position is considered a victory for some player, there exists a winning strategy for one of the players. If a game has a winning strategy for the next player, then we say that the game is an $\mathcal{N}$ position, otherwise it has a winning strategy for the previous player, so we call it a Pposition. Except for the game $0 = \{\}$ (the game with no options), which is an $\mathcal{N}$ position, a position is a $\mathcal{P}$ position in misère play if and only if all of its options are $\mathcal{N}$ positions. The value P or $\mathcal{N}$ associated with a position G is called its *outcome,* and is written as $o(G)$.

Since we are primarily concerned with the outcomes of sums of positions, we shall say that two positions G and H are *equal,* and shall write $G = H$, iff $o(G + T) = o(H + T)$ for all positions T. Two positions that look different when written as sets of options may, in fact, be equal in this sense. In **ONAG** chapter 12 Conway presents an elegant discussion of exactly when this happens. This rests on the notion of reversible moves. which I shall review.

To the position $G = \{A, B, C, \ldots\}$ we shall adjoin *reversible* moves X, Y, Z to get $H = \{A, B, C, \ldots, X, Y, Z\}$. For $G \neq 0$, we call these moves *reversible* if from each of X,Y,Z there is a move hack to a game equal to G. If $G = 0 = \{\}$. in addition to this we also require that one of X,Y,Z is a P position.

The things which we need to know about reversible moves are that:

1) For G, H as above, $G = H$
2) If neither G nor H has a reversible move, and $G = H$, then for any option G' of G we can find an option H' of H with $G' = H'$, and vice versa.

Detailed proofs of these can be found in ONAG chapter **12.**

Normal play of Impartial Games

Before we discuss miskre play, we shall review briefly some **of** the theory of normal play. [*]

We begin by examining the game of Nim. Nim is played with heaps of coins, the legal moves are to remove any number of coins from any one heap. The outcome is decided only when there are no coins remaining, and the player whose turn it is to move is unable to do so. Since we are considering **normal** play, this player loses, and the other player wins.

We will denote a nim heap **of** n coins by $*n$. In the notation of sets **of** options, we have that

$$*n = \{*(n-1), *(n-2), \ldots * 2, *1, *0\}$$

Notice that no play is possible from a heap of zero coins, so $*0 = 0$.

We define the outcome of a normal position just as we did **for** the miskre case, i.e., depending on which player has a winning strategy. **A** game is a P position exactly if no option of it is a P position. There is no exception for $*0$, so since $*0$ has no P options, $*0$ is a P position. We define sums and equality of games in the same way as we did in the miskre case.

It turns out that we have a simple, complete analysis of Nim, given by the following rule:

The sum of any number of nimheaps is equal in normal play to a single nimheap. The size of this single heap is the *nimsum* of the sizes of the summands.

The nimsum, which we write as $\overset{*}{+}$, is given by the following rules:

$$a \overset{*}{+} a = 0$$

$$2^m \overset{*}{+} 2^n = 2^m + 2^n \quad \text{for } m \neq n$$

To play Nim, we need only observe that the P positions are exactly those which equal *0. (c.f. Bouton [2])

This analysis is extended to an analysis of all impartial games by the following rule:

Any impartial game is equal in normal play to some nimheap. The size of this nimheap is called the *normal* Grundy *value* of the game. The normal Grundy value of a game is the *mex* of the normal Grundy values of its options. (c.f. Grundy [8] and Sprague [12])

Mex stands for minimal *excludant*. The *mex* of a set of integers is the least nonnegative integer which is not contained in that set; so that $\text{mex}\{0,1,4,7,12\} = 2$, $\text{mex}\{1,2,3,4,5\} = 0$. For this work, we need not worry about the mex of an infinite set.

Nimheaps, mex, and nim addition will all appear in the discussion of the miskre play of impartial games, with the same definitions as they have in the normal case.

[*] A detailed discussion of this can be found in ONAG chapter 11.

Misère Play

Following Conway in **ONAG,** we would like to write down some information **for** each position which will help us to compute outcomes **of** complicated games. The most obvious bit of information to keep about a game is its own outcome. This would be all the information we would need if we were always given a game as a set of options. However, a game will often appear as a sum of games. Hence, for any game G, we would like to write down the outcomes of $G+x$ for all $x \in X$, X some set of games. What should X be?

We would like X to have the property that if $x \in X$, then $x' \in X$ also. When this holds, we say that X is ***closed under descent.*** If we may assume that we have already computed all outcomes $G' + x \quad (x \in X)$, then this demand guarantees that we will be able to compute easily the outcomes $G + x$ as follows:

In order to compute the outcome of $G+x$ we need to know the outcomes of $G'+x$ and $G+x'$. The former we know by assumption, and, since X is closed, we have already computed the outcome of $G+x'$ (the construction for games allows us to partially order X *so* that we always compute $G+x'$ before $G+x$).

If we demand that the set X be closed under addition, then whenever we have written down the outcomes of $G+x$ for all $x \in X$, we have also written down, for any $y \in X$, all outcomes $G+y+x$, $x \in X$.

Theorem I will give us some guidance for choosing what set X to use. Before stating it, I must introduce the set of games we will call ***udders*** (c.f. the game ***udders*** on p. 409 of WW). We will write an adder as $:n$ for $n \geq 0$, and define them by the following formula:

$$:(2n) = \overbrace{*2 + \cdots + *2}^{n \text{ terms}}$$
$$:(2n+1) = :(2n) + *1$$

In terms of options, we have the following:

$$:0 = \{\} = *0 = 0$$
$$:1 = \{ :0\} = *1$$
$$:2 = \{ :0, :1\} = *2$$
$$:3 = \{ :0, :1, :2\} = *3$$
$$:4 = \{ :2, :3\}$$
$$:5 = \{ :2, :3, :4\}$$

$$:2n+2 = \{ :2n, :2n+1\}$$
$$:2n+3 = \{ :2n, :2n+1, :2n+2\}$$

Notice that $:0 = *0 = 0$, and $:1 = *1$.

Theorem I. The three sets $\{:0\}$, $\{:0, :1\}$ and $\{:0, :1, :2, \ldots\}$ (all 'adders') are all closed under both addition and descent. **Any** other set closed under both of these operations has each of these sets as subsets.

Proof.

The rule for addition tells us that $:0 + :0 = \{\} + \{\} = \{\} = :0$, so $\{:0\}$ is closed under addition. Clearly it is closed under descent.

The rule for addition tells us that

$$:1 + :1 = *1 + *1 = \{*0\} + \{*0\} = \{*1 + *0\} = \{*1\}$$

Now $*1 = \{*0\}$, and $*0$ is an $\mathcal{N}$ position, so $*1$ is a $\mathcal{P}$ position. Hence the move to $*1$ is reversible, whence $\{*1\} = \{\} = 0$, so that $:1 + :1 = :0$, and $\{0,1\}$ is closed under addition. Clearly it is closed under descent.

For any $n, m \geq 0$,

$$:n + :m = *2 + *2 + \ldots * 2(+ * 1) + *2 * 2 + \ldots * 2(+ * 1)$$

One summand $+*1$ appears if n is odd, the other appears if m is odd. Since $+$ is associative and commutative, and $*1 + *1 = 0$, at most one summand $+ * 1$ remains, and the sum is again an adder. This gives us the following rule for summing adders.

For N, M both odd, $:N + :M = :N + M - 2$
For N, M not both odd, $:N + :M = :(N + M)$

I will be making so much use of adders in this work that I will often omit the : when there is no possibility of ambiguity. However, to remind us that the sum for adders is not the same as the sum for the corresponding integers, I will write the sum with a : instead of a $+$, so that $:n + :m$ will be written as $n :m$. Since the adders are closed under addition, and clearly are generated by the adders **1** and **2**, we can write the set of adders as $\langle 1, 2 \rangle$.

That the set of adders is closed under descent is clear from the facts that $:2 = \{:0, :1\}$ and $:1 = \{:0\}$.

Let X be a set of games closed under addition and descent. Let s be simplest in $X \setminus \{0\}$. Since X is closed, all options of s are in $\{0\}$, so $s = 0$ or $s = 1$. Let t be simplest in $X \setminus \{0,1\}$, whence all options of t are in $\{0,1\}$, so t is one of $\{\} = 0$, $\{0\} = 1$, $\{1\} = 0$, $\{0,1\} = 2$, i.e., $t = 2$. Since X is closed under addition, we have $\langle 1, 2 \rangle \subset X$. ■

The simplicity argument forced our choice of which game to adjoin to the set $\{0\}$ to get $\{0,1\}$, and again which one to adjoin to get $\{0,1,2,3,\ldots\}$. This fails to determine uniquely which game should be adjoined to the adders to get another set with these properties, since several games (like $\{4,0\}$, $\{4,1\}$, $\{4,2,1\}$, etc.) can be admitted as simplest members of a larger closed set.

It now seems natural to use the three sets $\{0\}$, $\{0,1\}$, and $\{0,1,2,3,\ldots\}$ in the role of the set X above. For some game G, if we write down the outcome of $G + x$ for all $x \in \{0\}$, we have only written down the outcome of the game G. If we write the outcome of $G + x$ for all $x \in \{0,1\}$, we need to write two outcomes. However, since $1 = \{0\}$, we know that G is an option of $G + 1$, so we cannot have that both G and $G + 1$ are $\mathcal{P}$. One of three things is possible, either

i) the outcome of G is $\mathcal{P}$, and $G + :1$ is $\mathcal{N}$

ii) the outcome of $G + *1$ is $\mathcal{P}$, and G is $\mathcal{N}$

or iii) neither the outcome of G nor the outcome of $G + *1$ is $\mathcal{P}$.

For the case (i) we will write 0, for the case (ii) we will write 1, and for the case (iii) we will write the blurry symbol, #.

When we try to write down the outcome of $G + x$ for all $x \in \langle 1, 2\rangle$, we have an infinite number of outcomes to write down. At least we have a natural order in which to write them, that is, adding first 0, then 1, 2, etc. So, for example, since 0 is an $\mathcal{N}$ position, 1 is a $\mathcal{P}$ position, 2 is $\mathcal{N}$ etc., we could write down for 0 the sequence

NPNNPNNNPNNNP...
0 1 2 3 4 5 6 7 8 9 ...

This seems counterproductive, since we are using an infinitely long symbol to write down less information than we could extract from the line $0 = \{\}$! However, there is no need to write down the infinite string, since Conway has shown that this is an ultimately periodic sequence, which will repeat every four entries. That is, we could write the sequence for 1 as

NPNNPNNN

with the understanding that the last four values will repeat indefinitely.

Since we write down the outcomes for the sums $G + *2n$ and $G + *(2n + 1)$ next to each other, we can instead write one of the symbols 0, 1 and # as before for two of these possibilities. In this way the pairs **PN**, **NP** and **NN** become 0, 1 and # respectively (PP cannot occur), and the sequence for 0 becomes

1#0#

We shall call a sequence of this type a ***blurry genus sequence.*** We can interpret each location as describing the outcome of the games G, $G + *2$, $G + *2 + *2$, etc., writing a 1 in the n^{th} place if $G + *2n + *1$ is $\mathcal{P}$, a 0 if $G + *2n + *0$ is $\mathcal{P}$, and a # if neither case holds. We might also wish to replace #'s which precede **0's** by **2's** and #'s which precede **1's** by 3's, to indicate that $G + *2n + *2$ or $G + *2n + *3$ is $\mathcal{P}$ respectively. Then the above sequence becomes 1202. Notice that this form conveys no more information than the form with #'s and we shall regard them as identical. All computations presented in this work were therefore done with the blurry sequence, even though they may be printed with **2's** and 3's where appropriate. In **ONAG,** Conway distinguishes all possible occurrences of # in the same way, using other nimheaps. That is, the n^{th} digit of Conway's sequence is **k** if $G + *2n + *k$ is a $\mathcal{P}$ position. [*] I will call this form of the genus sequence the ***sharp*** genus sequence, since it makes a sharp distinction between all occurrences of #. I will seldom use the sharp sequence, and unless otherwise specified, any genus mentioned here is the blurry one.

[*] To use Conway's sequence in the computer would require us to allocate enough space for each entry to accommodate large nimheaps. Since we seldom need,to distinguish large nimheaps, I have chosen to economize and use the blurry genus as described in this work.

Because of the choice we have made for the set X, (i.e., $\langle 1,2\rangle$) which is closed both under addition and descent, we have a very simple algorithm for computing the genus sequence of G given the genus sequences for all the options G' of G. This algorithm uses the *mex* of a subset of $\{0,1,\#\}$. This mex is just the same as the normal mex defined above, except that all integers greater than 1 are blurred together, and are all written as #. We can write down the mex of all subsets of $\{0,1,\#\}$:

subset	$\{\}$	$\{0\}$	$\{1\}$	$\{0,1\}$	$\{\#\}$	$\{\#,0\}$	$\{\#,1\}$	$\{\#,1,0\}$
mex	0	1	0	#	0	1	0	3 #

If we refer to the entries in the genus sequence of a game G as $g_0(G)$, $g_1(G)$, etc., then the following rule tells us how to compute the genus of G given the genera of G' (From **ONAG**, chapter 12):

$g_0(0)=1$; if $G\neq 0$, then $g_0(G)$ is the mex of the $g_0(G')$
For all G, $g_{i+1}(G)=\mathrm{mex}\{g_{i+1}(G'), 0\overset{*}{+}g_i(G'), 1\overset{*}{+}g_i(G)\}$

These mex entries come from the computation of the outcome of

$$G:2n+:2=\{G':2n+:2, G+:2n+:1, G+:2n+:0\}$$

The outcome of the first option can be read from the genus sequence of G', the other two from the genus of G so far.

This algorithm can be performed easily with pencil and paper using the following format, which will be generalized later:

carries	$g_{i-1}\overset{*}{+}1$		0	#	#
	$g_{i-1}\overset{*}{+}0$		1	#	#
genus of 1		0	#	1	#
genus of :4		0	#	0	#
genus of $\{1,4\}$		1	#	#	0

This computes the genus of $\{1,4\}$ from the genera for 1 and 4; we write the genera as if we were doing ordinary sums, only we work from left to right, using mex instead of addition. We carry our result (and the result $\overset{*}{+}1$) to the next column on the right, to be used in that mex.

Reviewing the definition and computation of the genus sequence, we have that

1) G is a P position iff $g_0=0$
2) The genus of $G+:n$ can be trivially computed for any adder **:n**.
3) The genus of G can be computed without very much difficulty given the genus of all the options of G.
4) The genus sequence of any position is ultimately periodic of period 2.
(WW, chapter 13)

(1) follows immediately from the definition of the genus sequence, since from the start we have written the outcome of G first. (2) follows from the additive closure of the set $X=\langle 1,2\rangle$; in order to compute the genus of G :2, we merely shift the whole sequence

one to the left (since the i^{th} entry indicates the outcome information about $G:2i$, it also indicates outcome information about $G:2:2(i-1)$); to compute the genus of $G:1$, we nim-add **1** to each entry (if $G:0$ is P (i.e., $g_0(G)=0$), then $G:1:1$ is P ($g_0(G:1)=1$); if $G:1$ is P ($g_0(G)=1$) then $G:1:0$ is P ($g_0(G:1)=0$)). The algorithm for **(3)** was outlined above, and **(4)** follows easily from this algorithm.

During the rest of this work we will discuss two sorts of regularities which occur in the computations we make. Both sorts have a good claim to the name 'periodicity', so I would like to distinguish them clearly.

The first sort is the periodicity inside a genus sequence which we have already met in **(4)** above, i.e., we know that the genus sequence of a game repeats with period of **2** from some point on. I will spend most of this work discussing this sort of periodicity, and will refer to it as ***periodicity in a genus sequence.***

When we treat the positions which arise using a particular set of rules (say, those of Grundy's Game, q.v.), we often find that several different positions have some property **in** common, and that often this recurs at regular intervals (e.g., we may find that for some rules, heaps of size **10,** 20, **30,** etc. all have the same genus, as do heaps of size **11, 21, 31,** etc.). Since I will, on occasion, wish to refer to this phenomenon and the previously mentioned periodicity in the same discussion, I will refer to this phenomenon as ***recurrence of genus values.***

For the purpose of computing genus sequences, we are particularly interested in discovering when games are equal to adders, since it is these games which the genus sequence is designed to handle. **Also,** since games are often presented as sums of known games, we would like to be able to tell whether sums are equal to adders without working out all of their options. Happily, it turns out that by exploiting the structure of adders we find that the only sums which equal adders are sums of adders. To prove this, we need to know Conway's Cancellation theorem. The statement of the cancellation theorem appears in **ONAG,** but not the proof, so I have obtained his permission to include a proof here.

First we need some more definitions.

Let X and Y be equal games, $X=Y$, The properties of reversible moves tell us that for any option X' of X one of two things happens:

i) There exists a Y' for which $X'=Y'$
or ii) There exists a X'' for which $X''=Y$.

That is, either X' equals an option of Y, or Y equals an options of X'. With this in mind, we make the following definition:

A is ***adjacent*** to B, which we write as A @ B, means that either there exists A' with $A'=B$ or there exists B' with $B'=A$.

We have that $X=Y \Rightarrow X'$ @ Y, for any option X' of X.

If $X+Y=T$, we call X and Y ***parts*** of T.

We say that a game T is ***cancellable*** if for every pair of games G and H we have

i) $G+T=H+T \Rightarrow G=H$, and
ii) $G+T$ @ $H+T \Rightarrow G$ @ H

If (i) and (ii) hold for some particular games G and H, we will say that T is *cancellable* from G and H.

Before we can prove the cancellation theorem, we need some lemmas about parts and cancellability.

Lemma 1. 0 and 1 are the only parts of 0.

Proof. Let A be a simplest non-trivial part of 0, i.e., A is a part of 0, $A \neq 0, 1$, but every position of A is either 0, 1, or not a part of 0. Since A is a part of 0, it has a negative, so we have

$$A + (-A) = 0$$

Suppose A has an option A' not equal to 0 or 1. We have that

$$A' + (-A) @ 0$$

So either

$$A'' + (-A) = 0$$

or

$$A' + (-A)' = 0$$

From the former we can deduce that

$$A'' = A'' + (-A) + A = A$$

and hence A was not a simplest such position.

The latter leads to a contradiction since A' is neither 0 nor 1, and hence is not a part of 0.

Hence all of the options of A are either 0 or 1, so A is either 0, 1, or 2. Similarly, $-A$ is either 0, 1, or 2. Since $0 + 0 = 1 + 1 = 0$ and $2 + 2 \# 0$, 0 and 1 are the only parts of 0. ■

Notice that since $A + B = 1 \Rightarrow B + 1 = -A$, we can also say that 0 and 1 are the only parts of 1.

Lemma 2. If T is cancellable, so is any part of T.

Proof. For any part X of T, there is a Y such that $X + Y = T$.

i) $G + X = H + X \Rightarrow G + X + Y = H + X + Y \Rightarrow G + T = H + T \Rightarrow G = H$
ii) $G + X @ H + X \Rightarrow G + X + Y @ H + X + Y \Rightarrow G + T @ H + T \Rightarrow G @ H$

Theorem II (Conway's Cancellation).

For any game T,

i) T is cancellable, and
ii) T has finitely many parts (up to equality)

Proof. Let T be a simplest counterexample, i.e., T does not satisfy (i) and (ii), but any option of T does.

First we prove that T has only finitely many parts, or equivalently, that T has only finitely many partitions.

For any partition of T

$$X + Y = T$$

we have $X + Y$ @ T' for some option T' of T.

So either

a) $X' + Y = T'$
b) $X + Y' = T'$
or c) $X + Y = T''$

In case (a), Y takes only finitely many values, since it is a part of some T', and all such Y are cancellable (by Lemma 2), and so uniquely determine their corresponding X.

Similarly in case (b), X takes only finitely many values, and determines the corresponding Y.

Finally, case (c) implies that T was not a simplest counterexample.

Now we show that T is cancellable. If not, let G and H be a simplest pair of games from which T is not cancellable.

First suppose that

$$G + T \text{ @ } H + T$$

whence either

a) $G' + T = H + T$
b) $G + T' = H + T$
c) $G + T = H' + T$
or d) $G + T = H + T'$

In cases (a) and (c), since the pairs G', H and G, H' are simpler than G, H, we can cancel T and get that $G' = H$ or $H' = G$, that is, G @ H.

In cases (b), (d), we can deduce that $G + T'$ @ $H + T'$, so since T' is simpler than T, G @ H.

So $G + T$ @ $H + T \Rightarrow G$ @ H, in all cases.

Now suppose that $G + T = H + T$.

Then for any G' and option of G,

$$G' + T \text{ @ } H + T$$

and then

$$G' \text{ @ } H$$

Similarly, for any H', we have that H' @ G.

Our rules for reversible moves tell us that this implies that $G = H$, except in the case where one of G and H is 0. So in the remaining case, without loss of generality, we have

$$G + T = T$$

and we wish to show that $G = 0$.

We immediately deduce

$$G + T @ T'$$

so that either

a) $G' + T = T'$
b) $G + \tilde{T} = T'$ for $\tilde{T}$ an option of T (possibly distinct from T')
or c) $G + T = T''$

Case (a) tells us that T is a part of T', so by lemma 2, T is cancellable, and $G = 0$.

Case (b) tells us that G is a part of T', so G is cancellable. Now from $G + T = T$ we have that n .G is a part of T for all n. But T has finitely many parts, *so* n $.G = m \cdot G$ for some $m < n$, and since G is cancellable, (n – m) .G = 0. From lemma **1,** we have that G equals 0 or 1. Clearly $G \neq 1$, *so* $G = 0$, **as** desired.

Finally, case (c) implies that T was not simplest. ■

Corollary 1. If $A + B$ =:n, then A and B are both adders, say $:a$ and $:b$, and both a and $b \leq \max(n, n \overset{*}{+} 1)$.

Proof. Let $A + B$ =:n. If $n < 2$, then A and B are both 0 or 1. For larger n, :(n – **2)** is an option of $:n$, *so* without loss of generality, $A' + B \underset{*}{=}: (n - 2)$ for some A'. Now **by** induction B is an adder, say $:b$, $b \leq \max(n-2, n-2 \overset{*}{+} 1)$. Now $A + :b =: n$, and by Conway's Cancellation theorem, A is also an adder, say $:a$, and $a \leq \max(n, n \overset{*}{+} 1)$.

Now, if we are given a sum of games, and we wish to know whether they are adders, we know that all we have to do is to **look** at the summands. If they are adders, then so is the sum; if one of them is not, then neither is the sum.

THE GENUS SEQUENCE ALGORITHM

If we were to be given the tree of a perfectly arbitrary game, our genus techniques would probably not be of much use for determining its outcome. However, many games (Nim, for instance) have positions which are naturally expressed as sums, and rather than being given a game tree, we are given rules for expressing the options of one position **as** sums of the other positions. In such a case, it is quite useful to know the genus sequence of any fundamental positions which make up (as summands) all the possible positions. For Nim-like games (played with heaps of counters, where the rules specify allowable ways to remove counters and/or split up heaps) these are just the single-heap positions. For more complex games, it may be necessary to make up a dictionary of simple positions and their genera. Of course, since the genus sequence makes addition with adders easy, we should also note whether any games are equal to adders.

In the following algorithm I will represent adders by integers, and the operation on integers which corresponds to the sum on adders I will call the adder-sum.

I have experimented with several different sorts of algorithms, and have found the following 'nai've' algorithm to be the most effective: *

For each fundamental position, do the following:

Use the rules of the game to express the options of this fundamental position as sums of earlier fundamental positions. Given the genus of each of these options, compute the genus of this fundamental position.

In order to "compute the genus of a position, it suffices to compute recursively the genera of the options, and apply Conway's rule as demonstrated above

In order to compute the genus of a sum, first check to see if all the summands but one are adders. If so, adder-sum them together, and shift the genus of the remaining game by the amount specified by this adder-sum. Otherwise, use the rule $G + H = \{G' + H, G + H'\}$ for all options which are not adders (G' and H' are computed using the rules of the game). For future reference, record the wild (non-adder) summands alongside their sum.

Check to see if this fundamental position is an adder.

We know whether any earlier fundamental position is an adder, so by Corollary **1** we know which, if any, of the options of this fundamental position are adders. For any non-adder option, search its options for reversible moves, using Corollary **1** again.

This algorithm performs quite well in practice, and the results of using it to compute genera for the octal games (introduced by Guy and Smith [10], with some values tabulated

---

* The most sophisticated algorithm I tried stored each position as an ordered pair consisting **of** an adder and a more 'wild' position; each wild part pointed to the other positions, so that all options were held explicitly in the computer's memory. Whenever it was detected that two wild parts were equal, all pointers were re-organized so as to leave only one copy in the machine's memory. This scheme saved a substantial amount of time while working with games which would simplify; however, the storage bottleneck eventually dominated, and the nai've algorithm described in the text was able to perform better.

in WW chapter 13) can be found in appendix I. At the end of the next chapter are the genus sequences for Grundy's game. This game, in which a legal move is to split any heap into two heaps of different sizes, is an especially fine showcase for the genus sequence, since so many Grundy heaps are equal to adders. For this reason, I have chosen Grundy's game for a case study.

GRUNDY'S GAME

In table iii at the end of this chapter, I have included not only the genus sequences for the single heaps in Grundy's game, but also the genera of sums **of** two heaps (such **as** are needed to perform the single-heap calculations). Most **of** the information in the discussion to follow is condensed from table iii.

In the following, the letter **G** followed by an integer refers to a heap of the appropriate size in Grundy's game.

Some computational shortcuts in Grundy's Game

Conway (in ONAG, chapter **12)** has noted that in any sum of Grundy heaps smaller than **28,** any two heaps of size **13** can be neglected in the computation **of** the genus sequence of the sum. We can easily check to see if this trend continues for single larger Grundy heaps. I will illustrate the method for the **28** heap. We wish to know if **G28 + G13 + G13** has the same genus as **G28.** We could compute the genus of **G28** by computing the genus of all options of $G28$, which I will denote **as** $G28'$, and using Conway's rule. To compute the genus of **G28 + G13 + G13,** we need to know the genera of $G28'$ + **G13 + G13** and **G28 + G13' + G13.** Since the options of **G28** are just sums **of** smaller heaps, we know that the genus of $G28'$ + **G13 + G13** matches that of $G28'$. Since **G13** $= \{4,2,1\}$, this accounts for all the options of **G28 + G13 + G13** except except three, namely **G28 + G13 :4,** $G28+G13$ **:2,** and $G28+$ **G13 :1.** Look up the genus of $G28+$ **G13** in table iii; it is $2(2020)$. So we have:

```
                a b c d              a b c d
   options      e f g h              e f g h        options of
   of G28       i j k m              i j k m     G28+G13+G13
                  ...                  ...
                -------
                1 3 1 3              3 1 3 1      G28 + G13 : 1
                                     0 2 0 2      G28 + G13 : 2
                                     2 0 2 0      G28 + G13 : 4
                                   -----------
                                   ? 1 3 1 3?
```

The question mark indicates a line yet to be verified. This can be done by examining only the numbers shown; since we know that **1** is the mex of the first column of a,e,i,. . .and it does not appear in the first column of the printed numbers (3,0,2), it must be the mex of the entire first column, a,e,i,. . .3,0,2. **A** similar argument applies to the other columns. That is, all we need to do is to compute these three rows (given the genus of **G28 + G13,** this is easy), and check that they do not match the genus **of G28** in any place.

For the **29** heap, this argument as it stands fails, because we have not shown that **G13 + G13** can be neglected in any sum of Grundy heaps smaller than **29.** In fact, the only sums involving **G28** that we can deal with are of the form **G13 + G13 + G28 +** s, where s is an adder. Fortunately, the only option of **G29 + G13 + G13** which involves **G28** is just such a position, so the rule will still work. We check: genus of **G29** is (2020), **G29 + G13** is

(#313); does (#202), **(3131)** or **(1313)** match **(2020)** in anyplace? **No;** so $G13+G13+G29$ has the same genus as **G29.**

This sort of check continues to be relevant (and to work) up to **G41. G41** has an option **G28 + G13,** but we have not checked that **G28 + G13 + G13 + G13** has the same genus as $G28+G13$. We can check this by the same method, that is, check that $G28+G13+G13$ **:1, G28 + G13 + G13 :2,** and **G28 + G13 + G13 :4 do** not match **G28 + G13.** This is easy to do, since we know the genus of **G28 + G13 + G13.** Indeed this is the case, so we have justified our check for **G41.** From now on these checks will be this tedious, so we will leave them to the digital computer.

These observations are helpful when we are playing Grundy's game, since they allow us to ignore pairs of heaps in some common circumstances. Later we will see how a generalization of this method will allow us to ignore certain sums on even more occasions.

Two-heap P positions

We would like to be able to use the data we have from the computer to tell us the outcomes of heaps larger than **88** (the largest heap the computer treats). Toward this end we examine the options of single heaps, i.e., the positions with two heaps. We are particularly interested in those which are P positions.

We can arrange the two-heap P positions of Grundy's game in such a way that they are easy to write in a small space. Omitting sums where one heap equals an adder (i.e., heaps of sizes **17, 15, 14, 12** or less), every two-heap P position in Grundy's game involving less than **88** counters is in one of the following sets (for brevity in this list I will write n for a heap of n counters):

$$\{39,42,45\}+\{39,42,45\}$$
$$\{30,33,36\}+\{30,33,36\}$$
$$\{13,16,19,22,25\}+\{35,38,41,44,\ldots 68,71,74,\ldots?\}$$

$$\{18,21,24,27\}+\{55,58,61,64\}$$
$$\{20,23,26,29,32\}+\{20,23,26,29,32\}$$
$$\{41\}+\{38,41,44,\ldots?\}$$
$$\{13,16,19,22,25\}+\{18,21,24,27\}$$

$$\{29,32\}+\{48,51,54\}$$
$$\{28,31,34,37,\ldots 49,52\}+\{28,31,34,37,\ldots 49,52\}$$

table i. Possible two-heap P positions in Grundy's game which involve no adders. Each of these with fewer than 88 counters is known to be a $\mathcal{P}$ position.

The sets of games in this list must not be confused with sets of options of a game, so the sum $A+B$ refers to the set $\{a+b \mid a\in A, b\in B\}$.

Not only is it true that every two-heap $\mathcal{P}$ position (involving no adder) is mentioned in this list, but also, every sum in this list whose outcome is known is a $\mathcal{P}$ position. We also see that the elements of each set form an arithmetic sequence, with an increment of three. In most cases, we have enough data to show that these sets are maximal with these properties. Let us examine the sum $\{18,21,24,27\} + \{55,58,61,64\}$ as an example of this maximality. Any of the sets

$$\begin{aligned} &\{15,18,21,24,27\} + \{55,58,61,64\} \\ &\{18,21,24,27,30\} + \{55,58,61,64\} \\ &\{18,21,24,27\} + \{52,55,58,61,64\} \\ \text{and}\quad &\{18,21,24,27\} + \{55,58,61,64,67\} \end{aligned}$$

satisfies the .arithmetic sequence property. We eliminate the first case, since **G15** is an adder. In the other three sets we can find sums **(30 + 55, 18 + 52,** and **18 + 67** respectively) which we know are $\mathcal{N}$ positions. The two queried ellipses are the only cases where continuation may be possible.

In the next section we use these regularities to predict Ppositions with **88** or more counters. This greatly simplifies the task of finding outcomes of single large heaps.

Outcomes **of** large **Grundy** heaps

Since many small Grundy heaps equal adders, we can easily compute certain sums with them. From table iii, we find that the g_0 value of **G79** is **2;** since Grundy heaps of sizes **11, 14** and **17** equal **2,** we have that the heaps of sizes **90, 93** and **96** are all Upositions. A Grundy heap of **84** counters has go-value **1; G15** equals $*$**1,** ***so*** we have that the heap of **99** counters is an Nposition. **G87** has go-value **2,** and heaps of sizes **5,8,11,14** and **17** equal $*$**2,** so heaps of sizes **92, 95, 98, 101** and **104** are all Npositions. **A** Grundy heap of size **75** has g_0 value **2,** and the heap of size **14** equals ***2,** so the heap of **89** coins is also an $\mathcal{N}$ position.

From the last section, we predict that the sum **G27 + G64** is a *P* position. By hand I have verified this, and hence we conclude that **G92** is an $\mathcal{N}$ position.

The only line of table i that might contain a P position with **94** counters is $\{G41\} + \{G38,$ **G41,**$G44,\ldots\}$, which leads to $G41+G53$ as the only hopeful candidate for a *P* option of **G94.** However, from table iii we find that **G26 + G15 + G53** = **G26 +** $*$**1 + G53,** and again from table iii we find that this is a *P* position, whence **G41 + G53** is an $\mathcal{N}$ position. We now suspect that since there are no obvious guesses for a P option of **G94,** that **G94** may itself be a Pposition. In table ii, for each option $G94'$ of **G94** I find an option $G94''$ of $G94'$ which is a P position, verifying that **G94** is indeed a P position.

As a supplement to table iii, I show the outcomes of single heap positions in Grundy's game for some heaps larger than **88.**

89	**90**	**91**	**92**	**93**	**94**	**95**	**96**	**97**	**98**	**99**	**100**	**101**	**102**	**103**	**104**
N	N	N	N	N	P	N	N	?	N	N	?	N	?	?	N

Thus the *n* for which we know that *Gn* is a Pposition are **3,6,9,12,. . . ,42,45,50** and **94.**

1 + 93 - 79 + 14
2 + 92 - 87 + 5
3 + 91 - 3 + 84 + 7
4 + 90 - 4 + 79 + 11
5 + 89 - 5 + 79 + 10
6 + 88 - 6 + 84 + 4
7 + 87 - 7 + 79 + 8
8 + 86 - 8 + 79 + 7
9 + 85 - 9 + 84 + 1
10 + 84 - 10 + 79 + 5
11 + 83 - 11 + 79 + 7
12 + 82 - 12 + 81 + 1
13 + 81 - 6 + 7 + 81
14 + 80 - 14 + 79 + 1
15 + 79 - 15 + 78 + 1
16 + 78 - 7 + 9 + 78
17 + 77 - 17 + 70 + 1
18 + 76 - 7 + 11 + 76
19 + 75 - 10 + 9 + 75
20 + 74 - 11 + 9 + 74
21 + 73 - 10 + 11 + 73
22 + 72 - 10 + 12 + 72
23 + 71 - 11 + 12 + 71
24 + 70 - 14 + 10 + 70
25 + 69 - 10 + 15 + 69
26 + 68 - 12 + 14 + 68
27 + 67 - 10 + 17 + 67
28 + 66 - 28 + 55 + 11
29 + 65 - 15 + 14 + 65
30 + 64 - 30 + 26 + 38
31 + 63 - 31 + 8 + 55
32 + 62 - 15 + 17 + 62
33 + 61 - 33 + 26 + 35
34 + 60 - 19 + 15 + 60
35 + 59 - 26 + 9 + 59
36 + 58 - 28 + 8 + 58
37 + 57 - 22 + 15 + 57
38 + 56 - 26 + 12 + 56
39 + 55 - 28 + 11 + 55
40 + 54 - 25 + 15 + 54
41 + 53 - 26 + 15 + 53
42 + 52 - 25 + 17 + 52
43 + 51 - 43 + 15 + 36
44 + 50 - 29 + 15 + 50
45 + 49 - 45 + 42 + 7
46 + 48 - 46 + 29 + 19

table ii

Proof that **G94** is **a P** position. Most of the positions in **the** right-hand column **can** be **looked up in table iii; those that do not appear there were worked out** by hand.

```
grundy(  0)= *0(1202)
grundy(  1)- *0(1202)
grundy(  2)= *0(1202)
grundy(  3)= *1(0313)
grundy(  4)= *0(1202)
grundy(  5)= *2(2020)
grundy(  6)= *1(0313)
grundy(  7)- *0(1202)
grundy(  8)= *2(2020)
grundy(  9)= *1(0313)
grundy( 10)= *0(1202)
grundy( 11)= *2(2020)
grundy( 12)= *1(0313)
grundy( 13)=  3(1#31)
grundy( 14)= *2(2020)
grundy( 15)= *1(0313)
grundy( 16)=  3(1#31)
grundy( 17)= *2(2020)
grundy( 18)=  4(0###)
grundy( 19)=  3(1#31)
grundy( 20)=  0(2020)
grundy( 21)=  4(0###)
grundy( 22)=  3(1#31)
grundy( 23)=  0(2020)
grundy( 24)=  4(0###)
grundy( 25)=  3(1#31)
grundy( 26)=  0(2020)
grundy( 27)=  4(0###)
grundy( 28)=  1(1313)
13 + 16 = (1202)
grundy( 29)=  2(2020)
grundy( 30)-  3(0202)
13 + 18 = (0###)
grundy( 31)=  1(1313)
13 + 19 = (1202)
grundy( 32)=  2(2020)
13 + 20 = (#313)
grundy( 33)=  4(0202)
13 + 21   (0###)
16 + 18 = (0###)
grundy( 34)=  1(1313)
13 + 22 = (1202)
16 + 19 = (1202)
grundy( 35)=  2(####)
13 + 23 = (#313)
16 + 20 = (#313)
grundy( 36)=  4(0202)
13 + 24 = (0###)
16 + 21 = (0###)
18 + 19 = (0###)
grundy( 37)=  1(1313)
13 + 25 = (1202)
16 + 22 = (1202)
18 + 20 = (####)
grundy( 38)=  2(####)
13 + 26 = (#313)
16 + 23 = (#313)
18 + 21 = (1202)
19 + 20 = (#313)
grundy( 39)=  4(0###)
13 + 27 = (0###)
16 + 24 = (0###)
18 + 22 = (0###)
19 + 21 = (0###)
grundy( 40)=  1(1313)
13 + 28 = (2020)
16 + 25 = (1202)
18 + 23 = (####)
19 + 22 = (1202)
20 + 21 = (####)
grundy( 41)=  5(####)
13 + 29 = (#313)
16 + 26 = (#313)
18 + 24 = (1202)
19 + 23 = (#313)
20 + 22 = (#313)
grundy( 42)=  4(0###)
13 + 30 = (3131)
16 + 27 = (0###)
18 + 25 = (0###)
19 + 24 = (0###)
20 + 23 = (0202)
21 + 22 = (0###)
grundy( 43)-  1(1313)
13 + 31 = (2020)
16 + 28 = (2020)
18 + 26 = (####)
19 + 25 = (1202)
20 + 24 = (####)
21 + 23 = (####)
grundy( 44)=  5(####)
13 + 32 = (#313)
16 + 29 = (#313)
18 + 27 = (1202)
19 + 26 = (#313)
20 + 25 = (#313)
21 + 24 = (1202)
22 + 23 = (#313)
grundy( 45)=  4(0#20)
13 + 33 = (3131)
16 + 30 = (3131)
18 + 28 = (####)
19 + 27 = (0###)
20 + 26 = (0202)
21 + 25 = (0###)
22 + 24 = (0###)
grundy( 46)=  1(1313)
13 + 34 = (2020)
16 + 31 = (2020)
18 + 29 = (####)
19 + 28 = (2020)
20 + 27 = (####)
21 + 26 = (####)
22 + 25 = (1202)
23 + 24 = (####)
grundy( 47)=  5(####)
13 + 35 = (0###)
16 + 32 = (#313)
18 + 30 = (####)
19 + 29 = (#313)
20 + 28 = (1313)
21 + 27 = (1202)
22 + 26 = (#313)
23 + 25 = (#313)
grundy( 48)=  4(2020)
13 + 36 = (3131)
16 + 33 = (3131)
18 + 31 = (####)
19 + 30 = (3131)
20 + 29 = (0202)
21 + 28 = (####)
22 + 27 = (0###)
23 + 26 = (0202)
24 + 25 = (0###)
grundy( 49)=  1(1313)
13 + 37 = (2020)
16 + 34 = (2020)
18 + 32 = (1###)
19 + 31 = (2020)
20 + 30 = (2020)
21 + 29 = (####)
22 + 28 = (2020)
23 + 27 = (####)
24 + 26 = (####)
grundy( 50)=  0(0#31)
13 + 38 = (0###)
16 + 35 = (0###)
18 + 33 = (####)
19 + 32 = (#313)
20 + 31 = (1313)
```

table iii

```
21 + 30 = (####)
22 + 29 = (#313)
23 + 28 = (1313)
24 + 27 = (1202)
25 + 26 = (#313)
grundy( 51)= 2(2020)
13 + 39 = (31##)
16 + 36 = (3131)
18 + 34 = (####)
19 + 33 = (3131)
20 + 32 = (0202)
21 + 31 = (####)
22 + 30 = (3131)
23 + 29 = (0202)
24 + 28 = (####)
25 + 27 = (0###)
grundy( 52)- 1(1313)
13 + 40 = (2020)
16 + 37 = (2020)
18 + 35 = (1202)
19 + 34 = (2020)
20 + 33 = (2020)
21 + 32 = (1###)
22 + 31 = (2020)
23 + 30 = (2020)
24 + 29 = (####)
25 + 28 = (2020)
26 + 27 = (####)
grundy( 53)= 0(3131)
13 + 41 = (0###)
16 + 38 = (0###)
18 + 36 = (####)
19 + 35 = (0###)
20 + 34 = (1313)
21 + 33 = (####)
22 + 32 = (#313)
23 + 31 = (1313)
24 + 30 = (####)
25 + 29 = (#313)
26 + 28 = (1313)
grundy( 54)= 2(2020)
13 + 42 = (31##)
16 + 39 = (31##)
18 + 37 = (####)
19 + 36 = (3131)
20 + 35 = (3131)
21 + 34 = (####)
22 + 33 = (3131)
23 + 32 = (0202)
24 + 31 = (####)
```

```
25 + 30 = (3131)
26 + 29 = (0202)
27 + 28 = (####)
grundy( 55)- 1(1###)
13 + 43 = (2020)
16 + 40 = (2020)
18 + 38 = (1202)
19 + 37 = (2020)
20 + 36 = (2020)
21 + 35 = (1202)
22 + 34 = (2020)
23 + 33 = (2020)
24 + 32 = (1###)
25 + 31 = (2020)
26 + 30 = (2020)
27 + 29 = (####)
grundy( 56)= 5(3131)
13 + 44 = (0###)
16 + 41 = (0###)
18 + 39 = (2020)
19 + 38 = (0###)
20 + 37 = (1313)
21 + 36 = (####)
22 + 35 = (0###)
23 + 34 = (1313)
24 + 33 = (####)
25 + 32 = (#313)
26 + 31 = (1313)
27 + 30 = (####)
28 + 29 = (3131)
grundy( 57)= 2(#202)
13 + 45 = (31#3)
16 + 42 = (31##)
18 + 40 = (####)
19 + 39 = (31##)
20 + 38 = (3131)
21 + 37 = (####)
22 + 36 = (3131)
23 + 35 = (3131)
24 + 34 = (####)
25 + 33 = (3131)
26 + 32 = (0202)
27 + 31 = (####)
28 + 30 = (####)
grundy( 58)- 1(####)
13 + 46 = (2020)
16 + 43 = (2020)
18 + 41 = (1202)
19 + 40 = (2020)
20 + 39 = (####)
```

```
21 + 38 = (1202)
22 + 37 = (2020)
23 + 36 = (2020)
24 + 35 = (1202)
25 + 34 = (2020)
26 + 33 = (2020)
27 + 32 = (1###)
28 + 31 = (0202)
29 + 30 = (1313)
grundy( 59)= 3(3131)
13 + 47 = (0###)
16 + 44 = (0###)
18 + 42 = (2020)
19 + 41 = (0###)
20 + 40 = (####)
21 + 39 = (2020)
22 + 38 = (0###)
23 + 37 = (1313)
24 + 36 = (####)
25 + 35 = (0###)
26 + 34 = (1313)
27 + 33 = (####)
28 + 32 = (3131)
29 + 31 = (3131)
grundy( 60)= 2(#202)
13 + 48 = (#313)
16 + 45 = (31#3)
18 + 43 = (####)
19 + 42 = (31##)
20 + 41 = (3131)
21 + 40 = (####)
22 + 39 = (31##)
23 + 38 = (3131)
24 + 37 = (####)
25 + 36 = (3131)
26 + 35 = (3131)
27 + 34 = (####)
28 + 33 = (####)
29 + 32 = (0202)
30 + 31 = (####)
grundy( 61)= 1(####)
13 + 49 = (2020)
16 + 46 = (2020)
18 + 44 = (1#31)
19 + 43 = (2020)
20 + 42 = (1313)
21 + 41 = (1202)
22 + 40 = (2020)
23 + 39 = (####)
24 + 38 = (1202)
```

```
25 + 37 = (2020)
26 + 36 = (2020)
27 + 35 = (1202)
28 + 34 = (0202)
29 + 33 = (2020)
30 + 32 = (1313)
grundy( 62)= 3(31##)
13 + 50 = (0202)
16 + 47 = (0###)
18 + 45 = (20##)
19 + 44 = (0###)
20 + 43 = (####)
21 + 42 = (2020)
22 + 41 = (0###)
23 + 40 = (####)
24 + 39 = (2020)
25 + 38 = (0###)
26 + 37 = (1313)
27 + 36 = (####)
28 + 35 = (####)
29 + 34 = (3131)
30 + 33 = (0202)
31 + 32 = (3131)
grundy( 63)= 2(####)
13 + 51 = (#313)
16 + 48 = (#313)
18 + 46 = (####)
19 + 45 = (31#3)
20 + 44 = (31##)
21 + 43 = (####)
22 + 42 = (31##)
23 + 41 = (3131)
24 + 40 = (####)
25 + 39 = (31##)
26 + 38 = (3131)
27 + 37 = (####)
28 + 36 = (####)
29 + 35 = (####)
30 + 34 = (####)
31 + 33 = (####)
grundy( 64)= 4(####)
13 + 52 = (2020)
16 + 49 = (2020)
18 + 47 = (1#31)
19 + 46 = (2020)
20 + 45 = (1###)
21 + 44 = (1#31)
22 + 43 = (2020)
23 + 42 = (1313)
24 + 41 = (1202)
25 + 40 = (2020)
26 + 39 = (####)
27 + 38 = (1202)
28 + 37 = (0202)
29 + 36 = (2020)
30 + 35 = (1###)
31 + 34 = (0202)
32 + 33 = (2020)
grundy( 65)= 3(31##)
13 + 53 = (0202)
16 + 50 = (0202)
18 + 48 = (####)
19 + 47 = (0###)
20 + 46 = (##31)
21 + 45 = (20##)
22 + 44 = (0###)
23 + 43 = (####)
24 + 42 = (2020)
25 + 41 = (0###)
26 + 40 = (####)
27 + 39 = (2020)
28 + 38 = (####)
29 + 37 = (3131)
30 + 36 = (0202)
31 + 35 = (31##)
32 + 34 = (3131)
grundy( 66)= 2(1###)
13 + 54 = (1313)
16 + 51 = (#313)
18 + 49 = (####)
19 + 48 = (#313)
20 + 47 = (####)
21 + 46 = (####)
22 + 45 = (31#3)
23 + 44 = (31##)
24 + 43 = (####)
25 + 42 = (####)
26 + 41 = (3131)
27 + 40 = (####)
28 + 39 = (####)
29 + 38 = (####)
30 + 37 = (####)
31 + 36 = (####)
32 + 35 = (#202)
33 + 34 = (####)
grundy( 67)= 4(2020)
13 + 55 = (####)
16 + 52 = (2020)
18 + 50 = (1###)
19 + 49 = (2020)
20 + 48 = (####)
21 + 47 = (1#31)
22 + 46 = (2020)
23 + 45 = (1###)
24 + 44 = (1#31)
25 + 43 = (2020)
26 + 42 = (1313)
27 + 41 = (1202)
28 + 40 = (0202)
29 + 39 = (####)
30 + 38 = (1###)
31 + 37 = (0202)
32 + 36 = (2020)
33 + 35 = (1###)
grundy( 68)= 3(31##)
13 + 56 = (0202)
16 + 53 = (0202)
18 + 51 = (####)
19 + 50 = (0202)
20 + 49 = (3131)
21 + 48 = (####)
22 + 47 = (0###)
23 + 46 = (##31)
24 + 45 = (20##)
25 + 44 = (0###)
26 + 43 = (####)
27 + 42 = (2020)
28 + 41 = (####)
29 + 40 = (3131)
30 + 39 = (2020)
31 + 38 = (31##)
32 + 37 = (3131)
33 + 36 = (0202)
34 + 35 = (31##)
grundy( 69)= 2(1###)
13 + 57 = (1#31)
16 + 54 = (1313)
18 + 52 = (####)
19 + 51 = (#313)
20 + 50 = (#313)
21 + 49 = (####)
22 + 48 = (#313)
23 + 47 = (####)
24 + 46 = (####)
25 + 45 = (31#3)
26 + 44 = (31##)
27 + 43 = (####)
28 + 42 = (####)
29 + 41 = (####)
30 + 40 = (#313)
```

```
31 + 39 = (####)
32 + 38 = (####)
33 + 37 = (####)
34 + 36 = (####)
grundy( 70)- 4(2020)
13 + 58 = (####)
16 + 55 = (####)
18 + 53 = (####)
19 + 52 = (2020)
20 + 51 = (####)
21 + 50 = (1###)
22 + 49 = (2020)
23 + 48 = (####)
24 + 47 = (1#31)
25 + 46 = (2020)
26 + 45 = (1###)
27 + 44 = (1#31)
28 + 43 = (0202)
29 + 42 = (####)
30 + 41 = (1###)
31 + 40 = (0202)
32 + 39 = (####)
33 + 38 = (1###)
34 + 37 = (0202)
35 + 36 = (####)
grundy( 71)- 3(31#3)
13 + 59 = (0202)
16 + 56 = (0202)
18 + 54 = (####)
19 + 53 = (0202)
20 + 52 = (3131)
21 + 51 = (####)
22 + 50 = (0202)
23 + 49 = (3131)
24 + 48 = (####)
25 + 47 = (0###)
26 + 46 = (##31)
27 + 45 = (20##)
28 + 44 = (####)
29 + 43 = (3131)
30 + 42 = (####)
31 + 41 = (####)
32 + 40 = (3131)
33 + 39 = (####)
34 + 38 = (31##)
35 + 37 = (31##)
grundy( 72)= 2(1###)
13 + 60 = (1#31)
16 + 57 = (1#31)
18 + 55 = (0202)
19 + 54 = (1313)
20 + 53 = (1313)
21 + 52 = (####)
22 + 51 = (#313)
23 + 50 = (#313)
24 + 49 = (####)
25 + 48 = (#313)
26 + 47 = (####)
27 + 46 = (####)
28 + 45 = (3131)
29 + 44 = (####)
30 + 43 = (#313)
31 + 42 = (####)
32 + 41 = (####)
33 + 40 = (#313)
34 + 39 = (####)
35 + 38 = (#202)
36 + 37 = (####)
grundy( 73)= 4(2020)
13 + 61 = (####)
16 + 58 = (####)
18 + 56 = (####)
19 + 55 = (####)
20 + 54 = (2020)
21 + 53 = (####)
22 + 52 = (2020)
23 + 51 = (####)
24 + 50 = (1###)
25 + 49 = (2020)
26 + 48 = (####)
27 + 47 = (1#31)
28 + 46 = (0202)
29 + 45 = (####)
30 + 44 = (1#31)
31 + 43 = (0202)
32 + 42 = (####)
33 + 41 = (1###)
34 + 40 = (0202)
35 + 39 = (2020)
36 + 38 = (####)
grundy( 74)= 3(31#3)
13 + 62 = (0###)
16 + 59 = (0202)
18 + 57 = (####)
19 + 56 = (0202)
20 + 55 = (####)
21 + 54 = (####)
22 + 53 = (0202)
23 + 52 = (3131)
24 + 51 = (####)
25 + 50 = (0202)
26 + 49 = (3131)
27 + 48 = (####)
28 + 47 = (####)
29 + 46 = (3131)
30 + 45 = (####)
31 + 44 = (####)
32 + 43 = (3131)
33 + 42 = (####)
34 + 41 = (####)
35 + 40 = (####)
36 + 39 = (####)
37 + 38 = (31##)
grundy( 75)= 2(1###)
13 + 63 = (1###)
16 + 60 = (1#31)
18 + 58 = (0202)
19 + 57 = (1#31)
20 + 56 = (1313)
21 + 55 = (0202)
22 + 54 = (1313)
23 + 53 = (1313)
24 + 52 = (####)
25 + 51 = (#313)
26 + 50 = (#313)
27 + 49 = (####)
28 + 48 = (3131)
29 + 47 = (####)
30 + 46 = (####)
31 + 45 = (##31)
32 + 44 = (####)
33 + 43 = (#313)
34 + 42 = (####)
35 + 41 = (#202)
36 + 40 = (1313)
37 + 39 = (####)
grundy( 76)- 4(2020)
13 + 64 = (####)
16 + 61 = (####)
18 + 59 = (####)
19 + 58 = (####)
20 + 57 = (2020)
21 + 56 = (####)
22 + 55 = (####)
23 + 54 = (2020)
24 + 53 = (####)
25 + 52 = (2020)
26 + 51 = (####)
27 + 50 = (31##)
28 + 49 = (0202)
```

```
29 + 48 = (0202)
30 + 47 = (3131)
31 + 46 = (0202)
32 + 45 = (####)
33 + 44 = (1###)
34 + 43 = (0202)
35 + 42 = (2020)
36 + 41 = (####)
37 + 40 = (0202)
38 + 39 = (2020)
grundy( 77)-  3(#313)
13 + 65 = (0###)
16 + 62 = (0###)
18 + 60 = (####)
19 + 59 = (0202)
20 + 58 = (####)
21 + 57 = (####)
22 + 56 = (0202)
23 + 55 = (####)
24 + 54 = (####)
25 + 53 = (0202)
26 + 52 = (3131)
27 + 51 = (####)
28 + 50 = (2020)
29 + 49 = (3131)
30 + 48 = (####)
31 + 47 = (####)
32 + 46 = (3131)
33 + 45 = (2020)
34 + 44 = (####)
35 + 43 = (####)
36 + 42 = (####)
37 + 41 = (####)
38 + 40 = (####)
grundy( 78)=  2(1###)
13 + 66 = (1###)
16 + 63 = (1###)
18 + 61 = (0202)
19 + 60 = (1#31)
20 + 59 = (1313)
21 + 58 = (0202)
22 + 57 = (1#31)
23 + 56 = (1313)
24 + 55 = (0202)
25 + 54 = (1313)
26 + 53 = (1313)
27 + 52 = (####)
28 + 51 = (3131)
29 + 50 = (1313)
30 + 49 = (####)

31 + 48 = (##31)
32 + 47 = (####)
33 + 46 = (####)
34 + 45 = (##31)
35 + 44 = (##31)
36 + 43 = (1313)
37 + 42 = (####)
38 + 41 = (0202)
39 + 40 = (####)
grundy( 79)=  4(2020)
13 + 67 = (#313)
16 + 64 = (####)
18 + 62 = (2031)
19 + 61 = (####)
20 + 60 = (2020)
21 + 59 = (####)
22 + 58 = (####)
23 + 57 = (2020)
24 + 56 = (####)
25 + 55 = (####)
26 + 54 = (2020)
27 + 53 = (####)
28 + 52 = (0202)
29 + 51 = (0202)
30 + 50 = (3131)
31 + 49 = (0202)
32 + 48 = (0202)
33 + 47 = (1###)
34 + 46 = (0202)
35 + 45 = (#31#)
36 + 44 = (####)
37 + 43 = (0202)
38 + 42 = (2020)
39 + 41 = (1#20)
grundy( 80)=  5(####)
13 + 68 = (0###)
16 + 65 = (0###)
18 + 63 = (31#3)
19 + 62 = (0###)
20 + 61 = (####)
21 + 60 = (####)
22 + 59 = (0202)
23 + 58 = (####)
24 + 57 = (####)
25 + 56 = (0202)
26 + 55 = (####)
27 + 54 = (####)
28 + 53 = (##20)
29 + 52 = (3131)
30 + 51 = (####)

31 + 50 = (##20)
32 + 49 = (3131)
33 + 48 = (##20)
34 + 47 = (####)
35 + 46 = (31##)
36 + 45 = (2020)
37 + 44 = (####)
38 + 43 = (####)
39 + 42 = (0202)
40 + 41 = (####)
grundy( 81)=  2(1###)
13 + 69 = (1###)
16 + 66 = (1###)
18 + 64 = (0202)
19 + 63 = (1###)
20 + 62 = (1###)
21 + 61 = (0202)
22 + 60 = (1#31)
23 + 59 = (1313)
24 + 58 = (0202)
25 + 57 = (1#31)
26 + 56 = (1313)
27 + 55 = (0202)
28 + 54 = (3131)
29 + 53 = (1###)
30 + 52 = (2020)
31 + 51 = (3131)
32 + 50 = (####)
33 + 49 = (####)
34 + 48 = (##31)
35 + 47 = (2031)
36 + 46 = (####)
37 + 45 = (##31)
38 + 44 = (##31)
39 + 43 = (####)
40 + 42 = (####)
grundy( 82)-  4(####)
13 + 70 = (#313)
16 + 67 = (#313)
18 + 65 = (20##)
19 + 64 = (####)
20 + 63 = (####)
21 + 62 = (2031)
22 + 61 = (####)
23 + 60 = (2020)
24 + 59 = (####)
25 + 58 = (####)
26 + 57 = (2020)
27 + 56 = (####)
28 + 55 = (20##)
```

```
29 + 54 = (0202)        25 + 60 = (1#31)
30 + 53 = (####)
31 + 52 = (0202)
32 + 51 = (0202)
33 + 50 = (3131)
34 + 49 = (0202)
35 + 48 = (####)
36 + 47 = (####)
37 + 46 = (0202)
38 + 45 = (#313)
39 + 44 = (1313)
40 + 43 = (0202)
41 + 42 = (1#20)
grundy( 83 )=  5(####)
13 + 71 = (0#20)
16 + 68 = (0###)
18 + 66 = (31#3)
19 + 65 = (0###)
20 + 64 = (#313)
21 + 63 = (31#3)
22 + 62 = (0###)
23 + 61 = (####)
24 + 60 = (####)
25 + 59 = (0202)
26 + 58 = (####)
27 + 57 = (####)
28 + 56 = (##20)
29 + 55 = (####)
30 + 54 = (#313)
31 + 53 = (##20)
32 + 52 = (3131)
33 + 51 = (##20)
34 + 50 = (##20)
35 + 49 = (31##)
36 + 48 = (##20)
37 + 47 = (####)
38 + 46 = (31##)
39 + 45 = (0202)
40 + 44 = (####)
41 + 43 = (####)
grundy( 84 )=  2(1###)
grundy( 85 )=  4(####)
grundy( 86 )=  3(####)
grundy( 87 )=  7(20##)
grundy( 88 )=  4(#313)
```

OTHER GAMES

The computer's calculations have demonstrated regularities in some other games. **For** the most part, these regularities are not dramatic enough to work into a complete analysis, and in many cases they do not even continue as far we have data. Some of these games have been treated by Yamasaki [**13**] and Ferguson [**7**].

Tins

The game of Tins is played with rows of tin cans. The legal move is to remove Two adjacent cans from the INterior of a row (splitting that row in two). In the octal notation of Guy and Smith [**10**], Tins is the game **.04.**

The computer's calculations show a remarkable regularity which is not mentioned in WW, and which is not reflected by the normal play result. In order to make this regularity easier to follow, table iv shows the *go* values of the heaps from size **1** to 60 in the game .04:

```
 |     |
 |000221   A quarter per
 |     |
0/1   2/3
```

right one swapping 2's
this $5\frac{1}{2}$-figure string is

```
 |     |     |     |     |     |     |
 |00022133000|11133022 111|00022133000| ...
 |     |     |     |     |     |     |
0/1   2/3   1/0   3/
```

Guiles

In the game which is called Guiles in WW (octal code .15; rules as in .04, except that a row of length 1 or 2 may also be removed, analyzed extensively by Guy and Smith [10]), Berlekamp, Conway and Guy have observed that many of the different heaps are equal. It is not surprising to find that the computer's calculations have shown a strong tendency towards repetition of the genus values of the various heaps. In fact, we find that the genus sequences very nearly repeat at intervals of 10 starting from the row of **9** pins, in that the genus sequence of the row if n pins and the row of $n + 10$ pins always match, except possibly for the values g_0 and g_1. These discrepancies seem to damp out, and we find that the genus sequences of heaps of 22 or more counters (**as** far as we have data) match up exactly in intervals of 10, except that the heaps of sizes **47, 49** and 50 do not match the heaps of size of sizes 57, 59 and **60.** Perhaps these and other regularities are best illustrated by the genus sequences themselves, as shown in the following table.

n	1	2	3	4	5	6	7	8	9	10
$g_0(n+0)$	031	031	120	031	031	202	202	031	202	202
$g_0(n+10)$	031	031	120	#31	#31	202	202	202	202	1#2
$g_0(n+20)$	#31	#31	120	#31	#31	202	1#2	202	1#2	1#2
$g_0(n+30)$	131	#31	120	#31	#31	202	1#2	202	1#2	1#2
$g_0(n+40)$	131	#31	120	#31	#31	202	1#2	202	1#2	1#2
$g_0(n+50)$	131	#31	120	#31	#31	202	202	202	202	202
$g_0(n+60)$	131	#31	120	#31						

The first three digits of genus sequences for Guiles heaps

The Octal Game .17

In the octal game .17, (in which the legal move is to remove any two adjacent counters from a row, or to remove any row of exactly one counter) we find that the g_0 values show a tendency to recur at intervals of 7 (this is available from the computations done in WW). **Here** the computer's calculations show us that this pattern does not continue, since it breaks down at 27 (and every seventh game thereafter, as far as we have data) and again at **44.** The fact that the recurrence occurs only in the g_0 values makes it unsurprising that this trend is unstable.

n	0	1	2	3	4	5	6
$g_0(n_0)$	1	0	0	1	2	0	3
$g_0(n_7)$	1	0	0	1	2	0	3
$g_0(n_{14})$	1	0	0	1	2	0	3
$g_0(n_{21})$	1	0	0	1	2	0	#
$g_0(n_{28})$	1	0	0	1	2	0	#
$g_0(n_{35})$	1	0	0	1	2	0	#
$g_0(n_{42})$	1	0	0	1	2	0	#

g_0 values of .17

Errors in Winning Ways

Not only has the computer allowed us to **work** out more genus sequences than were printed in **WW,** it also has revealed some errors in the computations printed there.

Octal code	heap size	**WW** genus (sharp)	correct genus (blurry)
.14	**21**	0(620)	0(02)
.16	**19**	2(31)	1(##20)
.6	**29**	2(20)	2(0#20)
.6	**30**	0(02)	0(#20)
.72	**15**	2(20)	2(1#20)

All of these corrections can easily be verified by hand in a few minutes, except for the two errors in **.6.** I have verified by hand (a calculation too lengthy to be included here) that the heap of **29** counters in **.6** is a P position, as indicated by the computer. The genus sequences for .6 appear in appendix I under the code **.37.** Guy and Smith have shown that **a** heap of n + **1** counters in **.6** is equal to a heap of n counters in .37.

See also the section in the chapter 'An algorithm for Generalized Genera' about more errors in ***Winning Ways.***

A BIGGER GENUS SEQUENCE

When we were considering the genus sequence earlier in this work, and were deciding what games to include in the base set X, the condition of descent closure forced us into the choices of **2** and **1,** and the condition of additive closure mad us set $X = \langle 1,2\rangle$. If we wish to work with a larger base set, say $\langle 1,2,a\rangle$, we now have an infinite number of choices for a, which include any games all of whose options are in the set $\langle 1,2\rangle$. Some of the more common such games are $\{4,2,1\}$, $(4,1\}$, and $\{4,0\}$.

Let us consider how we might write such a genus sequence. Let $X = \{1,2,a\}$ for some game a with $a' \in \{1,2\}$. Naïvely, for a game G, we want to write down the outcomes of all sums $G + i \cdot a + j \cdot 2 + k \cdot 1$. However, we have already worked out a much shorter way to write the outcomes $H + j \cdot 2 + k \cdot 1$ for any game H; thus all we need is a sequence of genus sequences, one of each of $G + i \cdot a$. For example, since we know that the genus of 0 is **1202,** and that of $(4,1\}$ is $1\#20$, and we can compute the genus of $\{4,1\} + \{4,1\}$ which is also **1202,** we have that for $a = (4,1\}$, the $\langle 1,2,a\rangle$ genus of 0 can be written as

(1202) (1#20) **(1202) ...**

Unlike the blurry genus, we do not know that this will necessarily be an ultimately periodic sequence; hence it seems that we are dealing with a much more unwieldy sequence here.

The first move I will make towards tidying up this sequence will be to shorten the notation of the blurry genus sequence.

Returning to the sequence of $\mathcal{N}$'s and $\mathcal{P}$'s which underlie the blurry genus sequence, we notice that all we are really interested in is the location of the Pentries. So, for the string

NPNNPNNNPNNN
0 1 2 3 4 5 6 7 8 9 ...

All that **we** need to remember are the locations

$$1,4,8,12,\ldots$$

However, since we know that the sequence of $\mathcal{N}$'s and $\mathcal{P}$'s has ultimate period **4,** we know that this sequence has ultimate arithmetic period **1,** saltus **4;** with this in mind, we need only write

1 4

These sequences are usually very short, often only one or two entries long. Since it is easier to write long sequences horizontally, we will write these short sequences vertically, thus:

$$\begin{matrix}4\\1\end{matrix}$$

If the sequence of $\mathcal{P}$'s and N's has finitely many F's, NPNNNNNN. . .

then we will write the symbol ∞ to indicate that the next (unwritten) entry never occurs, thus:

$$\genfrac{}{}{0pt}{}{\infty}{1}$$

Now the $\langle \mathbf{1},\mathbf{2},a\rangle$ genus of **0** looks like

$$\genfrac{}{}{0pt}{}{464}{111}"'$$

The algorithm for computing the genus of a game given that of its options in this vertical notation is most easily described in terms of admissible sequences, i.e., those sequences which can occur as genus sequences of games. In this notation, the admissible sequences are the strictly increasing sequences of nonnegative integers, such that the integer which follows n in the sequence is greater or equal to n $+\mathbf{3}$ for n odd, and greater or equal to n $+\mathbf{4}$ if n is even. (This rule reflects the fact that in the old notation we were not allowed to follow a 0 or a **1** by a 0 or a **1). A** sequence also must have ultimate arithmetic period **1** saltus **4** to be admissible. We will show later that all admissible sequences do arise as genus sequences of actual games. Now the rule for computing the genus of a game given the genera of its options is as follows:

The genus of 0 is $\genfrac{}{}{0pt}{}{4}{1}$. For G $\neq$ 0, the genus of G is the lexicographically first admissible sequence which contains no integer contained in the genus of any option of G.

Notice the similarity between this operation and the mex operation defined earlier. For this reason I will call this function *mex!*.

It is easy to see, by casting this rule in the notation of strings of **0**'s, **1**'s and #'s, that this is the same rule as given earlier.

We are now in a position to present an algorithm for computing the $\langle \mathbf{1},\mathbf{2},a\rangle$ genus sequence, via analogy with the algorithm for computing the $\langle \mathbf{1},\mathbf{2}\rangle$ genus sequence. If we write the $\langle \mathbf{1},\mathbf{2},a\rangle$ genus of G as a sequence of $\langle \mathbf{1},\mathbf{2}\rangle$ genera, which we will write as $h_0(G)$, $h_1(G)$, $h_2(G)$, etc. (just as we wrote the $\langle \mathbf{1},\mathbf{2}\rangle$ genus of **G** as a sequence of **(1)** genera, which we called $g_0(G)$, $g_1(G)$, etc.), then we have the following rule:

$h_0(0) = \genfrac{}{}{0pt}{}{4}{1}$. If G $\neq$ 0, then $h_0(G)$ is the mex! of the $h_0(G')$.
For all G, $h_{i+1}(G) = \text{mex!}\{h_{i+1}(G'), h_i(G+a')]$.

Of course, since $a' \in \langle \mathbf{1},\mathbf{2}\rangle$, the computation of $h_i(G+a')$ is just a shift of $h_i(G)$. For example, if $h_i(G) = \genfrac{}{}{0pt}{}{5}{0}$, and **a' = :4,** then $h_i(G+\mathbf{a'}) = {}_1$, i.e., subtract **4** from **0,5,9,13,. . .,** to get **1,5,9,13,. . .;** or, in vertical notation, ${}_1$.

I will now show a prolonged calculation using this notation and algorithm, not only because this uses everything which I have mentioned so far, but also because this calculation itself is of some interest.

For this calculation, let $a = \{\mathbf{4},\mathbf{1}\}$.

First, I calculate the $\langle \mathbf{1},\mathbf{2},a)$ genus of 0:

$$\begin{array}{ll|ccc}
\text{carries} & \left\{\begin{array}{l} +:\mathbf{1} \\ +:\mathbf{4} \end{array}\right. & & \begin{array}{c}\genfrac{}{}{0pt}{}{5}{0}\\ \mathbf{0}\end{array} & \begin{array}{c}\genfrac{}{}{0pt}{}{7}{0}\\ \mathbf{2}\end{array} \\
\text{options of } 0 & & & & \\
\hline
\text{genus of } 0 & & \genfrac{}{}{0pt}{}{4}{1} & \genfrac{}{}{0pt}{}{6}{1} & \genfrac{}{}{0pt}{}{4}{1}
\end{array}$$

From this we can easily compute the $\langle 1,2,a\rangle$ genus of **1** and **2**, by simply shifting all the component $\langle 1,2\rangle$ genera by **1** and **2** respectively:

genus of 1	$\substack{5\\0}\ \substack{7\\0}\ \substack{5\\0}\ \ldots$
genus of 2	2 4 2 ...

Now the genus of $2_+ = \{2\}$:

carries	+ : 1		1	3
	+ : 4		0	2
options of 2_+	2	2	4	2
genus of 2_+		0	2	0

Now the genus of $G_0 = \{1,2,2_+\}$

carries	+ : 1		g	o	2
	+ : 4		3	1	3
options of G_0	1	$\substack{5\\0}$	$\substack{7\\0}$	$\substack{5\\0}$	$\substack{7\\0}$
	2	2	4	2	4
	2_+	0	2	0	2
genus of G_0		$\substack{7\\1}$	1	3	1

And $H = \{1,G_0\}$

carries	+ : 1		3	5	$\substack{7\\0}$	$\substack{9\\2}$
	+ : 4		2	0	2	4
options of H	1	$\substack{5\\0}$	$\substack{7\\0}$	$\substack{5\\0}$	$\substack{7\\0}$	$\substack{5\\0}$
	G_0	7	1	3	1	3
genus of H		2	4	$\substack{6\\1}$	$\substack{8\\3}$	$\substack{6\\1}$

Notice how, as in the $\langle 1,2\rangle$ case, all the genus sequences **are** ultimately periodic with **period 2 though they** take longer and longer before the periodicity starts. Any hopes in this direction are shattered by the genus of $J = \{1,H\}$:

carries	+ : 1		$\substack{5\\0}$	$\substack{7\\0}$	$\substack{9\\2}$	$\substack{11\\4\\0}$	$\substack{13\\6\\3}$	$\substack{15\\8\\5\\0}$
	+ : 4		0	2	4	$\substack{6\\1}$	$\substack{8\\3}$	$\substack{10\\5\\0}$
options of J	1	$\substack{5\\0}$	$\substack{7\\0}$	$\substack{5\\0}$	$\substack{7\\0}$	$\substack{5\\0}$	$\substack{7\\0}$	$\substack{5\\0}$
	H	2	4	$\substack{6\\1}$	$\substack{8\\3}$	$\substack{6\\1}$	$\substack{8\\3}$	$\substack{6\\1}$
genus of J		$\substack{4\\1}$	$\substack{6\\1}$	$\substack{8\\3}$	$\substack{10\\5\\1}$	$\substack{12\\7\\2}$	$\substack{14\\9\\4\\1}$	$\substack{16\\11\\7\\2}$

It is easier to see how this trend continues if we write this genus as an array of outcomes. This array is laid out in exactly the same way **as** the genus sequence written above. The numbers written horizontally across the bottom indicate the numbers of copies of the game a, while the vertical direction indicates an adder. So the entry in column ***6*** row ***5*** indicates the outcome of $6 \cdot a +\ :5$. I have found that this diagram is easier on the eyes if I use ***P*** for Pand · for $\mathcal{N}$.

20	*P*	·	*P*	·	*P*	·	*P*	·	*P*	·	·	·	·	·
19	·	·	·	·	·	·	·	·	·	·	***p***	·	***p***	·
18	·	*P*	·	*P*	·	*P*	·	*P*	·	·	·	·	·	·
17	·	·	·	·	·	·	·	·	·	*P*	·	*P*	·	*P*
16	*P*	·	*P*	·	*P*	·	*P*	·	·	·	·	·	·	·
15	·	·	·	·	·	·	·	·	*P*	·	*P*	·	*P*	·
14	·	***P***	·	***P***	·	***P***	·	·	·	·	·	·	·	·
13	·	·	·	·	·	·	·	*P*	·	*P*	·	*P*	·	*P*
12	*P*	·	*P*	·	*P*/	·	·	·	·	·	·	·	·	·
11	·	·	·	·	·	·	*P*	·	*P*	·	*P*	·	*P*	·
10	·	*P*	·	*P*	·	·	·	·	·	·	·	·	·	·
g	·	·	·	·	·	*P*	·	*P*	·	*P*	·	*P*	·	*P*
8	***P***	·	*P*/	·	·	·	·	·	·	·	·	·	·	·
7	·	·	/·	·	*P*	·	*P*	·	*P*	·	*P*	·	*P*	·
6	·	***p***	·	·	·	·	·	·	·	·	·	·	·	·
5	·		·	*P*	·	·	·		·	·	·	·	·	·
4	***p***	·	· ·	·	·	*P*	·	*P*	·	*P*	·	*P*	·	*P*
3	***í***	·	***p***	·	·	·	·		·	·	·	·	·	·
2	·		·	·	*P*	·	*P*	·	*P*	·	*P*	·	*P*	·
1	*P*	*P*	·	*P*	·	*P*	·	*P*	·	*P*	·	*P*	·	*P*
0	·	·	·	·		·	·	·	·	·		·	·	·
	0	1	2	3	4	5	6	7	8	9	10	11	12	13

The pattern that keeps this from being a periodic sequence is the highlighted strip of Upositions which goes diagonally up to the right through the top of this diagram. It is easy to see that this continues, and hence no two columns of this array match.

An example of this sort demonstrates that the $\langle 1, 2, a\rangle$ genus sequence will not be as useful **as** the $\langle 1, 2\rangle$ genus, since we cannot be certain that this will be a periodic sequence, which would allow us to write it conveniently in a small space.

It is easy to see how we could continue to define genus sequences for larger and larger sets ***X***. If we choose a game ***b*** such that $b' \in \langle 1, 2, a\rangle$, then we can define the $\langle 1, 2, a, b\rangle$ genus sequence as a sequence of $\langle 1, 2, a\rangle$ genus sequences. The **rulë** for computing such **a** genus, for a game G, given the genera of the options G', is also easy; we write the $\langle 1, 2, a, 6)$-genera of G' out in rows of $\langle 1, 2, a)$ genera, and use the $\langle 1, 2,$ a) genus algorithm on the columns (analogous to the way we used mex! on the columns of the $\langle 1, 2, a\rangle$ genus, and mex on the columns of the $\langle 1, 2\rangle$ genus). To each new column we carry the result of the previous column, shifted by **all** of the options of ***b*** (since we chose $b' \in \langle 1, 2, a\rangle$, these computations will indeed by simple shifts). What is not so easy to see is how we can conveniently write a $\langle 1,2,a,b\rangle$ genus sequence, which we should, perhaps, refer to as a genus array. The problem becomes worse and worse as we choose larger and larger sets for X. The genus sequence continues to be useful for conceptualizing the information we need to know about sums of games as we consider larger sets X, but because it is impossible to write the genus sequence in many dimensions, I will abandon it in favor of what I will call the ***generalized genus statement.***

THE GENERALIZED GENUS STATEMENT

Although a quick glance at table iii will allow us, during a high-stakes round of Grundy's game, to declare confidently that we can win the single heap of **76** coins (our move!), it is not so effective at making good that boast, because it does not tell us what moves to make (beyond the first). Of course, the computer did have the data available to tell us the right moves, but to print enough data of this sort to allow us to play Grundy's game effectively would yield a completely unwieldy table. Even if we did print such a table, it would be useless for telling us whether we can win a sum of **8** heaps **of** size **25** (say). In **ONAG,** Conway states a result which will answer questions of this sort **€or** Grundy heaps of up to **27** coins. I will work out a method of making such statements so that the computer can tell us these things. In order to do this, I will introduce the ***generalized genus statement.***

If two games G and H have the same outcome, I will write $G \sim H$. We say that $G=H$ if $G + T \sim H + T$ for all games T. If $G + H \sim H + T$ for all T in some set of games X, then I will write

$$G \equiv H \qquad (X) \tag{1}$$

This notation allows us to express conveniently several things about genus sequences. For example, when X is closed under addition and descent, we can define the X genus sequence as above, and (1) tells us that G and H have the same X-genus sequence.

Even if the set X is not closed (under addition, descent, or both), a generalized genus statement may still be interpreted in terms of genus sequences. Consider the statement

$$2 + 2 \equiv 0 \qquad (\langle 1,2\rangle + 1 + 2) \tag{2}$$

This says that the $\langle 1,2\rangle$ genus of the game $1 + 2$ has period 2 starting immediately, or equivalently, that the $\langle 1,2\rangle$ genus of 1 has period 2 starting at the second term. Of course, these two interpretations are reflected in two different ways of writing **(2):**

$$2 + 2 + 2 \equiv 2 \qquad (\langle 1,2\rangle + 1) \tag{2'}$$

This is an example of the sorts **of** operations we can perform on the genus statement. In Lemma 3 I catalogue the simple transformations we will **use** for genus statements.

Lemma 3.

i) $G + S \equiv H + S \quad (X) \iff G \equiv H \quad (X + S)$

iia) $G \equiv H \quad (X)$ and $G \equiv H \quad (Y) \iff G \equiv H \quad (X \cup Y)$

iib) $G \equiv H \quad (X) \implies G \equiv H(W)$ for any $W \subset X$

iii) $G \equiv H \quad (X)$ and $H \equiv J \quad (X) \implies G \equiv J(X)$

iv) For $S \in (Y)$, $G \equiv H \quad ((Y) + X) \implies G + S \equiv H + S \quad ((Y) + X)$

Proof. Trivial.

(iib) is just a special case of (iia), but since we will use the form (iib) so often, I have written it out explicitly.

Periodicity in Genus Sequences

We have seen that the multi-dimensional genus sequence has not retained the splendid periodicity properties enjoyed by the $\langle 1,2\rangle$ genus. However, experience calculating hundreds of such genus sequences (to be discussed later in this work) has shown that non-periodic genus sequences are quite rare. I will now present a result which will tell us when a genus sequence has begun to be periodic. Armed with this result, I will assemble an algorithm to compute generalized genera, similar to the one which computes $\langle 1,2\rangle$ genera. Since we can tell when a genus is periodic, the algorithm can guarantee us that it has not run into trouble (i.e., with a non-periodic genus).

One note before I proceed; although the genus sequence is intuitively satisfying, it tends to become unwieldy in an abstract discussion. For this reason, I will make all precise statements in terms of the genus statement, with occasional comments on the corresponding genus sequence interpretations.

In the example above where we computed the $\langle 1,2,a\rangle$ genus of $H = \{1,\{I,2,2_+\}\}$, the second time that we got $\substack{6\\1}$ as a result of a column computation, it was obvious that the answer would continue to repeat $\substack{6868\\1313}\ldots$ from that point onward. Lemma 4 casts this observation into the language of genus statements, and makes specific what conditions must be met by the games involved so that this periodicity will occur.

Lemma 4. (One-dimensional genus periodicity): Let S,N be sets of games, A a game, t,m positive integers such that

i) $t\cdot A\equiv 0 \qquad ((S)+N+m\cdot A)$
ii) $A' \in \langle S\rangle$ for all options A' of A
iii) $\langle A,S\rangle$ and N are closed under descent

then

$$t\cdot A\equiv 0 \qquad (\langle A,S\rangle + N + m\cdot A)$$

In terms of genus sequences, this tells us that, subject to the conditions on A, S and N, once the sequence repeats itself, it will continue to do so in a periodic fashion. If we set $t=2$, $m=2$, $A=\{4,1\}$, $N=[H]$, and $S=\langle 1,2\rangle$ (by $[H]$I mean the descent closure of H; that is, H, the options of H, the options of the options of H, etc.), then we get the example which we worked out earlier for the $\langle 1,2,a\rangle$–genus of H, the conclusion stating, as we observed in the example, that the final sequence is periodic.

Proof.
The conclusion is clearly equivalent to the statement that for all $s\in (S) h\in N, n\geq 0$

$$n\cdot A+s+h+m\cdot A \sim n_t\cdot A+s+h+m\cdot A$$

by n_t we mean the smallest nonnegative integer congruent to n (mod t).

Choose $n\cdot A+s+h+m\cdot A$ a simplest counterexample.

$$n\cdot A+s+h+m\cdot A \not\sim n_t\cdot A+s+h+m\cdot A$$

Clearly, $n>0$.

If $n \equiv 0 \pmod t$, I claim that

$$n \cdot A + s + h + m \cdot A \sim s + h + m \cdot A$$

Examine the options of the left hand side,

$$A' + (n-1) \cdot A + s + h + m \cdot A \sim A' + (t-1) \cdot A + s + h + m \cdot A$$
$$\text{and } n \cdot A + (s+h)' + m \cdot A \sim t \cdot A + (s+h)' + m \cdot A$$

both by the inductive hypothesis. But the right hand side of these similarities constitute all options of $t \cdot A + s + h + m \cdot A$, i.e., for any option of $n \cdot A + s + h + m \cdot A$ we can find an option of $t \cdot A + s + h + m \cdot A$ with the same outcome. Hence,

$$n \cdot A + s + h + m \cdot A \sim t \cdot A + s + h + m \cdot A \sim s + h + m \cdot A$$

and $n \cdot A + s + h + m \cdot A$ was not a counterexample after all.

If $n \not\equiv 0 \pmod t$, I claim that still

$$n \cdot A + s + h + m \cdot A \sim n_t \cdot A + s + h + m \cdot A$$

since the options of the left hand side

$$A' + (n-1) \cdot A + s + h + m \cdot A \sim A' + (n-1)_t \cdot A + s + h + m \cdot A$$

and

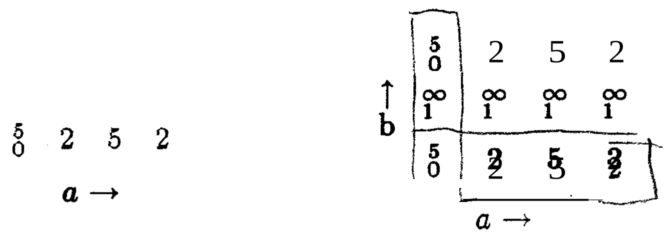
$$n \cdot A + (s+h)' + m \cdot A \sim n_t \cdot A + (s+h)' + m \cdot A$$

have the same outcome as the options of the right hand side, so again $n \cdot A + s + h + m \cdot A$ is not a counterexample. ■

If we view a d-dimensional genus sequence (i.e., one which characterizes classes over some set $\langle G_1, G_2, \ldots G_d \rangle$) as a sequence of $(d-1)$-dimensional sequences, this lemma allows us to conclude statements about the periodicity of this sequence. However, we often find that a high-dimensional symbol is periodic in several directions at once, in the sense that blocks of values repeat themselves. By analogy, we might expect that once the boundaries of such a block match, we can guarantee that the sequence is periodic (see fig. 1). Indeed this is the case. Most of the complication in the proof arises from the need to check the conditions on A, S, and N to allow us to use lemma 4.

When we discuss an n-dimensional genus sequence, which we may think of as an n-dimensional 'cube' filled with outcomes, it will be convenient to refer to the $(n-1)$-dimensional faces of this 'cube', as well as various cross-sections. The outcomes in the 'cube' refer to sums of all n games. Each $(n-1)$-dimensional face of the 'cube' refers to sums of $(n-1)$ of these games. For the $\langle g_1, g_2, \ldots g_n \rangle$-genus sequence: we define $F_{n,i}$ to be the set of games referred to by the $(n-1)$-dimensional face for which the sums do not involve the game g_i · $F_{n,i}$ is given by the formula

$$F_{n,i} = \left\{ \sum_{j=1}^{n} a_j \cdot g_j \;\middle|\; a_i = 0 \right\}$$



Let a $= \{4,2,1\}$ and $b = \{$a $: 2, a, 2, 1\}$. The $\langle 1,2,a\rangle$-genus of f $=$ $\{0,3,6,7\}$ begins to show periodicity by the 3^{rd} term; Lemma 2 tells us that since the third entry matches the $3-2=1^{st}$ entry, this sequence is periodic of period 2. The $\langle 1,2,\text{a},6\rangle$-genus of f (shown as a 2-dimensional array of $\langle 1,2\rangle$-genus sequences) begins to show periodicity by the 3^{rd} column, 2^{nd} row. The 3^{rd} column matches the first, and the second row matches the zeroth. Can we conclude that the sequence is henceforth periodic in both directions?

The entries corresponding to the sets $B_{2,1}$ (vertical) and $B_{2,2}$ (horizontal) are highlighted.

fig. 1

When we suspect that the $\langle g_1, g_2, \ldots g_n\rangle$-genus sequence is periodic in the i^{th} direction of period p_i which begins after s_i entries, we wish to show that any values beyond $p_i + s_i$ will not concern us. Hence I will use $B_{n,i}$ to indicate this part of the face $F_{n,i}$.

$$B_{n,i} = \left\{ \sum_{j=1}^{n} a_j \cdot g_j \mid a_j \leq s_j + p_j, a_i = 0 \right\}$$

When we wish to refer to cross-sections of this cube, we can add appropriate multiples of base games; for instance, the cross-sections parallel to $B_{n,i}$ refer to the sets $B_{n,i}$, $B_{n,i} + g_i$, $B_{n,i} + 2 \cdot g_i$, etc.

T eorem III.

For a set G, $= \{g_1, g_2, \ldots g_n\}$ *of n* games, denote by G_i the subset $\{g_1, g_2, \ldots g_i\}$. S ppose that **for** some sets of games T, N and some integers $s_i, p_i \geq 0, (1 \leq i \leq n)$,

i) $p_i \cdot g_i \equiv 0(\langle t\rangle + N + s_i \cdot g_i + B_{n,i}) \qquad (1 \leq i \leq n)$

i) g_i fit together in a cascade fashion, i.e.,

$$\begin{gathered} g_1' \in \langle T\rangle \\ g_2' \in \langle T, G_1\rangle \\ g_3' \in \langle T, G_2\rangle \\ \cdot \\ g_n' \in \langle T, G_{n-1}\rangle \end{gathered}$$

iii) $\langle T, g_1\rangle$ and N are closed under descent.

Then

$$p_i \cdot g_i \equiv 0 \qquad (\langle T G_n\rangle + N + s_i \cdot g_i) \qquad 1 \leq i \leq n$$

Proof

First, we notice that because of the cascade property, $\langle T G_i\rangle$ is closed under descent fc r all i, not just $i = 1$.

Proceed by induction on n. For $n = 1$, this is just Lemma 4.

For $n > 1$, we have that G_{n-1} satisfies all of the conditions in this theorem, so by ir duction we may conclude that

$$p_i \cdot g_i \equiv 0 \qquad (\langle T, G_{n-1}\rangle + N + s_i \cdot g_i) \qquad 1 \leq i \leq n \tag{1}$$

Now I claim that

$$p_i \cdot g_i \equiv 0 \qquad (\langle T, G_{n-1}\rangle + N + s_i \cdot g_i + [(s_n + p_n) \cdot g_n]) \qquad 1 \leq i \leq n-1 \tag{2}$$

If not, then choose k smallest such that

$$p_i \cdot g_i \equiv 0 \qquad (\langle T, G_{n-1}\rangle + N + s_i \cdot g_i + [k \cdot g_n]) \qquad 1 \leq i \leq n$$

fails. **Now** we know that $0 < k \leq p_n + s_n$. We want to work out an expression for $[k.g_n]$ that we can handle.

Since $(k \cdot g_n)' \in (k-1) \cdot g_n + \langle T, G_{n-1}\rangle$ and $\langle T, G_{n-1}\rangle$ is closed under descent, we have that

$$[k \cdot g_n] \subset \{k \cdot g_n\} \cup \bigcup_{j<k} j \cdot g_n + \langle T, G_{n-1}\rangle \tag{3}$$

Now, by minimality of k we have that

$$p_i \cdot g_i \equiv 0 \quad (\langle T, G_{n-1}\rangle + s_i . g_i + N + [j \cdot g_n]) \quad 1 \leq i \leq n \quad \forall j < k \tag{4}$$

and by Lemma 3 (iia),

$$p_i \cdot g_i \equiv 0 \quad (\langle T, g_{n-1}\rangle + s_i \cdot g_i + N + \bigcup_{j<k} [j \cdot g_n]) \quad 1 \le i \le n$$

by Lemma 3(iib)

$$p_i \cdot g_i \equiv 0 \quad (\langle T\rangle s_i \cdot g_i + N + B_{n-1,i} + \bigcup_{j<k} (j \cdot g_n + \langle T, G_{n-1}\rangle)) \qquad (5)$$
$$1 \le i \le n$$

Lemma 3(iib) was applicable because the base set of (5) is a subset of the base set of (4).

Since $k \le s_n + p_n$, we have that $B_{n-1,i} + k \cdot g_n \subset B_{n,i}$, so from (i) we have

$$p_i \cdot g_i \equiv 0 \quad ((T) s_i \cdot g_i + N + B_{n-1,i} + k \cdot g_n) \qquad (6)$$

We can now apply lemma 3(iia) to congruences (5) and (6) to obtain

$$p_i \cdot g_i \equiv 0 \quad (\langle T\rangle + s_i \cdot g_i + N + B_{n-1,i} + \{k \cdot g_n\} \cup \bigcup_{j<k} j \cdot g_n + \langle T, G_{n-1}\rangle)$$
$$1 \le i \le n-1$$

Now, from (3) and Lemma 3(iib) we have

$$p_i \cdot g_i \equiv 0 \quad (\langle T\rangle + s_i \cdot g_i + B_{n-1,i} + [k \cdot g_n] + N \qquad 1 \le i \le n$$

But $\langle T, G_{n-1}\rangle + [k \cdot g_n]$ is descent closed, so by induction we have

$$p_i \cdot g_i \equiv 0 \quad (\langle T, g_{n-1}\rangle + s_i \cdot g_i + [k \cdot g_n] + N) \qquad 1 \le i \le n$$

contrary to our choice of k. This proves (2).

We can specialize (2) to tell us that

$$p_i \cdot g_i \equiv 0 \quad (\langle T, G_{n-1}\rangle + s_i \cdot g_i + (s_n + p_n) \cdot g_n) \qquad 1 \le i \le n-1$$

and

$$p_i \cdot g_i \equiv 0 \quad (\langle T, g_{n-1}\rangle + s_i \cdot g_i + s_n \cdot g_n) \qquad 1 \le i \le n$$

So that for any $\sigma \in \langle G_{n-1}\rangle$, there exists a $\sigma_b \in B_{n,n}$ such that

$$\sigma \equiv \sigma_b \quad (\langle T\rangle + N + (s_n + p_n) \cdot g_n)$$

and

$$\sigma \equiv \sigma_b \quad (\langle T\rangle + N + s_n \cdot g_n)$$

From (i) we have

$$p_n \cdot g_n \equiv 0 \qquad (\langle T\rangle + N + s_n \cdot g_n + \sigma_b)$$

since $\sigma_b \in B_{n,n}$.

Combining these together we get

$$\sigma + p_n \cdot g_n \equiv \sigma_b + p_n \cdot g_n \equiv \sigma_b \equiv \sigma \qquad (\langle T\rangle + N + s_n \cdot g_n)$$

or equivalently,

$$p_n \cdot g_n \equiv 0 \qquad (\langle T, g_{n-1}\rangle + N + s_n \cdot g_n)$$

We check that $(T,G,)$ and N are closed under descent, so we can apply lemma 4 to obtain

$$p_n \cdot g_n \equiv 0 \qquad (\langle T, G_n\rangle + N + s_n \cdot g_n)$$

and hence,

$$p_i \cdot g_i \equiv 0 \qquad (\langle T, G_n\rangle + N + s_i \cdot g_i) \qquad 1 \leq i \leq n$$

■

There may be some concern about the applicability of this theorem because of the complicated nature of its hypotheses. However, the conditions of descent are almost always satisfied. In fact, any nim-like game will automatically satisfy **these** conditions. The options of any heap are just sums of earlier heaps (hence these heaps satisfy the cascade property). If we let $T = N = \{0\}$, and let g_1 be the heap of size 1, then $\langle T, g_1\rangle$ and N are closed.

AN ALGORITHM FOR GENERALIZED GENERA

Although Theorem III will not tell us that all genus sequences are periodic, it is very useful in computing those which are. In fact, examination of appendix II shows that no heap in any octal game shown has a non-periodic genus. Appendix II was prepared by the following algorithm, which exploits theorem 111, and the way in which the octal games automatically satisfy the hypotheses.

Since this algorithm is geared towards octal games, we will refer to fundamental positions as heaps, and will denote the heap of i counters as hi, and the set of positions $\{h_1,, h_2, \ldots h_i\}$ as Hi.

For each hi, do the following:

Assume that the $\langle 1,2,H_{i-1}\rangle$-genus of the game 0 is known, i.e., we know the $\langle 1,2\rangle$-genus of all sums of heaps smaller than h_i.

Look up all the options of h_i in this array, and compute the $\langle 1,2\rangle$-genus of h_i.

Check whether h_i is an adder (as before);

If not, then compute the $\langle 1,2\rangle$-genus of all sums of h_i and games in H_{i-1}, i.e., compute the $\langle 1,2,H_{i-1}\rangle$-genus sequence of h_i, $h_i + h_i$, etc. until a match is found in this sequence. By theorem III, this gives us the $\langle 1,2,H_i\rangle$-genus of 0.

The performance of this algorithm does not seem as impressive as that of the algorithm for the blurry genus, since we cannot give results for very large heaps. The results of running this algorithm on the octal games from WW are tabulated in appendix II.

For the games this algorithm does treat, the information we gain is very impressive. Thanks to theorem III, we know the outcome of arbitrary sums of the positions tabulated in any column of appendix 11. This is one step towards giving a complete solution to a game; often we find (as in the case of the game of Knots, which will be treated in the next section) that we have some rule for determining the outcome of a position provided that it contains at least one large heap. The output of the generalized genus algorithm can help us with the sums of small heaps.

Even when a complete solution is not obtained, the data in appendix II will guide us while playing the games which it treats. This is best shown by an example.

Suppose we are playing the game of Officers from WW (the octal code for officers is .37; see page **5** of appendix 11), and we are presented with the following 10 heap position:

2 7 10 10 11 11 11 11 11 13

From appendix II we find that the officers heaps of 2 and **7** counters equal $*2$ and $*1$ respectively. For the heap of 10 counters, we find the line

10 **0,2** index # **2.**

The first part of this line tells us that we may replace **2** + 0 heaps of **10** counters **by 0** such heaps. Similarly, the line for **11,**

11 2,1 index # **3,**

tells us that we may replace **2** + **1** heaps **of** size **11** by **2** such heaps. We have reduced our original sum to the following sum:

***3111113**   *3   11   11   13

The line for **11** gives an 'index' of **3** for **11,** and the line for **13** gives index **2.** We use these indices in the next part of the table for Officers (**.37**). The remaining sum has **2** index **3** games and one index **2** game, so we look for the line in appendix II with a '**2**' in column **3** and a '**1**' in column **2** (and a 0 in column **1,** since we have no index **1** games). This gives us the genus (####), so we know that this is an $\mathcal{N}$ position.

A reasonably simple search through the options of our original sum reveals the option

2 7 9 10 11 11 11 11 11 13

Since heaps of size **10** and **13** are both index **2,** we may cancel them in pairs as we did heaps of size **10.** This leaves us with the sum

***391111**   *3   9   11   11

We **look** up the line with **1** in column **1** (**9** is index **1**) and **2** in column **3** (**11** is index **3**), to find the genus (**3131**), so this sum is a $\mathcal{P}$ position, and is a correct move to make.

Errors in Winning Ways

In chapter **13** of WW, Berlekamp, Conway and Guy state that when computing the genus of a sum of heaps in Grundy's Game, we can pretend that the various heaps are equal, according to the following relations and addition table:

$G13= G16= G19= G22= G25= \boldsymbol{a}$ (say)
$G18= G21= G24= G27= b$ (say)
$G20= G23= G26=$ c (say)

$$a + a = 0$$
$$b + b = 0$$
$$c + c + c = c + c$$

+	0	*c*	c+c
0	1202	2020	0202
a	1431	4313	3131
b	0564	5646	4646
a + *b*	05875	5875	7575

The computer has verified this principle for the blurry genus. However, for the sharp genus, this statement is not true.

Consider the sum $G18$ + **G20** + **G13** + **G25.** The above relation tells us that **G13** and G**25** should be interchangeable, so we expect that

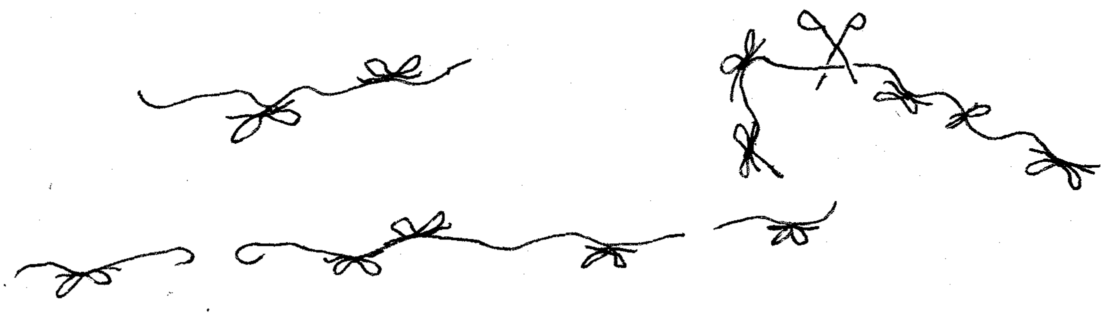
$$\mathbf{G18} + \mathbf{G20} + \mathbf{G13} + \mathbf{G25} \equiv \mathbf{G18} + \mathbf{G20} + \mathbf{G13} + \mathbf{G13} \qquad (\langle 1,2\rangle + \text{nimheaps})$$

We can also pretend that $a + \boldsymbol{a} = 0$, so we expect this sum to have the same genus as $G18+G20$, which is 4(564). However, $G18+G20+G13+$ **G25** has **G18-t**$G20+G13+G20+G5$ as an option, which has sharp genus 5(5757), which matches 4(564) in the first place. The actual sharp genus of this sum is 4(64964) which, of course, in blurry notation, is exactly the same as **4(564).**

THE GAME OF KNOTS TIES UP ALL GENUS SEQUENCES

We can use the information produced by the generalized genus to help us obtain a complete solution of the game of Knots.

The game of Knots is played with bits of string tied end to end, and a pair of scissors. All knots are treated the same, since they join two distinct pieces of string. There are two sorts of legal moves: one involving only a string and the other involving the scissors and a string. The first sort of move is to untie one of the knots (separating one length of string into two). The other is to use the scissors to cut *a* string between knots (also separating the string into lengths). In the octal notation of Guy and Smith, this is the game **4.7.**



making a move in Knots

As mentioned in WW, the genera of the strings of Knots look much alike, except that they take longer and longer to settle down. This trend does continue. This is a consequence of the following rule, which will determine the outcome on any position in Knots (we denote a string of n knots as Sn):

We state this rule in terms of the *tameness* $t(n)$ and *wildness* $w(n)$ of Sn as given by the following table:

$n =$	0	1	2	3	4	5	6	7	8	9	...
$t(n) =$	0	0	1	0	1	2	3	4	5	6	...
$w(n) =$	0	0	0	0	0	1	0	1	2	3	...

For $n \geq 3$, $t(n) = n - 3$, for $n > 6$, $w(n) = n - 6$. For all n, we have that $t(n) \geq n - 3$, and $w(n) \leq \max(0, n - 4)$.

For a Knots position K, let $n_0 \geq n_1 \geq \ldots > 0$ be the number of knots in each string. We may assume that every string has at least one knot, since no play is possible from a string with no knots.

Let $\mathbf{A} = w(n_0) - \sum_{i>0} t(n_i)$

$E(K)$ = number of strings with an even number of knots
$D(K)$ = number of strings with an odd number of knots

The outcome of K is determined by the following rule:

If every string has length 1 or 3, then K is a P position if and only if there is an odd number of strings. (The 'short strings proviso').

If there is any string of some other length, then

$$\mathbf{A} \le 0 \Rightarrow o(K) = \mathrm{P} \text{ iff E and D are both even}$$
$$\mathbf{A} = 1 \Rightarrow o(K) = \mathcal{N}$$
$$\mathbf{A} \ge 2 \Rightarrow o(K) = P \text{ iff E and D are both odd}$$

Proof.

It suffices to show that from each position which we claim to be a $\mathcal{N}$ position, we can make a *correcting* move to one of the positions we claim to be a P position, and that there is no *perverting* move from any position which we claim to be a P position to any other such position.

We note that the computer has verified this rule for strings with fewer than **9** knots. Hence we need only treat positions with at least one string of 9 or more knots. In particular, any option of such a position has at least one string with 4 or more knots, and we can safely ignore the small heaps proviso.

First we show that there are no perverting moves. For any position we list as P we have $E \equiv D \pmod{2}$. We treat the available moves in two cases.

First, we may separate a string into two non-trivial parts (either with the scissors or by untying a knot somewhere in the middle of **a** string). We notice that all P positions have an even number of strings. Such a move yields one more string, and hence cannot link a P position to another P position.

Second, we may untie a knot at the end of a string. No matter which string this is, we change the parity of both D and E, and change **A** by at most **1**, and again the result is not one of the listed P positions.

Now we show that from any listed $\mathcal{N}$ position, we can correct to a listed P position.

We now treat two cases,

i) $n_1 \le n_0 - 2$ (This includes the case when K consists of 1 string-only)
ii) $n_0 - 1 \le n_1 \le n_0$

Case (i), $n_1 \le n_0 - 2$

First we find a correcting move from any position for which D or E is even, but not both D and E are even.

For n_0 even: $n_0 = 2m$, the moves in Sn_0 to $Sm + Sm$ and $Sm + S(m-1)$ change the parity of E and D respectively. We see that for both options, $\mathbf{A} \le 0$. Since a position with $\mathbf{A} \leq 0$ is P if both D and E are even, one of these moves will correct a position for which exactly one of D and E is even.

For n_0 odd, similar remarks apply for $Sm + S(m+1)$ and $Sm + Sm$.

Next we correct positions for which both D and E are odd, and $\mathbf{A} \leq \mathbf{1}$. The move from Sn_0 to $S(n_0 - 1)$ changes the parities of both D and E, and reduces $\mathbf{A}$ by $\mathbf{1}$. Again this option has both D and E even, $\mathbf{A} \leq 0$, so it is listed as a P position.

Finally we correct positions for which both D and E are even and $\mathbf{A} \geq \mathbf{1}$. We treat separately the two cases

a) $n_i = 1$ or 3 for all $i > 0$
b) There exists some $n_i \in \{2,4,5,6,7,\ldots\}$ $(i > 0)$

In case (a), since $n_0 \geq 9$, we find that $w(n_0) \geq 3$ and $\mathbf{A} \geq 3$. The move from Sn_0 to $S(n_0 - 1)$ changes the parities of both D and E, and decreases $\mathbf{A}$ by $\mathbf{1}$, so that this option has D, E odd, $\mathbf{A} \geq \mathbf{2}$, hence it is listed as a P position.

In case (b), the play from Sn_i to $S(n_i - 1)$ increased $\mathbf{A}$ by $\mathbf{1}$, while changing the parities of both D and E. Hence for this option D and E are odd and $\mathbf{A} \geq \mathbf{2}$, so it is a P position.

Notice that the only occasion when we used a move in Sn_i for $i > 0$ was the final case (b), when we were guaranteed that it did indeed exist. Hence this discussion covers the case where $K = Sn_0$.

case (ii), $n_0 - 1 \leq n_1 \leq n_0$

Since $n_0 \geq 9$, we have $n_1 \geq 8$ and $\mathbf{A} \leq -\mathbf{2}$. The move from Sn_1 to $S(n_1 - 1) + S1$, $S(n_1 - 2) + S1$, or $S(n_1 - 1)$ changes the parity of D, E, or both D and E respectively, and changes $\mathbf{A}$ by at most $\mathbf{2}$. Hence we can find an option for which $\mathbf{A} \leq 0$ and D and E are both even.

In all cases we can find correcting moves from any $\mathcal{N}$ position to some P position, as desired.

Since the strings of 1 and 2 knots equal $:1$ and $:2$ respectively, this rule tells us about the blurry genus sequences of single long strings. Indeed, we find that the position of the blur in the genus sequence of a long string of size n is just $n - 7$.

Which sequences arise as genus sequences?

The various positions of Knots play an interesting role in answering the question, 'Which genus sequences can arise?'. We already know that any genus sequence is an ultimately periodic sequence of period 2, and that in such a sequence a 0 or a 1 is always followed by a #. We will use the games we called Si in the discussion of Knots to prove the following

Theorem IV. Every admissible sequence is the genus sequence of some game.

Proof. From appendix I we have the genera

S8(1#20)
*S*9(20#3 1)
*S*10(131#)
*S*11(2020#31)
S12(13131#20)
etc.

If we compute the genera of the following games, which I will call **Ji**, we get:

J4 = (S8 + 1 , 1^1,2,3} genus ##0#0...
J5 = (S8 ,0^0,1^1, 3} genus ##1##...
J6 = {*S*9 ,0^0,1^1, 3} genus ###0#0...
J7 = {*S*9 + 1 ,0^0,1^1,2 } genus ###1#1...
J8 = {*S*10 + 1, 1^1,2,3} genus ####0#0...
J9 = {*S*10 ,0^0, 2,3} genus ####1#1...
J10 = {*S*11 ,0^0,1^1, 3} genus #####0#0...
J11 = {*S*11 + 1,0^0,1^1,2 } genus #####1#1...
etc.

0^0 indicates any game with genus sequence 0#0#, say :4, and 1' indicates any game of genus 1#1#, say :**5**. We build one last sequence,

KO = (J4 , 1^1,2,3 } genus 0##...
K1 = {*J*5 ,0^0, 2,3 } genus 1##...
K2 = {*J*6 ,0^0,1^1, 3 } genus #0##...
K3 = (J7 ,0^0,1^1,2 } genus #1##...
K4 = {*J*8 , 1^1,2,3, *G*0 } genus ##0##...
K5 = {*J*9 ,0^0, 2,3, *G*1 } genus ##1##...
K6 = {*J*10,0^0,1^1, 3, *G*2 } genus ###0##...
K7 = {*J*11,0^0,1^1,2 , *G*3 } genus ###1##...
K8 = {*J*12, 1^1,2,3, GO, G4} genus ####0##...
etc.

For any admissible sequence we can construct a game with that sequence as its genus by choosing its options from the lists of games *Ji* and *Ki*. For example, consider the admissible sequence

# 1# # 0 # 0 #...
0 2 4 6 8 ...

We choose *K*'s and *J*'s that have 1's and 0's exactly where this sequence does *not,* thus:

$J6$	#	#	#	0	#	0	#	...
$J7$	#	#	#	1	#	1	#	...
$J9$	#	#	#	#	1	#	1	...
$K0$	0	#	#	#	#	#	#	...
$K1$	1	#	#	#	#	#	#	...
$K2$	#	0	#	#	#	#	#	...
K_4	#	#	0	#	#	#	#	...
K_5	#	#	1	#	#	#	#	...
	#	1	#	#	0	#	0	...

using the J's for the periodic part, and the K's **for** the sporadic part at the start. Thus we find that any admissible string (i.e., no adjacent 0's and 1's, ultimately alternating) does in fact occur as the genus of some game.