Combinatorial Game Theory
A Playful Introduction

Matt DeVos and Deborah Kent

Contents

Chapter 1

Combinatorial Games

The best way to get a feel for combinatorial games is to play some! Try playing these games in which two players alternate making moves.

Game 1.1 (Pick-Up-Bricks)**.** This game is played with a pile of bricks. Each move consists of removing 1 or 2 bricks from the pile. The game ends when the pile is empty, and the last player to take a brick wins.

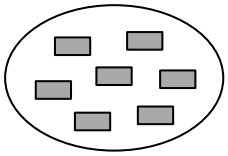


Figure 1.1: A 7-brick position in Pick-Up-Bricks

Game 1.2 (Chop)**.** Start with an $m \times n$ array viewed as a plank that is secured only at the lower left corner. On each turn, a player must either make a vertical or horizontal chop of the plank, and then any piece no longer connected to the lower left corner falls off into the water. The last player to make a move wins.

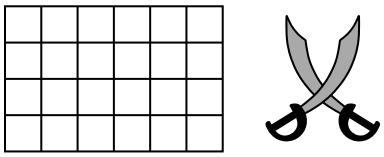


Figure 1.2: A 4×6 position in Chop

Game 1.3 (Chomp)**.** Start with an $m \times n$ array viewed as a chocolate bar, but with the lower left corner square poisoned. On each turn, a player chooses a square and eats this square and all other squares that lie above and to the right of this one (i.e. the NE corner). The last player to eat a non-poison square wins.

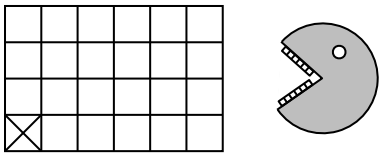


Figure 1.3: A 4×6 position in Chomp

Game 1.4 (Hex)**.** This is a game played on the board pictured in Figure 1.4. Similar to Tic-Tac-Toe, on each turn a player marks an empty hexagon. One player uses $*$ as their mark, while the other uses $\circ$. If the player using $*$ can form a chain of hexagons with this mark connecting the left and right sides of the board, that player wins. The player using $\circ$ will win by forming a chain of hexagons with this mark connecting the top and bottom.

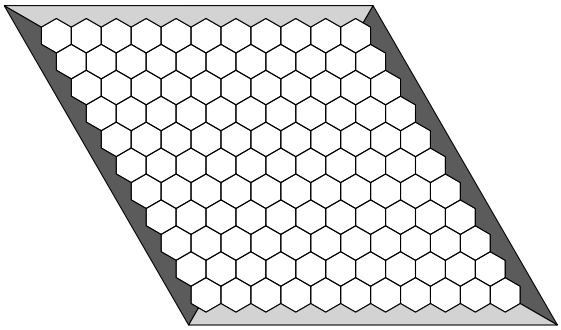


Figure 1.4: The board for Hex

Now that we have a handful of combinatorial games in mind, let's introduce some formal definitions that will allow us to treat these objects mathematically.

Definition 1.5. A *combinatorial game* is a two-player game played between **L**ouise (for **L**eft) and **R**ichard (for **R**ight).[1] The game consists of the following:

1. A set of possible *positions*. These are the states of the game (eg. a 2×3 board in Chop).

2. A *move rule* indicating for each position, what positions Louise can move to and what positions Richard can move to.

3. A *win rule* indicating a set of *terminal positions* where the game ends. Each terminal position has an associated *outcome*, either Louise wins and Richard loses (denoted $+-$), Louise loses and Richard wins ($-+$), or it is a draw (00).

Observe that the definition of a combinatorial game does not indicate which player moves first, nor does it explicitly state which position is the starting position. To *play* one of these games, we choose a starting position and designate a player to move first. From then on, the players alternate making moves until a terminal position is reached and the game ends.

[1] These names are chosen in honor of Richard Guy, one of the founders of modern combinatorial game theory, and his wife Louise.

In three of the games we have seen so far, Pick-Up-Bricks, Chop, and Chomp, the loser is the first player to have no available move. This is a common win rule, and we call a combinatorial game with this win rule a *normal-play* game.

Although numerous common games like Checkers and Chess are combinatorial games, there also exist many games that are not combinatorial games. Notably, combinatorial games have no element of randomness—so die rolls or spinners cannot be used to determine actions. Combinatorial games also require each player to have full information about the position of the game.

1.1 Game Trees

In this section, we will introduce a powerful tool called a game tree, which will be helpful for understanding the play of a game. We will begin by seeing how to model the play of any combinatorial game in this manner.

Modelling Play

There is a natural way to depict all possible sequences of moves in the play of a game using a tree, where each branch node models a choice point for one of the players and every terminal node indicates an outcome. This construct will prove extremely useful for us as we start to think strategically. We first introduce the simple game of Tic and consider its game tree.

Game 1.6 (Tic)**.** This is a combinatorial game similar to Tic-Tac-Toe but played on a 1×3 array. To move, Louise marks an empty square with a $\circ$ and Richard marks an empty square with a $\times$. If Louise or Richard gets two adjacent squares marked with their symbol, then he or she wins. If all squares get marked without this happening, the games ends in a draw. Figure 1.5 depicts a game tree for Tic starting from a blank board with Richard moving first.

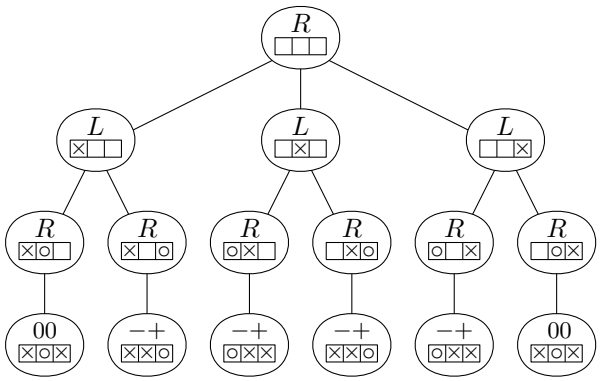


Figure 1.5: A game tree for Tic

Our game tree in Figure 1.5 depicts every possible sequence of moves starting from a blank board with Richard moving first. Observe that each node of the game tree contains the current position of the game, so we can see the positions updating as moves are made

and we work downward in the tree. Each branch node also contains either an L or R to indicate that it is either Louise or Richard's turn to play. The terminal nodes indicate that the game has ended with an outcome, either a win for Richard $-+$, a win for Louise $+-$, or a draw 00. The topmost node containing the starting position is called the *root* node.

More generally, we can model any combinatorial game with this process. We will call these figures *game trees* and they will prove quite helpful for our analysis.

Procedure 1.7 (Build a Game Tree)**.** To make a game tree starting at position α with Louise moving first, we begin by making a root node containing an L (since Louise is first) and an α (since this is this starting position). If Louise can move to positions $\alpha_1, \ldots, \alpha_k$, then we join k new nodes to the root node, each one of the new nodes contains one of these α_i positions. If any of these positions is terminal, then we put the appropriate outcome (either $+-$, $-+$, or 00) in this node. For the other nonterminal positions, it will be Richard's turn to play, so all of these nodes will contain an R. Continue this process until it is complete.

W-L-D Game Trees

We introduced the game tree in Figure 1.5 as a way to model the play of Tic. However, once we have this tree in hand, we could actually play it instead of the game. Rather than starting with an empty 1×3 array and having Louise and Richard alternately mark boxes with their symbols, we could start at the root node of this game tree and descend it by having Louise choose at the nodes marked L and having Richard choose at the nodes marked R. As you can see, these two different ways of playing are essentially equivalent.

Once we are operating in this game tree model, the position information that is contained in each node is superfluous. Indeed, all that our players need to play this game tree is the information concerning which player has a choice to make at each branch node and what the outcome is at each terminal node. Ignoring the position information for the game Tic gives us the tree depicted in Figure 1.6.

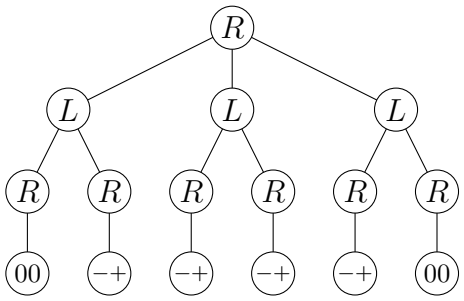


Figure 1.6: A W-L-D game tree for Tic

This type of game tree without the position information is extremely useful, so we shall give it a name. We define a **W-L-D** *game tree* (for **W**in-**L**ose-**D**raw) to be a tree with a distinguished root node (as the starting position) in which each terminal node contains an outcome ($+-$, $-+$, or 00) and each branch node contains either an L or an R indicating which player moves by making a choice at that node. The nice property of W-L-D game trees is that they give us a unified way to think about the play of combinatorial games. So, instead

of having to consider players taking tokens and marking squares and eating chocolate, we can always view play in terms of descending a W-L-D game tree.

It is possible to have a W-L-D game tree consisting of just a single terminal node (so neither player has any decisions to make and the outcome is already decided). This wouldn't be much fun to play in practice, but it will be convenient for our theory. There are three such trees with no moves (see Figure 1.7) and we will call them *trivial* game trees.

Figure 1.7: The trivial W-L-D game trees

There is another extreme to consider... it is possible that our game tree could go on forever and never end! Although the definition does not force combinatorial games to end, we will restrict our attention only to games that must end after a finite number of moves. Accordingly, we will always assume that W-L-D game trees are finite.

Strategy

When playing a game, we like to win! To do so, we will want a plan. The idea of strategy formalizes this notion of a plan. The term strategy is familiar to game players everywhere and can be used to indicate a general principle of play — in chess, for example, one may wish to "control the center" — but we will adopt a more refined and specific usage of this term. We define a *strategy* for a player in a W-L-D game tree to be a set of decisions indicating which move to make at each node where that player has a choice. In the following game tree, we have depicted a strategy for Richard by boldfacing the edges indicating his choices.

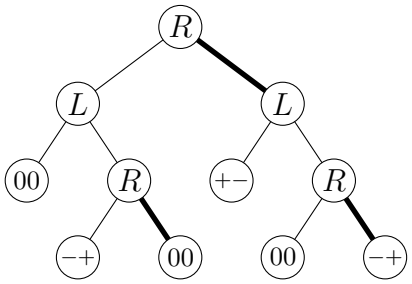


Figure 1.8: A strategy for Richard

In the play of our game, a player may *follow* a strategy by using it to make every decision. Note that the strategy indicated for Richard in Figure 1.8 is not a terribly good one, since it gives Louise the opportunity to move to a terminal node with outcome +− where Richard will lose. Richard would do better to follow a strategy that makes the opposite choice at the root node, since then it will be impossible for him to lose.

We will generally be interested in finding good strategies for our players. The best we could hope for is a strategy that guarantees a win for a player who follows it. We will call any such strategy a *winning* strategy. Next best would be a strategy that guarantees a player doesn't lose. A strategy with the property that following it guarantees either a win or a

draw is called a *drawing* strategy. Note that a player following a drawing strategy could end up winning, the only guarantee is that this player will not lose.

Let us note that there may be some extraneous information included in a strategy as we have defined it. For example, in the strategy for Richard depicted in Figure 1.8, Richard will choose the right branch as his first move. As a result, he will never encounter the node in the lower left part of the tree labelled R. Since he will never encounter this node, it may seem unnecessary for him to decide what to do there. Nevertheless, our definition of strategy includes the decision Richard would make at every node labelled R. The simplicity of this definition makes strategies easier to work with.

Working Backwards

Let's now assume that our players are highly rational with perfect foresight and consider how they might play in a large W-L-D game tree. From the nodes at the top of the tree, it is not clear what choices either player would prefer since there are so many decisions still ahead. In contrast, for the nodes close to the bottom it is much easier to see how best to play. Consider the choice for Richard in the game tree depicted in Figure 1.9. It is clear that from here Richard will choose the right branch for a win.

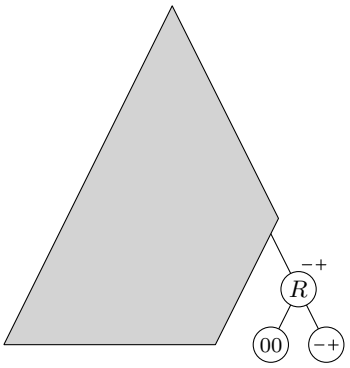


Figure 1.9: An easy decision for Richard

Since we now know that reaching Richard's decision node in Figure 1.9 will result in a win for Richard, we have written a "$-+$" next to this node. In fact, we can now think of this as a new terminal node, and then apply a similar process elsewhere in the game tree. Next we formalize this approach.

Procedure 1.8 (Working Backward)**.** Suppose one of our players has a decision to make at node N. Suppose also that we have already determined what the outcome will be under rational play for all possible nodes from node N. Then choose a best possible outcome for this player, indicate this choice by darkening this edge, and then mark the node N with the resulting outcome. Continue this process until the root node has been marked with an outcome.

Figure 1.10 shows the result of carrying this procedure to completion on a larger game tree. Note in this figure that the root node has been labelled $-+$. This means that under rational play, Richard will win this game.

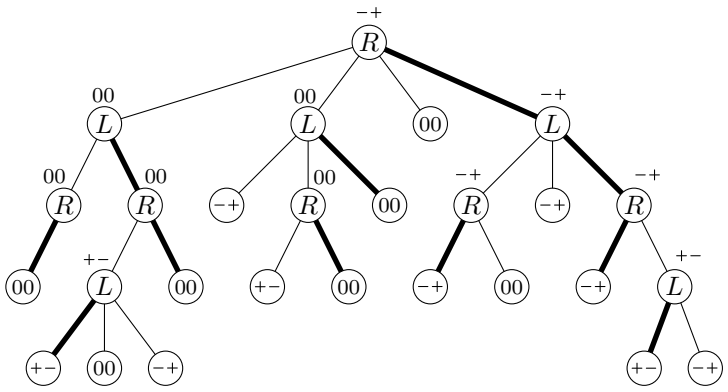


Figure 1.10: Rational decision making

In fact, as you may readily verify, the strategy for Richard indicated in Figure 1.10 is a winning strategy. In the following section, we will show that this procedure works more generally for any W-L-D game tree. More precisely, when we apply our working backward procedure to any W-L-D game tree, we either get a $+-$ on the root node and a winning strategy for Louise, or a $-+$ on the root node and a winning strategy for Richard, or a 00 on the root node and a drawing strategy for each player.

1.2 Zermelo's Theorem

In this section, we will prove a famous theorem due to Ernst Zermelo. This theorem tells us that in every combinatorial game, either Louise has a winning strategy, or Richard has a winning strategy, or both players have a strategy to guarantee them a draw. In the process of proving this theorem, we will develop the tools to prove that the working backwards procedure described above really does work as claimed on game trees of all sizes. The proof of Zermelo's Theorem is based on the mathematical principle of induction, so we begin with a brief discussion of this important concept.

Mathematical Induction

Induction is an extremely powerful tool for proving theorems. The simplest proofs by induction are used to prove properties of the nonnegative integers. For instance, suppose that $P(n)$ is a certain property of the number n that we wish to prove holds true for every $n \geq 0$. Since there are infinitely many nonnegative integers, it would be impossible to make a new proof for each individual one! Mathematical induction instead provides a general method to prove that $P(n)$ holds true for every $n \geq 0$.

The inductive approach is to view $P(0)$, $P(1)$, $P(2)$, ... as dominoes. We will think of knocking a domino over as showing that property P holds true for integer n. The proof involves two stages. We first show that the first domino falls over—that is, we prove that $P(0)$ is true. This part is called the *base case*. The second part is to prove that for every $n \geq 1$, if all the dominos before the n^{th} domino fall, then the n^{th} domino also falls. That is, we must prove that if $P(k)$ is true for all $k < n$ (this is the *inductive hypothesis*) then $P(n)$

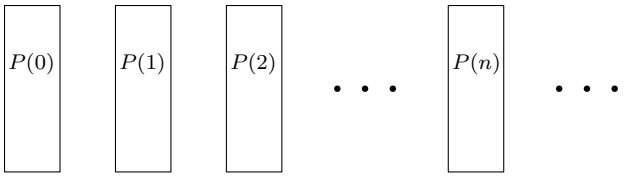


Figure 1.11: A proof by induction

is also true.[2] This part is called the *inductive step*. Of course, if this happens, every domino will be knocked over.

Base Case $P(0)$ is true.

Inductive Step If $P(k)$ is true for all $k < n$ then $P(n)$ is true.

Let's consider a straightforward example that exhibits a nice property of positive integers.

Example 1.9. For every $n \geq 0$, the sum of the first n odd integers is n^2.
For the proof, we proceed by induction on n. To verify the base case, observe that $0^2 = 0$ is the sum of the first 0 odd integers. For the inductive step, let $n \geq 1$ be an arbitrary integer, and assume that our formula holds true for all nonnegative integers less than n. In particular, the formula holds for $n - 1$, which means

$$1 + 3 + 5 + \ldots + (2(n-1) - 1) = (n-1)^2.$$

Starting with this equation and adding $2n - 1$ to both sides gives us

$$1 + 3 + 5 + \ldots + (2n-3) + (2n-1) = (n-1)^2 + (2n-1) = n^2$$

and this completes the proof.

Notice in this example that property $P(n)$ is never formally defined in the proof. Nevertheless, the idea is there: $P(n)$ is the property that the sum of the first n odd numbers is n^2. This proof handles the base case by verifying $P(0)$, or showing that the sum of the first 0 integers is 0^2 (as usual, the base case was the easy part). For the inductive step, we assumed $P(k)$ to be true for all $k < n$ (our inductive hypothesis), and then used this to prove that $P(n)$ holds true for every $n \geq 1$. The inductive hypothesis gave us the advantage of starting with the equation for $P(n-1)$, from which we then deduced $P(n)$.

Mathematical induction enjoys extremely wide application, well beyond proving nice properties of positive integers. Indeed, we will later see more involved instances of induction at work in our investigations of combinatorial games. The general context in which induction applies involves some property P, which we want to show holds true for infinitely many things. To proceed by induction, we will want to organize these things into different *sizes* (i.e. some things have size 0, some have size 1, etc). Now, instead of trying to prove all at

[2]Technically speaking, we are introducing "strong induction" here, since our inductive assumption is that $P(k)$ holds for all $k < n$. In "weak induction" this is replaced by the weaker assumption that $P(n-1)$ holds. These two principles are logically equivalent, but this text frequently utilizes the strong form, so that is what we will adopt throughout.

once that all of these things satisfy property P, we can proceed by induction on the size. The base case will be to prove that P is true for all things of size 0. Then, for the inductive step, we will assume (the inductive hypothesis) that P is true for all things of size less than n and use this to show that P then holds true for an arbitrary thing of size n.

In some cases where induction applies, the smallest relevant size might not be 0, and could instead be 1 or 2 or something else. Most generally, the base case involves proving the result for the smallest size that makes sense in context. The inductive step handles all things of larger size.

Proof of Zermelo's Theorem

With the concept of induction in hand, we are now ready to give a proof of Zermelo's famous theorem. Our proof will rely upon an inductive argument that applies to W-L-D game trees, so we will need to decide upon a "size" for these trees. We will use a quantity called depth as the size of a game tree. The *depth* of a game tree is the maximum number of possible moves from the start to the end of the game (eg. the tree in Figure 1.10 has depth 4). Since every game tree is finite, every game tree has some depth and that depth will always be a non-negative integer. This sets us up to use induction on depth to prove common properties of all game trees.

To prove a property P of trees by induction on depth, we first prove the base case, that P is true for all trees of depth 0. For the inductive step, we need to prove that P holds for an arbitrary tree of depth $n > 0$ under the inductive assumption that P holds true for all trees with depth less than n. Next we'll see this idea in action in the proof of Zermelo's Theorem. This theorem introduces a new definition, the *type* of a W-L-D game tree, and it establishes that every possible W-L-D game tree has one of three types.

Theorem 1.10 (Zermelo)**.** *Every W-L-D game tree is one of*

Type	*Description*
$+-$	*Louise has a winning strategy.*
$-+$	*Richard has a winning strategy.*
00	*Both players have drawing strategies.*

Proof. We proceed by induction on the depth of the game tree. As a base case, observe that if the tree has depth 0 (i.e. it is trivial), then the game is already decided and it is either $+-$ in which case Louise has a winning strategy, $-+$ in which case Richard has a winning strategy, or 00 in which case both players have drawing strategies.

For the inductive step, let T be a W-L-D game tree of depth $n > 0$ and assume the theorem holds for every tree with smaller depth. Suppose that Richard has the first move in T (the case where Louise has the first move follows from a similar argument), and he can move to one of the nodes $N_1, N_2, \ldots, N_\ell$.

We can consider each node N_i as the root of a new W-L-D game tree T_i, consisting of N_i and all the nodes below it. Since each T_i has depth $< n$, our inductive hypothesis tells us that all of these games must satisfy the theorem (i.e. must be type $+-$, $-+$, or 00). So, for every $1 \leq i \leq \ell$ we may choose strategies $\mathcal{L}_i$ for Louise and $\mathcal{R}_i$ for Richard in the game tree T_i with the property that either one of these strategies is winning, or both are drawing.

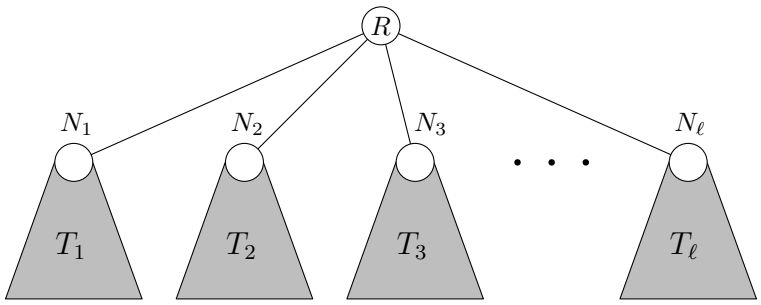


Figure 1.12: The first move in a game tree

We form a strategy $\mathcal{L}$ for Louise in the original game tree by combining $\mathcal{L}_1 \dots \mathcal{L}_\ell$. To form a strategy $\mathcal{R}$ for Richard, we will combine $\mathcal{R}_1 \dots \mathcal{R}_\ell$ but we will also need to make a decision at the root node. Next we split into cases.

Case 1. *At least one of $T_1, \dots, T_\ell$ is type $-+$.*

Let T_i be type $-+$. Then Richard's strategy $\mathcal{R}_i$ is winning in T_i and we may form a winning strategy $\mathcal{R}$ in the original game by having Richard play to node N_i.

Case 2. *All of $T_1, \dots, T_\ell$ are type $+-$.*

In this case, every $\mathcal{L}_i$ strategy is winning, so $\mathcal{L}$ is a winning strategy for Louise.

Case 3. *None of $T_1, \dots, T_\ell$ is type $-+$, but at least one is type 00.*

Let T_i be type 00. Then Richard's strategy $\mathcal{R}_i$ is drawing and we may form a drawing strategy $\mathcal{R}$ in the original game by having Richard play to N_i. Since none of $T_1, \dots, T_\ell$ is type $-+$, each of Louise's strategies $\mathcal{L}_1, \dots, \mathcal{L}_\ell$ is drawing or winning, and it follows that $\mathcal{L}$ is a drawing strategy for Louise. □

So, Zermelo's theorem gives us a classification of game trees into types $+-$, $-+$, and 00. Furthermore, the proof of this result implies that our 'Working Backwards' technique from the previous section will always have one of the following results.

Corollary 1.11. *For every W-L-D game tree applying the Working Backward procedure results in one of the following.*

root label	*result*
$+-$	*A winning strategy for Louise*
$-+$	*A winning strategy for Richard*
00	*Drawing strategies for both players*

For the purposes of analyzing small games, constructing a game tree is a convenient way to determine whether one of the players has a winning strategy, or if they both have drawing strategies. For large games like Chess, it is theoretically possible to construct a game tree[3]. However, the number of positions in Chess has been estimated at approximately 10^{120} while the number of atoms in the universe is around 10^{80}... so our universe isn't big enough for

[3]Although it is possible to repeat positions in a game of chess, there are certain lesser-known rules that make it a finite game.

such analysis! The fact that such a game tree does exist for Chess nevertheless means that Zermelo's Theorem applies to it. So, in Chess, either one of the two players has a winning strategy, or both have drawing strategies... we just don't know which of these it is.

1.3 Strategy

Game trees can be a useful tool to determine who will win when playing from a particular position in a game such as Chop. However, Chop boards come in infinitely many sizes, so there are infinitely many different game trees, and this approach will be limited at best! In this section we will introduce a pair of useful ideas — namely symmetry and strategy stealing — that can help us to determine which player has a winning strategy without having to analyze the game tree. In particular, we will learn exactly which player has a winning strategy in every Chop position. Although these techniques do not apply to all games, they are powerful when they work.

Before we introduce these new ideas, let us pause to discuss how we represent or define a strategy. Suppose that we are interested in playing a certain position in a combinatorial game (with someone going first). To describe a strategy for a player, it is possible to construct the associated game tree and then depict the strategy there. However this is a bit tedious, so it will be helpful for us to adopt a more relaxed treatment. Accordingly, we will generally describe a strategy for a player (in words) by giving a rule that tells this player what to do at each possible position. Given a strategy described in this manner, we could form the game tree and depict it there, but there is generally no need to do this.

Symmetry

The key to finding winning strategies in certain games is symmetry of the positions. Indeed, this simple idea is the key to understanding both Chop and Pick-Up-Bricks. By definition, every position in the game of Chop is a rectangle of the form $m \times n$. Some of these rectangles have the additional symmetry of being square (so $m = n$) and these positions are the key to understanding this game.

Proposition 1.12. *Consider an $m \times n$ position in Chop.*

1. *If $n = m$ the second player has a winning strategy.*
2. *If $n \neq m$ the first player has a winning strategy.*

Proof. First we prove that the second player has a winning strategy whenever the initial position is square. This winning strategy for the second player is easy to describe: On each turn, move the position to a square one. Assuming the second player does this, every time the first player has a move to make (including the first) the position will be a square and any move is to a non-square position. Note that the second player can always move a non-square position to a square one. Assuming this is done, the second player will eventually move to a 1×1 position and win the game.

A similar idea reveals a first-player winning strategy when the initial position is not square. On the first turn, the first player may move the board to a square position. From

there, that player may adopt the above strategy (always moving to a square position). This will guarantee the first player a win. □

In Pick-Up-Bricks, the positions where the number of bricks is a multiple of 3 are the symmetric positions, and they play the same role as the square positions in Chop.

Proposition 1.13. *Consider a Pick-Up-Bricks position of n bricks.*

1. *If 3 divides n, the second player has a winning strategy.*
2. *Otherwise, the first player has a winning strategy.*

Proof. For the first part, the following strategy is winning for the second player: On each turn, do the opposite of the first player's move. So, if the first player picks up one brick, then the second player picks up two, and if the first player picks up two, then the second player picks up one. This ensures that after both players have played, the total number of bricks is three fewer and the new position is again be a multiple of three. Following this, the second player will eventually take the last brick and win.

The first player can win when the starting position is not a multiple of three. To start, the first player may remove either one or two bricks to bring the position to a multiple of three. Now the first player may adopt the second player strategy described above and win from here. □

Strategy Stealing

Strategy stealing is another tool to approach the general question of which player has a winning strategy. Much how symmetry sometimes works well to understand strategy for games, strategy stealing also is very effective when it applies. However, unlike the symmetry arguments that explicitly construct winning strategies, strategy-stealing arguments prove the existence of a winning strategy without giving any indication of what this strategy is! This is because strategy-stealing arguments employ proofs by contradiction.

To create a proof by contradiction, we begin by assuming the opposite of what we are trying to prove. We then argue deductively that a necessary consequence of that assumption is something impossible. It follows that our assumption generating this contradiction must have been false, and therefore, the opposite is true. The strategy stealing arguments that we will introduce here rely on somewhat subtle proofs by contradiction, which will benefit from careful contemplation. Before taking them on, we begin with a warm-up proof by contradiction in the following example.

Example 1.14. Let n be an integer and assume that n^2 is even. Then n must also be even. We will shall prove this (admittedly easy) fact by contradiction. So, our first step will be to assume that n is odd (i.e. assume the negation of what we are trying to prove). Since we are now assuming that n is odd, we may express it as $n = 2k+1$ for another integer k. This gives us

$$n^2 = (2k+1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1.$$

We conclude from the above equation that n^2 is odd, but this contradicts the hypothesis that n^2 is even. So, our initial assumption that n is odd has lead us to a contradiction, and thus we can conclude that n must be even, as desired.

We next use proof by contradiction to determine which player has a winning strategy in Chomp and in Hex. These proofs are called strategy-stealing arguments because we will assume (for a contradiction) that one of the players has a winning strategy, and then the other player will try to steal this strategy to win. Note again that both of these arguments will only prove the existence of a winning strategy. The proofs give no information about the specific moves that could be used to win the game.

Proposition 1.15. *For every rectangular position in Chomp except* 1×1*, the first player has a winning strategy.*

Proof. Chomp cannot end in a draw, so it follows from Zermelo's Theorem that one of the two players has a winning strategy. To prove that it is the first player who has a winning strategy, we will employ a proof by contradiction. For our proof, we will assume the first player does *not* have a winning strategy. It then follows from Zermelo's Theorem that the second player does have a winning strategy, which we will call $\mathcal{S}$. Now, consider what would happen if the first player removed just the upper rightmost square, and then the second player chose her move according to $\mathcal{S}$. The resulting position must be some version of one of the shapes in the figure below.

Figure 1.13: Simple positions in Chomp

This position must be one from which the second player to move has a winning strategy (since the second player followed the winning strategy $\mathcal{S}$ to get here). However, this very same position could also be reached in one step! The first player could have moved the board to this position on their first move. Then the first player would be second to play from this position, and this means that the first player has a winning strategy. This contradicts our assumption that the first player did *not* have a winning strategy. We conclude that our assumption is false, which means this game does indeed have a winning strategy for the first player. □

It turns out that the game of Hex cannot end in a draw. Our next theorem uses this property together with a strategy stealing argument to show that the first player has a winning strategy.

Proposition 1.16. *The first player has a winning strategy in Hex (starting from an empty board).*

Proof. Using the fact that Hex cannot end in a draw, Zermelo's Theorem tells us that one of the two players must have a winning strategy. Let's assume (for the sake of contradiction) that the first player does not have a winning strategy. In this case we may choose a winning strategy $\mathcal{S}$ for the second player.

Now we will take control of the first player and we'll take advantage of the strategy $\mathcal{S}$ to win (this will give us a contradiction, thus proving $\mathcal{S}$ cannot exist). On our first move,

we choose an arbitrary hexagon h and mark it with our symbol $*$. However, we will pretend that the hexagon h has no mark, and that we are the second player. This allows us to adopt the strategy $\mathcal{S}$ to make our moves. By following this winning strategy, we are guaranteed to win in the pretend version of our game. Since the true game just has one extra hexagon marked with $*$, this also gives us a win in the true game.

There is one slight complication we ignored in the above argument, but it isn't hard to fix. Namely, it might be the case that at some point, the strategy $\mathcal{S}$ we are following in the pretend game instructs us to mark the hexagon h with $*$. Since h is already occupied by a $*$ this is not a possible move for us in the true game. In this case, we just choose another unoccupied hexagon h', mark it with $*$ and now pretend that h' is empty. This permits us to keep following the strategy $\mathcal{S}$ in our pretend game. We may later end up with other pretend-empty hexagons h'', h''', and so on. But in any case, it follows from the assumption that $\mathcal{S}$ is a winning strategy that we win our pretend game. Since any win in the pretend game guarantees us a win in the true game, we have constructed a winning strategy for the first player. This contradiction shows that the second player does not have a winning strategy, so the first player does. □

Chapter 2

Normal-Play Games

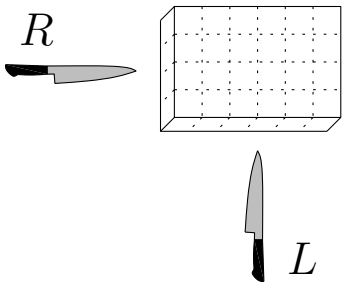


Figure 2.1: A game of Cut-Cake

Combinatorial games encompass a wide assortment of games, including those we played in the previous chapter and many others such as Chess, Go, and Checkers. Many of these games have quite complicated rules and, apart from Zermelo's Theorem, there is little general theory to apply. Normal-play games, however, have some additional structure that facilitates further analysis. Recall that a combinatorial game is normal-play if the win rule dictates that the winner is the last player to move. This win rule gives rise to both a type of addition on normal-play games and a rich notion of equivalence. We have already encountered several normal-play games, namely Pick-Up-Bricks, Chop, and Chomp. Let's play another one now.

Game 2.1 (Cut-Cake)**.** In this normal-play game played between Louise and Richard, each position consists of a collection of uncut pieces of cake. Each uncut piece of cake is rectangular and has dotted lines running horizontally and vertically indicating where it can be cut. On her turn, Louise chooses any piece of cake and makes a vertical cut, while on Richard's turn he chooses any piece and makes a horizontal cut. The last player to make a move wins.

From a position in Cut-Cake, the available moves for Louise and Richard are generally quite different. This is in sharp contrast to the games Pick-Up-Bricks, Chop, and Chomp, where the available moves from every position are always the same for either player. We define a normal-play game to be *impartial* if, from every position, the set of moves available to either player is the same. Otherwise, we say that the game is *partizan*. In the following chapter, we will study impartial games in depth and Chapter 4 will focus on partizan games.

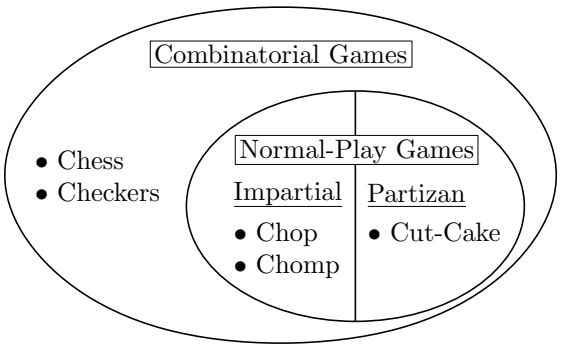


Figure 2.2: A Venn diagram of games

The above Venn Diagram shows our present division of combinatorial games. Note that Chess and Checkers are not normal-play games, since both of these games can end in a draw.

2.1 Positions and their Types

Now we begin our mathematical investigation of normal-play games. We will develop notation to represent positions, introduce a classification of positions into types, and then learn how to determine the type of a given position.

Positions

Normal-play games are generally simple to describe. They consist of a set of positions together with a rule that dictates, for each position, to which positions Louise can move and to which positions Richard can move. That's it! The diagram below depicts an example position in Cut-Cake, together with the available moves for Louise and for Richard.

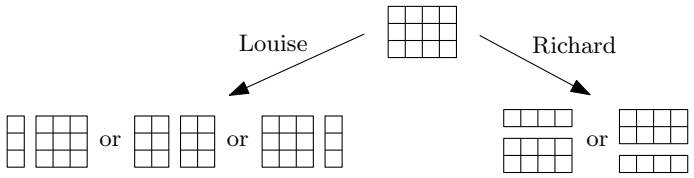


Figure 2.3: Moves for Louise and Richard in a Cut-Cake position

The key property of any position is to which other positions each player can move, so our notation will emphasize this. We will represent a position by a pair of set braces containing all moves Louise can make and then a bar followed by all possible moves Richard can make. So, we would represent the above position in Cut-Cake by:

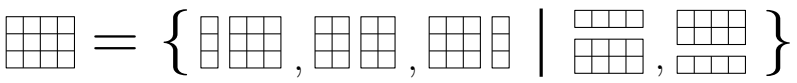


Figure 2.4: Position notation for Cut-Cake

In the diagram above, the two positions to which Richard can move are essentially the same, each consists of one 1×4 piece and one 2×4 piece. Likewise, two of Louise's possible positions consist of one 3×1 piece and one 3×3 piece. Going forward, we will usually simplify such expressions by listing just one of the two.

As a convention, we will generally reserve the equals symbol for use in position notation $\gamma = \{\alpha_1, \ldots, \alpha_m \mid \beta_1, \ldots, \beta_n\}$. If we wish to express that γ and γ' are positions from the same game that are exactly the same, we will say that γ *is identical to* γ'.

Types of Positions

Let us consider playing a normal-play game starting from a given position. If Louise is first to play, then Zermelo's Theorem says either Louise or Richard will have a winning strategy (since normal-play games cannot end in a draw). Similarly, if Richard plays first, then one of our two players is guaranteed to have a winning strategy. Thus, Zermelo's Theorem gives us the following classification of positions in normal-play games into four *types*.[1]

Corollary 2.2. *Every position in a normal-play game is one of*

Type	*Description*
L	**L***ouise has a winning strategy whomever goes first.*
R	**R***ichard has a winning strategy whomever goes first.*
N	*The* **N***ext player to play has a winning strategy.*
P	*The second (or* **P***revious) player has a winning strategy.*

We use familiar games (Pick-Up-Bricks and Cut-Cake) to illustrate one position of each type in the following figure.

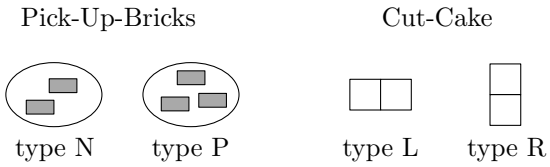


Figure 2.5: A position of each type

Determining Type

The type of a position is its most important characteristic, so we focus on determining it. As before, we rely on principles of recursive analysis, although now the game trees are generally suppressed. Consider a position α in a normal-play game. How might we determine its type? Suppose that Louise will move first from α and that she has a move to a position β of type L or type P. By the definition of type, she has a winning strategy playing second from β, and this gives her a winning strategy playing first from α. Conversely, if every possible move for Louise results in a position of type R or type N, then Richard will have a winning strategy whenever she plays first. A similar analysis for Richard gives the following result.

[1] We have chosen to use the term 'type' instead of the standard term 'outcome class.' This is to keep this concept separate from that of outcome, which is used extensively in the classical game theory part of this book.

Proposition 2.3. *If* $\gamma = \{\alpha_1 \dots \alpha_m \mid \beta_1 \dots \beta_n\}$, *the type of* γ *is given by the following chart.*

	some β_j *is type R or P*	*all of* $\beta_1 \dots \beta_n$ *are types L or N*
some α_i *is type L or P*	N	L
all of $\alpha_1 \dots \alpha_m$ *are types R or N*	R	P

This proposition gives us a straightforward recursive procedure to determine the type of a position. We will demonstrate this procedure by finding the types of some small positions in Cut-Cake. We shall begin with the simplest positions and build up to determine the type of a 2×3 position.

Notice first that whenever a Cut-Cake position has all pieces of size 1×1, neither player has any available moves, so such a position will be type P. Another easily-understood position is one in which just one player has available moves. If Richard can move but Louise cannot, the position is type R. If Louise can move but Richard cannot, the position is type L. So, in particular, we can determine the types of the positions in Figure 2.6.

Figure 2.6: Cut-Cake positions where only one player can move

Let's determine the types of a couple more simple positions. In each of the positions in Figure 2.7, whoever moves first puts the game in a position from which only their opponent can move. So, Richard can only bring the position to one of type L, whereas Louise can only bring the position to one of type R. It's easy to see that any such position will be type P (you may also deduce this by applying Proposition 2.3). Since 1×1 pieces are irrelevant in Cut-Cake positions, the position notation ignores them for simplicity.

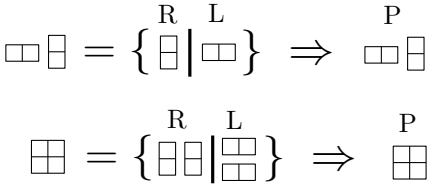


Figure 2.7: Types of some Cut-Cake positions

In Figure 2.8, Proposition 2.3 is used to complete our example. We have determined that a 2×3 position in Cut-Cake has type P. More generally, anytime we want to determine the type of a position in a normal-play game, we can use Proposition 2.3 to work backward.

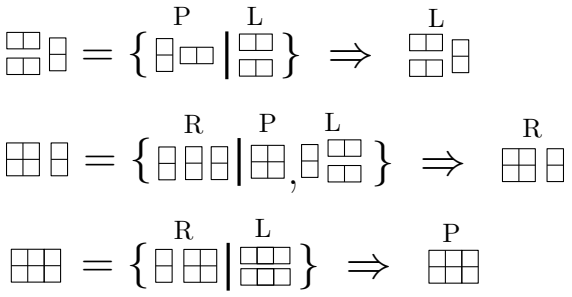


Figure 2.8: Types of more Cut-Cake positions

2.2 Sums of Positions

This section continues to develop our algebraic treatment of games. We will define a sum operation on normal-play games and begin to explore the properties of positions under this operation. This is a significant departure from the usual way we think about games – we will now treat positions like numbers.

Sums

In fact, we have already seen sums in the game of Cut-Cake. Even if we begin with a single piece of cake, it will break into many pieces over the course of the game. If we call each piece a component, then we can view a Cut-Cake position as a sum of these components. Each player, in turn, must select one component in which to move.

Although we will continue to focus exclusively on normal-play games, it is worth noting that a similar phenomena is encountered in some well-known (non-normal-play) combinatorial games. For instance, in the late stages of a game of Go, the board will be divided into many small regions, and on each turn a player chooses one of these "components" in which to move.

We next introduce a general notion of sum that allows addition of positions even from different games.

Definition 2.4. If α and β are positions in normal-play games, then we define $\alpha + \beta$ to be a new position consisting of the *components* α and β. To move in $\alpha + \beta$, a player chooses one of the components in which to move. So, for instance if a player may move from α to α', then the player may move from $\alpha + \beta$ to $\alpha' + \beta$. Similarly, if a player can move from β to β' then he or she may move from $\alpha + \beta$ to $\alpha + \beta'$.

Example 2.5. Figure 2.9 exhibits a sequence of three moves in a position that is the sum of two positions, one from Cut-Cake and the other from Pick-Up-Bricks. As you can see, Louise chooses to play from the Pick-Up-Bricks component for her first move. The next two moves are both made in the Cut-Cake component.

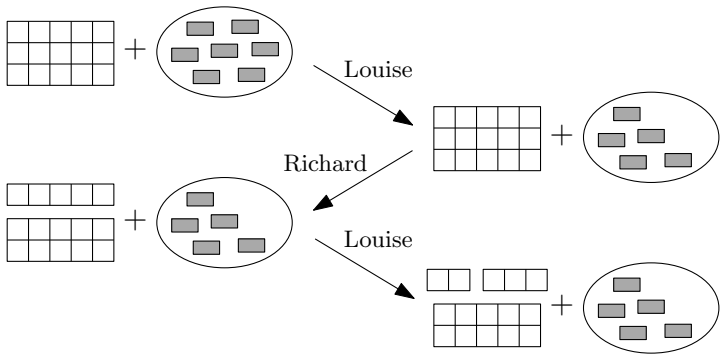


Figure 2.9: Three moves in a sum of Cut-Cake and Pick-Up-Bricks

Determinate Sums

Now that we can add positions, we want to understand strategy in sums of positions. More precisely, we will be interested in understanding the behaviour of our types (L, R, P, N) under sums. First we will consider some instances where we can determine the type of a sum based on the types of its components.

Let's imagine that we have a particular position α that is type R, so as first player, Richard has a winning strategy $\mathcal{S}$. What would you advise Richard to do if he and Louise were to play the following position with Richard going first?

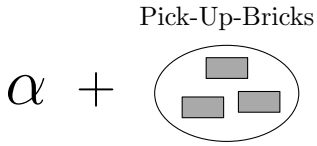


Figure 2.10: What should Richard do?

In fact, the component consisting of three bricks in Pick-Up-Bricks doesn't change things significantly. Richard can just ignore this component, and follow his strategy $\mathcal{S}$, pretending just to play against Louise in the position α. At some point, Louise may choose to pick up either 1 or 2 bricks for a move in the Pick-Up-Bricks component. At that point, Richard should respond to Louise by making the opposite play in the Pick-Up-Bricks component. This sequence would cost Louise and Richard one move each, but would return the game to the α component. From there, Richard can continue to follow the strategy $\mathcal{S}$, which guarantees him a win. The next proposition replaces this Pick-Up-Bricks position with an arbitrary position of type P to generalize the result.

Proposition 2.6. *If β is type P then α and $\alpha + \beta$ are the same type.*

Proof. To prove this, we will need to show that whenever a certain player (Louise or Richard) has a winning strategy moving either first or second in the game α, then the same holds true in the game $\alpha + \beta$. We will show below that whenever Louise has a winning strategy as second player in α, then she also has a winning strategy as the second player in $\alpha + \beta$. The other cases are similar.

Let $\mathcal{S}$ be a winning strategy for Louise playing second in α and let $\mathcal{T}$ be a second player winning strategy in β. Here is Louise's strategy for playing $\alpha + \beta$ as second player: if Richard moves in component α, then Louise follows strategy $\mathcal{S}$ to respond in this component; if Richard moves in the component β, then Louise follows strategy $\mathcal{T}$ to respond in β. Since the strategies $\mathcal{S}$ and $\mathcal{T}$ guarantee Louise that she will make the last move, it follows that Louise will make the last move while following this strategy in $\alpha + \beta$, thus guaranteeing her a win. □

A similar argument establishes the following.

Proposition 2.7. *If α and β are both type L then $\alpha + \beta$ is type L. Similarly, if α and β are both type R then $\alpha + \beta$ is type R.*

Our current state of knowledge concerning the behaviour of types under summation is given by the following table.

+	L	R	N	P
L	L	?	?	L
R	?	R	?	R
N	?	?	?	N
P	L	R	N	P

Figure 2.11: Types of sums

It turns out that none of the question marks in the above chart can be truthfully replaced by one of our types. For instance, if we know that α is type R and β is type N, it is not in general possible to determine the type of $\alpha + \beta$. In the next part of this section we will introduce a new game to demonstrate this.

Indeterminate Sums

To assist us in exploring sums, we now introduce a game called Domineering.

Game 2.8 (Domineering)**.** This is a normal-play game played using some squares from a rectangular array. On Louise's turn, she may place a 2×1 domino over two unoccupied squares. On Richard's turn, he may place a 1×2 domino over two unoccupied squares. Since this is a normal-play game, the last player to move wins.

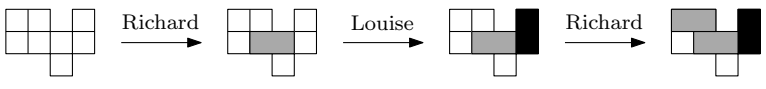


Figure 2.12: A sequence of moves in Domineering

It is most appealing to play domineering with actual dominoes, but for the purposes of calculation, it is easier to imagine that the players move by deleting squares (so Louise deletes

two vertically adjacent squares while Richard deletes two horizontally adjacent squares). We can treat disconnected parts of the grid separately, and may ignore any 1×1 component. This allows us to view the game from Figure 2.12 as seen in Figure 2.13.

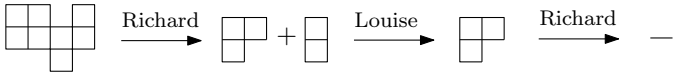


Figure 2.13: Domineering notation

Consider the simple Domineering positions in Figure 2.14. The type of each of these positions is straightforward to verify: The position on the left is type N since both players have an available move, but after one player makes a first move, the other player has no move. The other positions are type R since Richard can move, but Louise cannot.

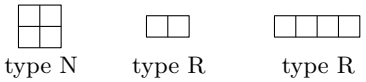


Figure 2.14: Domineering positions

Now we will consider sums of positions from Figure 2.14. Were Richard to play first in the Figure 2.15 position, he could win by playing in the left component (after which Louise would have no move available). On the other hand, if Louise were to play first, then she would play in the left component, and then win the game. It follows that this position is type N.

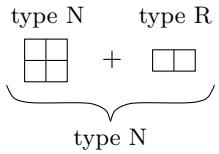


Figure 2.15: An N + R position that is type N.

Nevertheless, not every N + R position is type N. Both a 1×2 array in Domineering and a 1×4 array are type R, but the 1×4 position gives Richard a greater advantage than the 1×2 position. In fact, this difference is enough so that the sum from Figure 2.16 behaves differently from the sum in Figure 2.15.

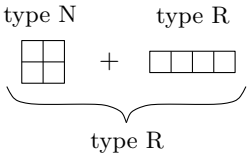


Figure 2.16: An N + R position that is type R

In the above position, Richard can win as first player by moving first in the 2×2 component; after that Louise has no available move so Richard wins. On the other hand, if

Louise moves first she must move in the 2×2 component; after this Richard can move in the 1×4 component so as to leave himself another move. Assuming he does so, he will win. We conclude that this position is type R.

Now we have seen an N + R position of type N and an N + R position of type R. This example illustrates it is not always possible to determine the type of a sum from the types of its components. In fact, all of the other question marks in Table 2.11 are likewise ambiguous. In the next section we will introduce a finer notion than that of type that will allow us better to understand sums.

2.3 Equivalence

Consider the following two positions, one from Domineering and one from Pick-Up-Bricks.

Figure 2.17: Two different positions that play the same

As you can see, the player who moves first in either α or α' will leave the opponent with no available moves. So, although α and α' are from two completely different games, the positions are very much the same. Moreover, if we were to take any other position β, then the positions $\alpha + \beta$ and $\alpha' + \beta$ would play similarly, too.

We would like our theory to express this intuition that the positions α and α' are essentially the same. With a meaningful notion of equivalence that allows us to assert that two positions (possibly in different games) are strategically the same, we can move beyond investigating one game at a time. We will be able to identify similar strategic features in different games and prove broad theorems that apply to many games at once.

To do this, we will need the following definition of equivalence for two positions. Although this definition is straightforward to state, the reader should be cautious of its subtleties.

Definition 2.9. We say that two positions α and α' in (possibly different) normal-play games are *equivalent* if for every position β in any normal-play game, the two positions $\alpha + \beta$ and $\alpha' + \beta$ have the same type. We write $\alpha \equiv \alpha'$ to indicate this.

In other words, two positions are equivalent if they behave the same under summation. Our positions α and α' from Figure 2.17 are indeed equivalent. To prove this, we need to show that for every position β, the positions $\alpha + \beta$ and $\alpha' + \beta$ have the same type. This is easy to check, since one may immediately translate a winning strategy from one of these positions into one for the other.

Be aware that this instance of equivalence is a very simple one. In fact, there are many other positions that are also equivalent to α and α', many of which have lots of available moves for either player. The power of equivalence is that it allows us to treat all of these positions as the same.

Equivalence Relations

There are some fundamental properties of equivalence that follow directly from our definitions. The next proposition features three important properties known as reflexivity, symmetry, and transitivity. We leave it to the reader to verify these.

Proposition 2.10. *If α, β, γ are positions in normal-play games,*

1. *$\alpha \equiv \alpha$ (reflexivity),*
2. *$\alpha \equiv \beta$ implies $\beta \equiv \alpha$ (symmetry),*
3. *$\alpha \equiv \beta$ and $\beta \equiv \gamma$ implies $\alpha \equiv \gamma$ (transitivity).*

The term *equivalence relation* is used to describe any relation that is reflexive, symmetric and transitive. A few relatively well-known equivalence relations include similarity for triangles, congruence modulo n for the integers, and row equivalence for matrices. As the term "equivalence relation" suggests, these properties behave much like equals does for numbers. Indeed, $=$ is an equivalence relation — one of fundamental importance.

In addition to the equivalence relation on positions defined above, we have already implicitly introduced two other equivalence relations for positions. Specifically, note that — although we have not given them symbols — the notions "is identical to" and "is the same type as" are also reflexive, symmetric, and transitive and thus also give equivalence relations for positions.

We now have three equivalence relations for positions. How do they compare with one another? If two positions α and β are identical, then it is straightforward to check that they must be equivalent and they must also have the same type. So the relation "is identical to" implies both of the other equivalence relations. Next we will investigate the relationship between equivalence and type.

Equivalence vs. Type

It is a direct consequence of the definitions that anytime two positions are equivalent, they must have the same type. This is formalized in the following proposition.

Proposition 2.11. *If $\alpha \equiv \alpha'$, then α and α' have the same type.*

Proof. Let β be a position in a normal-play game with no moves left for either player (for instance β could be a position in Pick-Up-Bricks with 0 bricks). We claim that the following holds.

$$\alpha \quad \overset{\text{same}}{\underset{\text{type}}{\longleftrightarrow}} \quad \alpha+\beta \quad \overset{\text{same}}{\underset{\text{type}}{\longleftrightarrow}} \quad \alpha'+\beta \quad \overset{\text{same}}{\underset{\text{type}}{\longleftrightarrow}} \quad \alpha'$$

Since β has no moves, it is immediate that α and $\alpha+\beta$ have the same type. Similarly, $\alpha'+\beta$ and α' have the same type. To complete the argument we need to show that $\alpha+\beta$ and $\alpha'+\beta$ have the same type. This follows from the assumption $\alpha \equiv \alpha'$. □

Could there be two positions that have the same type but are not equivalent? To answer this question, consider the following Domineering sums from the previous section.

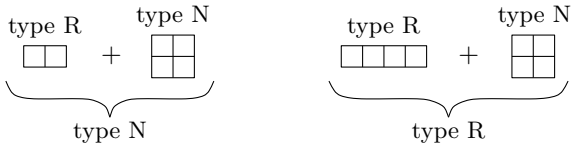


Figure 2.18: Two sums in Domineering with different type

In Figure 2.18, we see that the sum of a 1×2 array and 2×2 array in Domineering has a different type than the sum of a 1×4 array and a 2×2 array. Therefore, a 1×2 array and a 1×4 array are not equivalent, even though they are both type R.

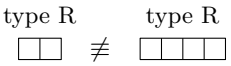


Figure 2.19: Inequivalent Domineering positions of the same type

In short, equivalent positions must have the same type, but not all positions of the same type are equivalent. We have just exhibited two inequivalent positions of type R. Similar arguments show that there exist pairs of inequivalent positions of types L and N. In contrast to this, we shall prove at the end of this section that any two positions of type P are equivalent. In preparation for that, we will need to establish some basic properties of $+$ and $\equiv$.

The Algebra of Sums and Equivalence

In this part of the section we will prove a handful of properties which demonstrate that positions (with $+$ and $\equiv$) behave very much like numbers (with $+$ and $=$). The first such result follows immediately from our definitions, so we leave the reader to verify it.

Proposition 2.12. *If α, β, γ are positions in normal-play games,*

1. $\alpha + \beta \equiv \beta + \alpha$ *(commutativity)*
2. $(\alpha + \beta) + \gamma \equiv \alpha + (\beta + \gamma)$ *(associativity)*

When working with numbers, we know that anytime we have two expressions that give equal numbers, we can substitute one for the other in an equation. Next we establish a similar result for positions.

Lemma 2.13. *All α and β terms in the following expressions are assumed to be positions in normal-play games.*

1. *If $\alpha \equiv \alpha'$ then $\alpha + \beta \equiv \alpha' + \beta$.*
2. *If $\alpha_i \equiv \alpha_i'$ for $1 \leq i \leq n$ then $\alpha_1 + \ldots + \alpha_n \equiv \alpha_1' + \ldots + \alpha_n'$.*

3. *If* $\alpha_i \equiv \alpha_i'$ *for* $1 \leq i \leq m$ *and* $\beta_j \equiv \beta_j'$ *for* $1 \leq j \leq n$, *then* $\{\alpha_1, \ldots, \alpha_m \mid \beta_1, \ldots, \beta_n\} \equiv \{\alpha_1', \ldots, \alpha_m' \mid \beta_1', \ldots, \beta_n'\}$.

Proof. To prove part (1), let γ be an arbitrary position in a normal-play game. We must show that the positions $(\alpha+\beta)+\gamma$ and $(\alpha'+\beta)+\gamma$ have the same type. By associativity, the first of these positions is equivalent to $\alpha+(\beta+\gamma)$ and the second equivalent to $\alpha'+(\beta+\gamma)$. It now follows from the assumption $\alpha \equiv \alpha'$ that these two positions have the same type.

For part (2), we repeatedly apply part (1) as follows

$$\begin{aligned} \alpha_1 + \ldots + \alpha_n &\equiv \alpha_1 + \ldots + \alpha_{n-2} + \alpha_{n-1} + \alpha_n' \\ &\equiv \alpha_1, + \ldots + \alpha_{n-2} + \alpha_{n-1}' + \alpha_n' \\ &\vdots \\ &\equiv \alpha_1' + \ldots + \alpha_n' \end{aligned}$$

We omit the proof of Part (3). □

As the lemmas in this part of the section demonstrate, positions really do behave like numbers!

Type P

Positions of type P play a very special role in the world of normal-play games. In fact, these positions behave under sums just like the number 0 does under addition. As we prove next, whenever a position of type P is added to a position α, the resulting position is still equivalent to α (just as $a+0=a$).

Lemma 2.14. *If* β *is type P, then* $\alpha+\beta \equiv \alpha$.

Proof. Let γ be an arbitrary position in a normal-play game. By Proposition 2.6, we find that $\alpha+\beta+\gamma$ and $\alpha+\gamma$ have the same type. It follows that $\alpha+\beta \equiv \alpha$, as desired. □

Below, we prove that any two positions of type P are equivalent. Recall this differs from the behavior of positions of type N, L, and R.

Proposition 2.15. *If* α *and* α' *are type P, then* $\alpha \equiv \alpha'$.

Proof. For every position γ, the previous lemma gives us

$$\alpha+\gamma \equiv \gamma \equiv \alpha'+\gamma.$$

Since $\alpha+\gamma$ and $\alpha'+\gamma$ are equivalent, they have the same type. This holds for every position γ, so we have $\alpha \equiv \alpha'$ as desired. □

The final result in this chapter gives another special property of type P positions analogous to the number zero. The next lemma parallels for positions the following simple property for numbers: If $a+b=0$ and $a'+b=0$, then $a=a'$.

Lemma 2.16. *If* $\alpha+\beta$ *and* $\alpha'+\beta$ *are both type P, then* $\alpha \equiv \alpha'$.

Proof. Lemma 2.14 implies $\alpha \equiv \alpha + (\alpha' + \beta) \equiv \alpha' + (\alpha + \beta) \equiv \alpha'$. □

At this point we have established the basic principles of type, sums, and equivalence for normal-play games. We have begun to think of positions in a manner similar to numbers, and we have learned that type P positions (which are all equivalent to one another) behave like the number 0 does. In the next two chapters we will call upon these ideas to delve more deeply into the structure of impartial and partizan games.

Chapter 3

Impartial Games

This chapter focuses on impartial games (recall that these are normal-play games in which the available moves for Louise and Richard are always the same). Although these games appear simple in form, they exhibit fascinating structure. Our central result, the Sprague-Grundy Theorem, will provide a thorough understanding of equivalence for impartial games. The proof of this well-known theorem was foundational to the subject of combinatorial game theory.

The particular impartial game Nim will be of central importance in our investigation. Let's get started by playing it!

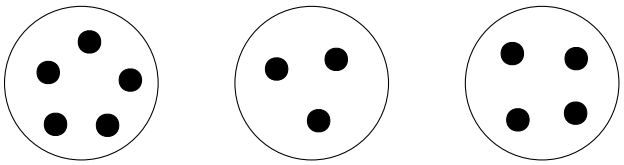


Figure 3.1: Play Nim!

Game 3.1 (Nim)**.** This is an impartial game in which a position consists of ℓ piles of stones of sizes $a_1, a_2, \ldots, a_\ell$. To make a move, a player removes one or more (up to all) stones from a chosen pile. The last player to take a stone wins.

In an impartial game, the identity of the players does not affect the available moves. So if, for instance, Richard has a winning strategy as the first player, then Louise will also have a winning strategy as the first player. This means impartial games have no positions of Type L or type R, so Corollary 2.2 gives us the following.

Corollary 3.2. *Every position in an impartial game is either*

Type N	*The next player to play has a winning strategy,*
	or
Type P	*The second (or previous) player has a winning*
	strategy.

In the previous chapter, we established Proposition 2.3 to determine the type of a position γ from the types of the possible positions to which Louise and Richard can move from γ.

Incorporating the fact that all impartial game positions have type N or P yields the following essential corollary.

Corollary 3.3. *A position in an impartial game is*

Type N *If there exists a move to a position of type P*
Type P *If there is no move to a position of type P*

Can you determine the type of a Nim position with two equal piles? What about a Nim position with two unequal piles?

3.1 Nim

In this section, we will follow the work of Charles Bouton to develop the theory of Nim. In particular, we will determine the type of every position in Nim and learn to implement winning strategies. Then we will prove a theorem which tells us exactly when two Nim positions are equivalent. Much of this analysis is based upon properties of powers of two, so that is where we begin.

Binary Expansion

Here we will establish the basic properties of binary expansion necessary for our analysis of Nim. We begin with a simple but crucial fact.

Proposition 3.4. *For every nonnegative integer ℓ,*

$$2^0 + 2^1 + 2^2 + \ldots + 2^\ell = 2^{\ell+1} - 1.$$

Proof. This follows from the equation

$$\begin{aligned} 2^0 + 2^1 + \ldots + 2^\ell &= 2\big(2^0 + 2^1 + \ldots + 2^\ell\big) - \big(2^0 + 2^1 + \ldots + 2^\ell\big) \\ &= 2^1 + 2^2 + \ldots + 2^{\ell+1} - 2^0 - 2^1 - \ldots - 2^\ell \\ &= 2^{\ell+1} - 2^0. \square \end{aligned}$$

For a nonnegative number n, the *binary expansion* of n is a representation of n as a sum of distinct powers of two. To construct the binary expansion of n, repeatedly take out the largest possible power of two.

Example 3.5. Here is the binary expansion of 45:

$$\begin{aligned} 45 &= 32 + 13 \\ &= 32 + 8 + 5 \\ &= 32 + 8 + 4 + 1. \end{aligned}$$

Next, a formal proof that binary expansions always exist and are unique.

Proposition 3.6. *Every nonnegative integer n has a unique binary expansion.*

Proof. We proceed by induction on n. The base case $n = 0$ holds because 0 is (uniquely expressed as) a sum of no powers of 2. For the inductive step, assume $n > 0$ and that the every integer smaller than n has a unique binary expansion. Choose k so that $2^k \leq n < 2^{k+1}$. No power of two greater than 2^k can appear in a binary expansion of n since it would already be too big. The biggest number we can represent without 2^k is $1+2+\ldots+2^{k-1} = 2^k - 1 < n$. Therefore, every binary expansion of n must consist of 2^k plus a representation of $n' = n - 2^k$. By induction n' has a unique binary expansion and we thus conclude that n has a unique binary expansion. □

With this basic understanding of binary expansion, we will turn our attention to positions in Nim .

Types of Positions

Let's introduce some notation to work with Nim positions. For any nonnegative integer a let $*a$ denote a Nim position consisting of a single pile of a stones. Then a position with piles of sizes $a_1, a_2, \ldots, a_\ell$ may be denoted in our sum notation as $*a_1 + *a_2 + \ldots + *a_\ell$. Figure 3.2 depicts the position $*11 + *13 + *7 + *12$.

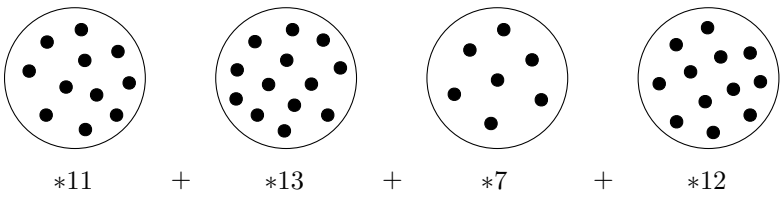


Figure 3.2: A position in Nim

Now we will break each pile into subpiles based on the binary expansion of the size of the pile. For example, since $13 = 8 + 4 + 1$, we will divide the pile of size 13 into subpiles with sizes $8, 4$, and 1.

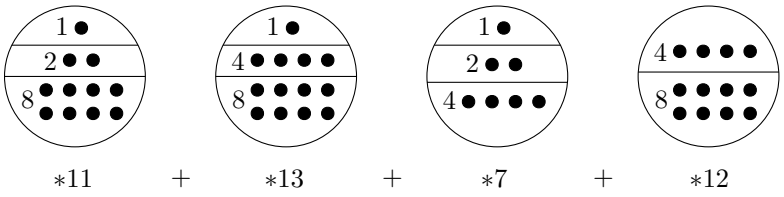


Figure 3.3: Division into subpiles

Now we are ready to introduce a crucial concept in Nim. Define a position $*a_1 + *a_2 \ldots + *a_k$ to be *balanced* if, for every power of 2, the total number of subpiles of that size is even. For example, the position depicted in Figure 3.3 is unbalanced since there are three subpiles of size 8. In contrast, the position $*3 + *6 + *5$ is balanced since $*(1+2) + *(2+4) + *(1+4)$ has two subpiles of sizes 1,2, and 4 and zero subpiles of every other size. The ability to move from an unbalanced position to a balanced position figures prominently in a winning Nim strategy.

Procedure 3.7 (Balancing an Unbalanced Position)**.** Let $*a_1 + \ldots + *a_k$ be an unbalanced game of Nim. Suppose 2^m is the largest power of 2 for which there are an odd number of subpiles of that size. Choose a pile, say $*a_i$, which has a subpile of size 2^m. Now (temporarily – this isn't the move!) pick up all subpiles of this pile which have size less than or equal to 2^m. At the moment, there are an even number of subpiles of size 2^j for every $j \geq m$. Now, for every 2^j with $j < m$, if the number of subpiles of size 2^j is odd, return 2^j stones. Since at least 2^m stones were initially picked up and at most $2^0 + 2^1 + \ldots + 2^{m-1} < 2^m$ stones were put back, this is indeed a legal move. It is straightforward to verify that the resulting position is balanced.

Let's apply the balancing procedure to the position depicted in Figures 3.2 and 3.3. Here, 8 is the largest power of two appearing an odd number of times, so we can move on any pile with a subpile of size 8. Let's choose the first one. Now we pick up the subpile of size 8 and all smaller subpiles (in this case, the entire pile). At this point, there remain an even number of subpiles of sizes 8 and 1, but an odd number of subpiles of sizes 2 and 4. To return to a balanced position, we put back subpiles of sizes 2 and 4. This moves the game to the following balanced position.

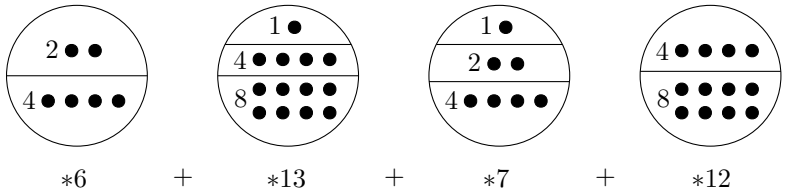


Figure 3.4: A balanced position

Proposition 3.8. *Every balanced Nim position is type P and every unbalanced Nim position is type N. In both cases, the balancing procedure provides a winning strategy.*

Proof. We start by showing that every balanced position is type P. To do so, consider an arbitrary balanced position. Let's adopt the role of the second player and show that the balancing procedure gives us a winning strategy. Any move the first player makes will leave an unbalanced position since that player alters just one pile, which, consequently, will divide into different subpiles. On our turn as the second player, then, we may apply the balancing procedure and return the position to a balanced one. After the first player moves again, the position will again be unbalanced. In fact, this pattern will continue. After each move the first player makes the position will be unbalanced and we, as the second player, can again move to make the position balanced. The game-ending position with no stones is balanced. It follows that we will be the player to move to that position, so we have a winning strategy and every balanced position is type P.

Starting from an unbalanced position, the first player may apply the balancing procedure to move the position to one of type P. Then the above strategy gives the first player a winning strategy, so every unbalanced position is type N. □

This proposition gives us a simple classification of Nim positions by type. In answer to a question from the start of this chapter, we find that a Nim position with two piles $*a + *b$ will

be type P if $a = b$ and type N otherwise. Now that we understand types of Nim positions, our next goal will be to understand equivalence of Nim positions.

Equivalence

Moving forward, it will be helpful to think about positions in Nim like numbers and, to this end, we will call a position of the form $*a$ a *nimber*. We determined above which Nim positions are type P and which are type N. More precisely, we showed that balanced positions (those with an even number of subpiles of each size) are type P and all other positions are type N. This means $*0$ is type P and, since any two type P positions are equivalent, this tells us that every balanced position is equivalent to $*0$. In this section, our consideration of equivalence for general Nim positions will lead to a proof of the surprising fact that every general Nim position is equivalent to a single nimber!

Let's consider a Nim position given by $*a_1 + *a_2 + \ldots + *a_\ell$ from the perspective of the second player. Imagine that before the start of play the second player could add one entirely new pile, say $*b$, to the position. The second player would want the new position $*a_1 + \ldots + *a_\ell + *b$ to be balanced so that they would have a winning strategy. So the pile $*b$ should have a subpile of size 2^j if and only if there are an odd number of subpiles of that size in $*a_1, \ldots, *a_\ell$. This gives rise to a new type of summation as follows.

Definition 3.9. The *Nim-sum* of the nonnegative integers $a_1, \ldots, a_\ell$, denoted $a_1 \oplus \ldots \oplus a_\ell$, is the nonnegative integer b with the property that 2^j appears in the binary expansion of b if and only if this term appears an odd number of times in the expansions of $a_1, \ldots, a_\ell$.

To compute b, find the binary expansion of each a_i and then cross off equal terms in pairs. When there are no more pairs of equal terms, the sum of the remaining terms is b. Here is an example:

$$\begin{aligned} 13 \oplus 19 \oplus 10 &= (8+4+1) \oplus (16+2+1) \oplus (8+2) \\ &= (\cancel{8}+4+\cancel{1}) \oplus (16+\cancel{2}+\cancel{1}) \oplus (\cancel{8}+\cancel{2}) \\ &= 4+16 \\ &= 20. \end{aligned}$$

We are now ready to prove the main result from this section.

Theorem 3.10. *If $a_1, \ldots, a_\ell$ are nonnegative integers and $b = a_1 \oplus a_2 \ldots \oplus a_\ell$ then*

$$*a_1 + *a_2 \ldots + *a_\ell \equiv *b.$$

Proof. Consider the game $*a_1 + *a_2 \ldots + *a_\ell + *b$. It follows from the above definition of Nim-sum that this game is balanced. Therefore, it is type P and equivalent to $*0$. This gives us $*a_1 + *a_2 \ldots + *a_\ell + *b \equiv *0$. Together with the fact that $*b + *b \equiv *0$, this gives us

$$\begin{aligned} *a_1 \ldots + *a_\ell &\equiv *a_1 + \ldots + *a_\ell + *0 \\ &\equiv *a_1 + \ldots + *a_\ell + *b + *b \\ &\equiv *0 + *b \\ &\equiv *b. \square \end{aligned}$$

This result may be viewed as a refinement of Proposition 3.8. Every Nim position $*a_1 + \ldots + *a_\ell$ is equivalent to a nimber $*b$. If $*a_1 + \ldots + *a_\ell$ is balanced, then $b = 0$, and the position is type P. Otherwise, $*a_1 + \ldots + *a_\ell$ is unbalanced, so $b \neq 0$ and the position (which is equivalent to $*b$) will be type N.

We can now view all Nim positions according to their equivalent nimber as in the following figure.

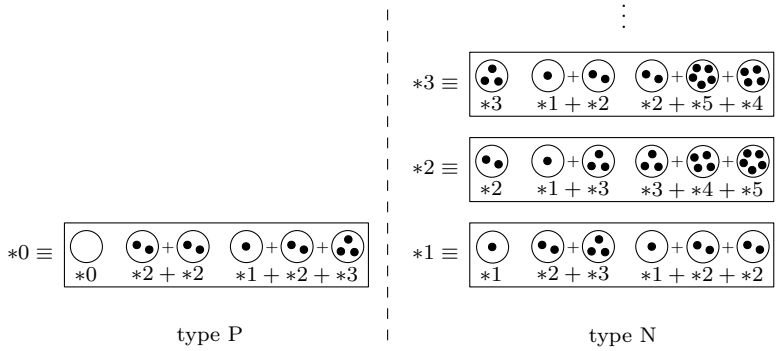


Figure 3.5: Equivalence of Nim positions

3.2 The Sprague-Grundy Theorem

We showed above that every Nim position is equivalent to a nimber. In fact, every position in every impartial game is equivalent to a nimber! In this section we give a proof of this surprising result.

First, we introduce some simplified notation for working with positions in impartial games. For general normal-play games, we denote a position α by $\alpha = \{\beta_1, \ldots, \beta_\ell \mid \gamma_1, \ldots, \gamma_m\}$ if Louise can move to one of $\beta_1, \ldots, \beta_\ell$ and Richard can move to one of $\gamma_1, \ldots, \gamma_m$. The moves for both players are always the same in impartial games, so this notation is redundant. Accordingly, if α is a position in an impartial game, and $\alpha_1, \ldots, \alpha_k$ are the moves available to either player, then we write $\alpha = \{\alpha_1, \ldots, \alpha_k\}$. For a Nim example, we have $*4 = \{*0, *1, *2, *3\}$. Figure 3.6 depicts a representation of a 2×4 position in Chop.

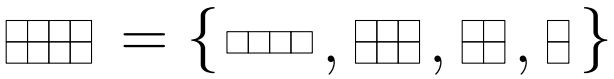


Figure 3.6: Notation for a position in Chop

Definition 3.11. For a set $S = \{a_1, a_2, \ldots, a_n\}$ of nonnegative integers, we define the **M***inimal* **EX***cluded value*, abbreviated *MEX*, of S to be the smallest nonnegative integer b which is not one of $a_1, \ldots, a_n$. So, for instance, the MEX of the set $\{0, 1, 2, 5, 8\}$ is 3.

Theorem 3.12 (The MEX Principle)**.** *Let $\alpha = \{\alpha_1, \ldots, \alpha_k\}$ be a position in an impartial game. Suppose that*

$$\alpha_i \equiv *a_i \text{ for every } 1 \leq i \leq k.$$

Then $\alpha \equiv *b$ where b is the MEX of the set $\{a_1, \ldots, a_k\}$.

Proof. If we can prove that $\alpha + *b$ is of type P, then $\alpha + *b \equiv *0$ and then adding $*b$ to both sides gives the desired result $\alpha \equiv *b$. To complete the proof, then, it will suffice to show that $\alpha + *b$ is type P. To do this, we need to show that the second player can always win from this position.

First, suppose that the first player moves the component $*b$ to $*b'$ for some $b' < b$. Since b is the MEX of the set $\{a_1, \ldots, a_k\}$ there must exist $1 \leq i \leq k$ so that $a_i = b'$. Now the second player may move to bring the component α to α_i. The position of the entire game is now $*b' + \alpha_i \equiv *b' + *a_i \equiv *0$. Therefore this position is type P, and the second player has a winning strategy from here.

Next, suppose that the first player moves the α component to α_j. After this move the position is $*b + \alpha_j \equiv *b + *a_j$. Since $a_j \neq b$ (this follows from the fact that b is the MEX of $\{a_1, \ldots, a_k\}$), this position is equivalent to the nonzero nimber $*(b \oplus a_j)$. It follows from this that the second player has a winning strategy, and this completes the proof. □

Given the MEX Principle, we need just one additional definition for the main result of this chapter, the Sprague-Grundy Theorem. For a position α in an arbitrary combinatorial game, we define the *depth* of α to be the maximum number of moves before α reaches a terminal position, where any sequence of moves is permitted (i.e. instead of alternating, either Louise or Richard is permitted to make multiple moves in a row). As usual, we will assume that all positions have finite depth. Just as we used the depth of a game tree to prove theorems about these trees by induction, we will use the depth of a position in our inductive proof of the following theorem.

Theorem 3.13 (Sprague-Grundy)**.** *Every position in an impartial game is equivalent to a nimber.*

Proof. Let α be a position in an impartial game. We will prove this theorem by induction on the depth of α. As a base case, if α has depth 0, then $\alpha \equiv *0$ since our game is normal-play. For the inductive step, suppose that α has depth > 0 and assume that the theorem holds for every position of depth smaller than α. Using our notation, $\alpha = \{\alpha_1, \ldots, \alpha_\ell\}$. Each of the positions α_i has depth less than that of α. By induction, then, every α_i is equivalent to $*a_i$ for some a_i. Take b to be the MEX of $\{a_1, \ldots, a_\ell\}$ and apply the MEX Principle to conclude

$$\alpha = \{\alpha_1, \ldots, \alpha_\ell\} \equiv \{*a_1, \ldots, *a_\ell\} \equiv *b.$$

This proves that α is equivalent to a nimber. □

The Sprague-Grundy Theorem gives us a thorough understanding of equivalence in impartial games. Furthermore, its proof gives an algorithm for finding nimber equivalents for any basic position. The key idea is to work recursively using the MEX principle. To find a nimber equivalent to the position α, we first need to find nimber equivalents for the positions to which we can move from α. So, if $\alpha = \{\alpha_1, \ldots, \alpha_\ell\}$ and we have nimbers $*a_i$ so that $\alpha_i \equiv *a_i$ for $1 \leq i \leq \ell$ then α will be equivalent to $*b$ where b is the MEX of $\{a_1, \ldots a_\ell\}$. The next section gives several examples of this procedure in action.

3.3 Applying the MEX Principle

So far, we have encountered the impartial games Nim, Pick-Up-Bricks, Chop, and Chomp. The Sprague-Grundy Theorem tells us that every position in every one of these games is equivalent to some nimber, and the MEX Principle gives us a recipe for finding that equivalent nimber.

Small Positions

Let's first see the MEX Principle in action with a couple of small positions.

Example 3.14 (Chop)**.** Here we work out the nimber equivalents for some small positions in Chop.

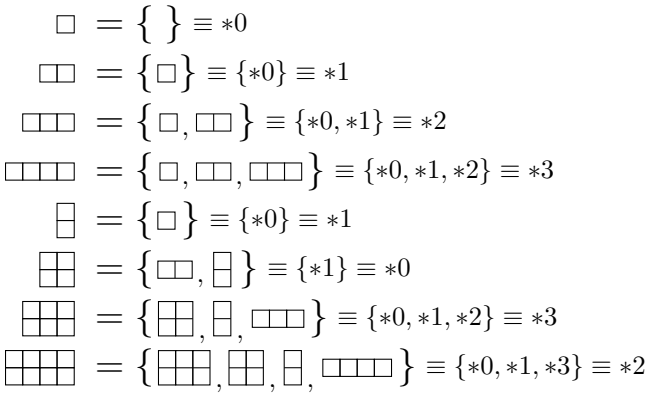


Figure 3.7: Equivalence in Chop

Example 3.15 (Chomp)**.** Here we work out the nimber equivalents for some small positions in Chomp.

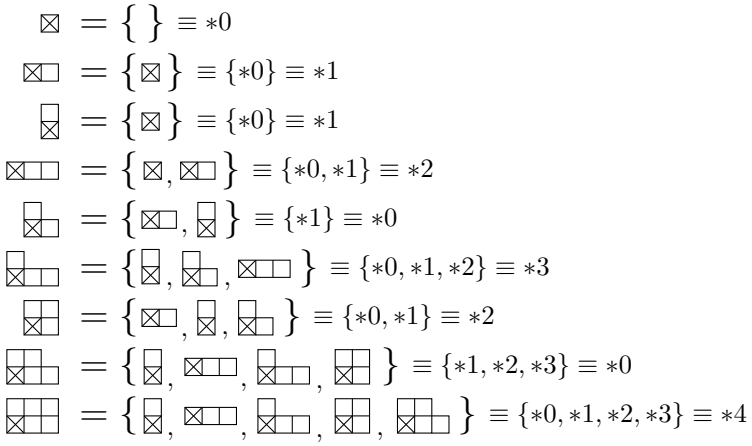


Figure 3.8: Equivalence in Chomp

Next, we'll see how our theory allows us efficiently to compute nimber equivalents to sums of positions.

Example 3.16. Find a nimber equivalent to the following position.

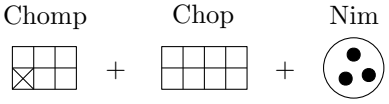


Figure 3.9: A sum of impartial games

We have shown that a 2×3 array in Chomp is equivalent to $*4$ and a 2×4 grid in Chop is equivalent to $*2$. Therefore the position in the figure is equivalent to $*4 + *2 + *3 \equiv *(4 \oplus 2 \oplus 3) \equiv *5$.

While knowing a nimber equivalent is useful, what we really want to know is how to win! If we are moving in an impartial game from a position α, we want to move to a position β of type P, if possible. This is because, by the definition of type P, we will have a winning strategy playing second from β. Accordingly, we define a *winning move* in an impartial game to be any move to a position of type P. Note that winning moves only exist from positions of type N.

In Nim, the balancing procedure always provides a winning move. We can also apply this procedure to play sums of positions involving other impartial games. For instance, if we wish to play the sum $\alpha + \beta + \gamma$ and we have found that $\alpha \equiv *a$ and $\beta \equiv *b$ and $\gamma \equiv *c$, then we just operate as if we were playing the Nim position $*a + *b + *c$.

Example 3.16 Continued. For the position in Figure 3.9, we have shown that the Chomp component is equivalent to $*4$ and the Chop component is equivalent to $*2$, so this position behaves like $*4 + *2 + *3$. The balancing procedure dictates that in the position $*4 + *2 + *3$, the $*4$ component is moved to $*1$. Therefore, a winning move for this Figure 3.9 position is to move the Chomp component to a 2×1 array which is equivalent to $*1$. At this point the full position will be equivalent to $*1 + *2 + *3 \equiv *0$ and thus of type P.

Pick-Up-Bricks and Chop

Now we will revisit the games of Pick-Up-Bricks and Chop. With the MEX Principle and induction, we can determine the nimber equivalent to any position in one of these games.

Theorem 3.17. *Let $n = 3\ell + k$ where $0 \leq k \leq 2$. Then a Pick-Up-Bricks position of n bricks is equivalent to $*k$.*

Proof. For every nonnegative integer m, define P_m to be a position in Pick-Up-Bricks with m bricks. Our goal is to prove $P_n = P_{3\ell+k} \equiv *k$ by induction on n. First consider both $n = 0$ and $n = 1$. When $n = 0$, the position has no moves, so it is equivalent to $*0$, or using our notation, $P_0 \equiv *0$. When $n = 1$, apply the MEX Principle and $P_1 = \{P_0\} \equiv \{*0\} \equiv *1$. This verifies our base cases. For the inductive step, assume $n \geq 2$ and that our formula holds for all smaller values. From a position with $n \geq 2$ bricks, a player may either remove one or two bricks, so

$$P_n = \{P_{n-1}, P_{n-2}\}.$$

As above, assume $n = 3\ell + k$ with $0 \leq k \leq 2$ and consider cases depending on k. In all of the cases, we may utilize our inductive hypothesis to determine both P_{n-1} and P_{n-2}.

$$\begin{aligned} k = 2 \quad & P_{3\ell+2} = \{P_{3\ell+1}, P_{3\ell}\} \equiv \{*1, *0\} \equiv *2 \\ k = 1 \quad & P_{3\ell+1} = \{P_{3\ell}, P_{3(\ell-1)+2}\} \equiv \{*0, *2\} \equiv *1 \\ k = 0 \quad & P_{3\ell} = \{P_{3(\ell-1)+2}, P_{3(\ell-1)+1}\} \equiv \{*2, *1\} \equiv *0 \end{aligned}$$

So we see that in all cases, $P_{3\ell+k} \equiv *k$ as desired. □

Next we turn our attention to the game of Chop. Given an $m \times n$ position in Chop, a move by either player will either reduce the number of rows, or the number of columns, but not both. So, from an $m \times n$ position, a player may move either to an $m' \times n$ position with $1 \leq m' < m$ or to a $m \times n'$ position with $1 \leq n' < n$ (i.e. a player can either decrease m or decrease n). In fact, this behaves just like a two pile game of Nim given by $*(m-1)+*(n-1)$. Pause to convince yourself of this equivalence before reading the formal proof appearing next.

Theorem 3.18. *For every $m, n \geq 1$, an $m \times n$ position in Chop is equivalent to $*(m-1) + *(n-1)$.*

Proof. We let $C_{m,n}$ denote an $m \times n$ position in Chop and will prove by induction on $m+n$ that $C_{m,n} \equiv *(m-1) + *(n-1)$. As a base case, when $m + n = 2$ we have $m = n = 1$ and $C_{1,1}$ has no moves for either player, so $C_{1,1} \equiv *0 \equiv *0 + *0$ and the formula holds. For the inductive step, $m + n > 2$, and we may assume the formula holds true for all smaller values of $m + n$. This gives us the following equation (here we use position notation $=$ twice, and the one instance of $\equiv$ comes from our inductive assumption).

$$\begin{aligned} C_{m,n} &= \big\{C_{1,n} \dots C_{(m-1),n}, \\ &\qquad C_{m,1} \dots C_{m,(n-1)}\big\} \\ &\equiv \Big\{\big(*0 + *(n-1)\big) \dots \big(*(m-2) + *(n-1)\big), \\ &\qquad \big(*(m-1) + *0\big) \dots \big(*(m-1) + *(n-2)\big)\Big\} \\ &= *(m-1) + *(n-1). \square \end{aligned}$$

Let's combine our results to analyze a more complicated sum.

Example 3.19. Find a nimber equivalent for the following position and a winning move, if it exists.

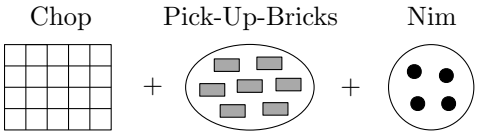


Figure 3.10: A complicated sum

By Theorem 3.18, a 4×5 game of Chop is equivalent to $*3 + *4 \equiv *7$ and, by Theorem 3.17, a Pick-Up-Bricks game with 7 bricks is equivalent to $*1$. Therefore, the position in

Figure 3.10 is equivalent to $*7 + *1 + *4 \equiv *(7 \oplus 1 \oplus 4) \equiv *2$. The balancing procedure applied to $*7 + *1 + *4$ instructs us to move the $*7$ component to $*5$. Therefore, a winning move in the given position is to move the Chop position from one equivalent to $*7$ to one equivalent to $*5$. This can be achieved by moving the 4×5 array to a 2×5 array.

Chapter 4

Hackenbush and Partizan Games

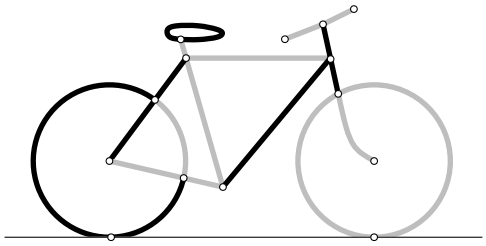


Figure 4.1: A Hackenbush bicycle

The central goal in combinatorial game theory is to understand positions. How can we determine a position's type? What's a good way to play? In the previous chapter we developed a powerful theory for impartial games which provides answers to these questions. We proved the Sprague-Grundy Theorem which says that every position in an impartial game is equivalent to a nimber. Further, we established the MEX Principle which gave us a way to compute these nimbers.

In this chapter, we turn to partizan games to develop a roughly analogous theory. Where our analysis of impartial games centered on the game Nim, here a game called Hackenbush takes center stage. We will assign numbers (similar to nimbers) to certain special positions in Hackenbush. We will then prove the Simplicity Principle, which can be used (like the MEX Principle) to show that some (but not all!) positions in other partizan games are equivalent to these special Hackenbush positions.

Since the game Hackenbush will play such a key role in this chapter, let's play it!

Game 4.1 (Hackenbush)**.** This is a normal-play game played between Louise and Richard. The game consists of a graph drawn with black and gray edges, some of which are attached to the ground. A move for **L**ouise is to erase a b**L**ack edge, while a move for **R**ichard is to erase a g**R**ay edge.[1] After an erasure, any part of the graph no longer connected to the ground floats away, out of the game. Since this game is normal-play, the last player to make a move wins.

[1]This is a grayscale version of the usual definition featuring blue and red edges, with **L**ouise deleting b**L**ue edges and Richard deleting **R**ed edges.

4.1 Hackenbush

To study Nim, we assigned numbers to certain positions (the nimbers) and then used them to understand more general positions. We adopt a similar approach here by assigning numbers to certain Hackenbush positions and then using these to understand more general positions. To start, define $\bullet 0$ to be the Hackenbush position with no edges (so there are no available moves). The position $\bullet 0$ is type P, so Proposition 2.15 gives us the following.

Proposition 4.2. *A Hackenbush position α satisfies $\alpha \equiv \bullet 0$ if and only if α is type P.*

The sum operation defined in Chapter 2 works on every normal-play game, so we can use it to add any two Hackenbush positions. Since the position $\bullet 0$ has no available moves for either player, we can see that, for every Hackenbush position α, we have $\alpha + \bullet 0 \equiv \alpha$. Indeed, the position $\bullet 0$ in Hackenbush behaves like the position $*0$ in Nim and the number 0 in the integers.

Negation

Something new for us in the world of games is a notion of negation. In Hackenbush, negation arises naturally since we can reverse the roles of the players in any position α by switching the colors of all the edges . This new position is denoted $-\alpha$ and is called the *negative* of α.

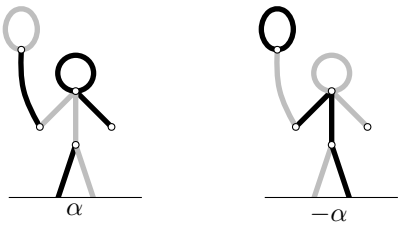


Figure 4.2: A position and its negative

The next proposition shows that negation of Hackenbush positions behaves just like negation of numbers.

Proposition 4.3. *If α and β are Hackenbush positions, then*

1. $-(-\alpha) \equiv \alpha$,
2. $\alpha + (-\alpha) \equiv \bullet 0$, *and*
3. $\beta + (-\alpha) \equiv \bullet 0$ *implies* $\alpha \equiv \beta$.

Proof. The first part follows immediately from the definition of negation, since $-(-\alpha)$ is identical to α. The previous proposition helps with the second part — it suffices to show that the position $\alpha + (-\alpha)$ is type P. So we will assume the role of the second player and show that we have a winning strategy. Symmetry is the key idea here. We call a Hackenbush position *symmetric* if it has the form $\gamma + (-\gamma)$. If we play second from a symmetric position, then whichever edge our opponent erases from one component, say γ, we will erase the

corresponding edge from the other component, $-\gamma$. This pair of moves returns the position to another symmetric one. Repeating this procedure guarantees us the last move and the win. Finally, the last part of the proposition follows from

$$\beta + (-\alpha) \equiv \bullet 0 \quad \Rightarrow \quad \beta + (-\alpha) + \alpha \equiv \alpha \quad \Rightarrow \quad \beta \equiv \alpha. \qquad \square$$

In the world of Hackenbush, we have a zero element, addition, and negation. These concepts all behave in ways familiar from the integers. Further, the last part of the previous proposition gives us a very convenient tool for proving that two Hackenbush positions are equivalent. This will be quite useful moving forward.

Integer Positions

Our next step will be to give names to some more Hackenbush positions. For every positive integer n, define $\bullet n$ to be the Hackenbush position consisting of n isolated black edges. For a negative integer $-m$, we define $\bullet(-m)$ to be the Hackenbush position consisting of m isolated gray edges.

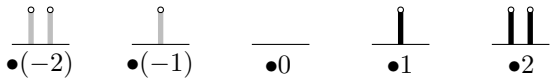


Figure 4.3: Some integral positions

At this point, we have a Hackenbush position $\bullet n$ associated to every integer n, and we call any position of this form an *integral position*. Just as we can negate integers and add integers, we can also negate Hackenbush positions and add Hackenbush positions. We next demonstrate that addition and negation on integral Hackenbush positions correspond naturally to addition and negation of integers.

We may view $\bullet$ as an operation that takes an integer and outputs a special kind of Hackenbush position. In this context, imagine starting with the integer n and applying negation and the $\bullet$ operation. We can do these two operations in two different orders (since we can negate both integers and Hackenbush positions). Specifically, we can go from n to $-n$ to $\bullet(-n)$ or we can go from n to $\bullet n$ to $-(\bullet n)$. Our definitions happily imply that the positions $\bullet(-n)$ and $-(\bullet n)$ are equivalent (and, in fact, $\bullet(-n)$ and $-(\bullet n)$ are identical). In short, we have $- \bullet n \equiv \bullet(-n)$.

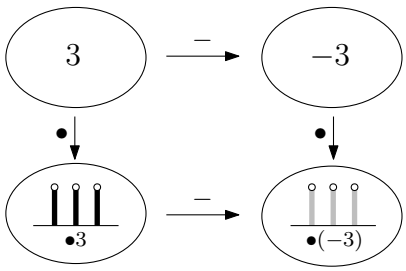


Figure 4.4: Negation and $\bullet$

What about the relationship between the operations of $\bullet$ and addition? Given two integers m and n, we can perform the operations of $\bullet$ and $+$ in two different orders. We can first add integers m and n to $m+n$ and then apply the $\bullet$ operation to the sum to get $\bullet(m+n)$. Alternately, we can first apply the $\bullet$ operation to integers m and n to get $\bullet m$ and $\bullet n$, and then add these to get $\bullet m + \bullet n$. We claim that the two positions resulting from this process will always be equivalent, or, more succinctly, $\bullet m + \bullet n \equiv \bullet(m+n)$.

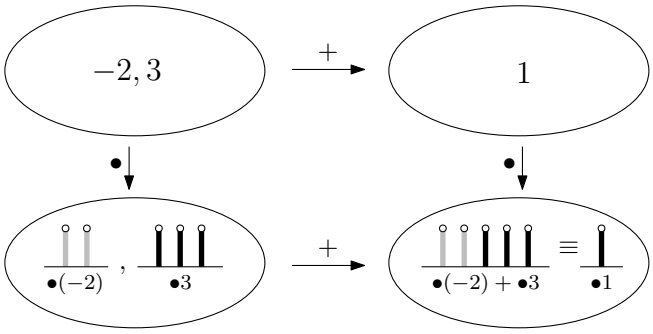


Figure 4.5: Sum and $\bullet$

If m and n have the same sign, then these two positions will be identical (and they will consist entirely of edges of the same color) and the result is automatic. Suppose next that m and n have different signs and set $\ell = |m-n|$. In this case, $\bullet(m+n)$ will consist of ℓ isolated edges of the same color, while $\bullet m + \bullet n$ will have edges of both colors, but ℓ more of one color than the other. Fortunately, Proposition 4.3 and the above analysis of negation imply that $\bullet k + \bullet(-k) \equiv \bullet 0$ for every integer k. Therefore, modifying a position by deleting the same number of isolated gray edges and isolated black edges results in an equivalent position. It follows that $\bullet(m+n) \equiv \bullet m + \bullet n$, as desired. The following theorem summarizes these results.

Theorem 4.4. *For any integers m and n, we have*

1. $-(\bullet n) \equiv \bullet(-n)$
2. $(\bullet m) + (\bullet n) \equiv \bullet(m+n)$.

The above theorem provides a complete understanding of the behavior of integral Hackenbush positions under sums and negation. These positions behave just like the integers with which they are named. For instance, the position $\bullet a + \bullet(-b) + \bullet(c) + \bullet(-d)$ will be equivalent to the position $\bullet(a-b+c-d)$. It is straightforward to determine the type of an integer position, since in each such position some player has no moves. The position $\bullet n$ will be type L when $n > 0$, type P when $n = 0$, and type R when $n < 0$.

We have just observed that for $n > 0$, the integral position $\bullet n$ will be type L and thus have an advantage for Louise. In fact, it is reasonable to quantify this advantage and say that the position $\bullet n$ gives Louise an advantage of n. We would likewise consider the integral position $\bullet(-n)$ as one that has an advantage of n for Richard. Of course, this interpretation of these integral positions applies more broadly also to positions which are equivalent to integral positions. We would thus consider an arbitrary Hackenbush position α to offer

Louise an advantage of n if $\alpha \equiv \bullet n$ and we would consider α to offer Richard an advantage of n if $\alpha \equiv \bullet(-n)$.

Figure 4.6: A position with an advantage of 2 for Louise

What happens to this concept of advantage when we add positions? Suppose that α offers Louise an advantage of m and β offers her an advantage of n. Then we have $\alpha \equiv \bullet m$ and $\beta \equiv \bullet n$, so $\alpha + \beta \equiv \bullet m + \bullet n \equiv \bullet(m+n)$. Thus, $\alpha + \beta$ will give Louise an advantage of $m + n$. So, Louise's advantage in $\alpha + \beta$ is the sum of her advantages in α and β.

Fractional Positions?

Could there be a Hackenbush position α that gives Louise an advantage of $\frac{1}{2}$? If so, we would expect $\alpha + \alpha$ to offer Louise an advantage of 1, and then the position $\alpha + \alpha + \bullet(-1)$ would be equivalent to $\bullet 0$ (of type P), with no advantage for either player. Test this for the position α consisting of one gray edge on top of one black edge, as seen in the following figure.

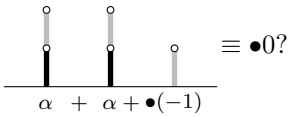


Figure 4.7: Is α worth $\frac{1}{2}$ to Louise?

A bit of analysis reveals that the full position in Figure 4.7 is of type P, so it is indeed equivalent to $\bullet 0$. So, the position α is one that gives Louise an advantage of $\frac{1}{2}$! Using negation, we would consider $-\alpha$ to be one which offers Richard an advantage of $\frac{1}{2}$. Could there be a position β which offers Louise an advantage of $\frac{1}{4}$? If so, we would expect $\beta + \beta + (-\alpha) \equiv \bullet 0$. Test this by considering the position β below.

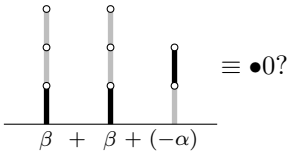


Figure 4.8: is β worth $\frac{1}{4}$ to Louise?

Again, a bit of analysis reveals that the full position in Figure 4.8 is type P and thus equivalent to $\bullet 0$. So we consider β to be a position with an advantage for Louise of $\frac{1}{4}$. We now introduce some terminology to help formalize these new ideas. For every positive integer k, define $\bullet\frac{1}{2^k}$ to be the following Hackenbush position.

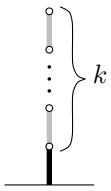
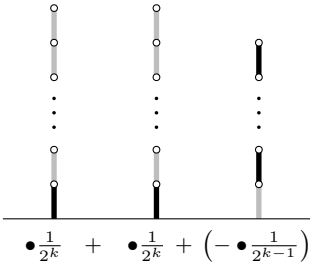


Figure 4.9: The position $\bullet\frac{1}{2^k}$

Now extend the analysis we used on Figures 4.7 and 4.8 to the more general $\bullet\frac{1}{2^k}$ positions defined above.

Lemma 4.5. *For every positive integer k, we have $\bullet\frac{1}{2^k} + \bullet\frac{1}{2^k} \equiv \bullet\frac{1}{2^{k-1}}$.*

Proof. It suffices to prove $\bullet\frac{1}{2^k}+\bullet\frac{1}{2^k}+(-\bullet\frac{1}{2^{k-1}}) \equiv \bullet 0$ for all $k \geq 1$, and we do this by induction on k. The base case $k = 1$ was handled in our earlier discussion. For the inductive step ($k \geq 2$), we will prove the desired equation by showing that $\bullet\frac{1}{2^k} + \bullet\frac{1}{2^k} + (-\bullet\frac{1}{2^{k-1}})$ is type P.



We first consider the case that Richard plays first, and will show that Louise has a winning strategy as the second player. If Richard choses the gray edge from the $-\bullet\frac{1}{2^{k-1}}$ component, then Louise will win no matter what. Suppose instead that Richard chooses a gray edge from one of the $\bullet\frac{1}{2^k}$ components. Since it cannot hurt his future options and does not affect Louise's, we may assume that he chooses the topmost edge (convince yourself this makes sense). Now Louise can remove the bottom edge from the other $\bullet\frac{1}{2^k}$ component to bring the full position to $\bullet\frac{1}{2^{k-1}} + (-\bullet\frac{1}{2^{k-1}}) \equiv \bullet 0$. As second player from this position, Louise has a winning strategy.

We next consider the case that Louise plays first, and will show that Richard has a winning strategy as the second player. If Louise removes the black edge from one of the $\bullet\frac{1}{2^k}$ components, then Richard may remove the topmost gray edge from the other $\bullet\frac{1}{2^k}$ component to bring the position to $\bullet\frac{1}{2^{k-1}} + (-\bullet\frac{1}{2^{k-1}}) \equiv \bullet 0$. From there, Richard has a winning strategy as second player. If Louise removes a black edge from the $-\bullet\frac{1}{2^{k-1}}$ component, we may assume that she removes the topmost, bringing this to $-\bullet\frac{1}{2^{k-2}}$. Now, let Richard remove the topmost gray edge from one of the $\bullet\frac{1}{2^k}$ components. The position is then $\bullet\frac{1}{2^k} + \bullet\frac{1}{2^{k-1}} + (-\bullet\frac{1}{2^{k-2}})$. By induction, $\bullet\frac{1}{2^{k-1}}+\bullet\frac{1}{2^{k-1}}+(-\bullet\frac{1}{2^{k-2}}) \equiv \bullet 0$, so Richard has a winning strategy as second player in $\bullet\frac{1}{2^{k-1}} + \bullet\frac{1}{2^{k-1}} + (-\bullet\frac{1}{2^{k-2}})$. Since the present position is obtained from this one by adding a gray edge on top, this strategy also guarantees Richard a win in the present position. □

For every positive integer k, we have now established a position $\bullet\frac{1}{2^k}$ that may be interpreted as having an advantage of $\frac{1}{2^k}$ for Louise. Although it is clear that these type L

positions offer Louise an advantage, the precise quantification of this advantage is somewhat surprising. We can use these curious new fractional positions to analyze more general Hackenbush positions, but first we need to introduce an important set of numbers.

4.2 Dyadic Numbers and Positions

This section introduces the dyadic numbers and establishes some of their basic properties. With the requisite properties of these numbers in place, we will then define some special Hackenbush positions based on them.

The Dyadic Numbers

Any number that can be expressed as a fraction where the denominator is a power of two (and the numerator is an integer) is called a *dyadic number*. So for instance, $\frac{17}{32}$ and $\frac{531}{64}$ are dyadic numbers, but $\frac{2}{7}$ and π are not.

As before, the *binary expansion* of a number q is a representation of q as a sum of distinct powers of 2. We now will also use negative powers of 2 (or, equivalently, fractions of the form $\frac{1}{2^d}$ with $d > 0$). Finding binary expansions of dyadic numbers is straightforward given the procedure for integers, as illustrated by the following example:

$$\begin{aligned}\tfrac{83}{64} &= \tfrac{1}{64}(83)\\ &= \tfrac{1}{64}(64+19)\\ &= \tfrac{1}{64}(64+16+3)\\ &= \tfrac{1}{64}(64+16+2+1)\\ &= 1+\tfrac{1}{4}+\tfrac{1}{32}+\tfrac{1}{64}.\end{aligned}$$

Next is formal proof that this procedure always works.

Proposition 4.6. *Every dyadic number number has a unique (finite) binary expansion.*

Proof. Let $q = \frac{n}{2^k}$ be a dyadic number. Since n is an integer, it has a binary expansion $n = 2^{d_1} + 2^{d_2} + \ldots + 2^{d_\ell}$. Therefore, q has binary expansion given by $q = 2^{d_1-k} + 2^{d_2-k} + \ldots + 2^{d_\ell-k}$. To see that this is unique, suppose (for a contradiction) that we have two different binary expansions of q. If the smallest power of two appearing in either one is 2^{-d}, then multiplying both expressions through by 2^d gives two different binary expansions of the integer $2^d q$ which is impossible. □

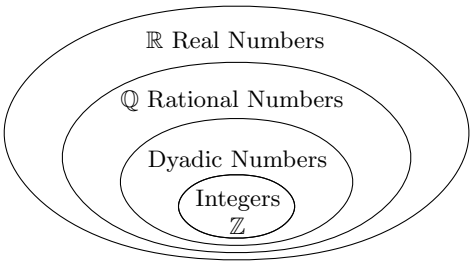


Figure 4.10: Sets of numbers

For every dyadic number $\frac{m}{2^k}$, its negation $\frac{-m}{2^k}$ is another dyadic number. Similarly, the sum $(2^h m + 2^k n)/2^{k+h}$ of a pair of dyadic numbers, $\frac{m}{2^k}$ and $\frac{n}{2^h}$, is likewise another dyadic number. This property is known as *closure*, so we say that the dyadic numbers are *closed* under negation and addition. Many familiar and fundamental sets of numbers such as $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$ are closed under these operations. There are other, lesser-known, sets like the dyadic numbers that also exhibit these properties. The Venn Diagram in Figure 4.10 shows the relationship between these various closed subsets of numbers.

Birthdays of Dyadic Numbers

Although we generally view the dyadic numbers as static, we are going to take a very different perspective here. We will imagine time moving forward one day at a time, starting with day 0, and we will think of each dyadic number as appearing, or being born, on a certain day. This gives each dyadic number a *birthday*, and these birthdays will be a key concept moving forward. Before formalizing the procedure, let's start by seeing which numbers are born on the first few days. For each day n, the following figure shows in black the new numbers born on day n, while the numbers born on days before n are gray.

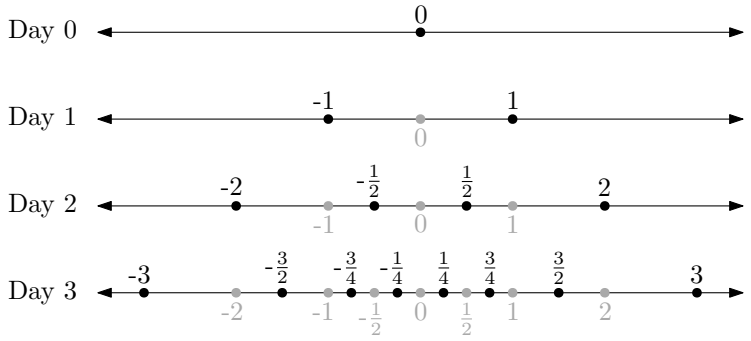


Figure 4.11: Birthdays of some dyadic numbers

Here is the general rule: On day 0 the number 0 is born. If $a_1 < a_2 < \ldots < a_\ell$ are the numbers born on days $0, 1, \ldots, n$ then on day $n+1$ the following new numbers are born:

- The largest integer which is less than a_1
- The smallest integer which is greater than a_ℓ
- The number $(a_i + a_{i+1})/2$ for every $1 \leq i \leq \ell - 1$

A standard exercise shows that the set of all numbers created in the above manner is precisely the set of dyadic numbers. The other necessary property we require is the following.

Proposition 4.7. *Every open interval of real numbers (a, b), (a, ∞), $(-\infty, b)$, or even $(-\infty, \infty)$ contains a unique oldest dyadic number.*

Proof. First, observe that every interval of length $> \frac{1}{2^k}$ will contain a dyadic number which can be expressed as a fraction with denominator 2^k. It follows that every open interval

contains at least one dyadic number. To complete the proof, we need to show that there cannot be two distinct dyadic numbers q_1, q_2 in the interval I so that both q_1 and q_2 are oldest numbers in I. Suppose (for a contradiction) that this is true. If q_1 and q_2 are both born on day n, then by the construction, there must exist a dyadic number q born on a day before n which is between q_1 and q_2. However, then q is also in the interval I and is older than q_1 or q_2, which is a contradiction. □

Note, tangentially, that this recursive process with which we have constructed the dyadic rationals can be continued to day infinity and beyond! This results in an amazing number system called the Surreal Numbers.

Dyadic Positions

Now that we have a good handle on dyadic numbers, we will introduce associated Hackenbush positions called dyadic positions. For every dyadic number $q > 0$ with binary expansion $2^{d_1} + 2^{d_2} + \ldots + 2^{d_\ell}$ (here $d_1 > d_2 > \ldots > d_\ell$ are integers which may be positive or negative), we define the position[2]

$$\bullet q = \bullet 2^{d_1} + \bullet 2^{d_2} + \ldots + \bullet 2^{d_\ell}.$$

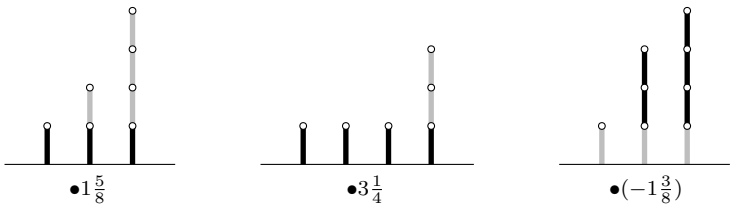


Figure 4.12: Some dyadic positions

To get negative positions, we define $\bullet(-q) = -(\bullet q)$. We will call any position of the form $\bullet q$ a *dyadic position*. In the previous section, we proved that addition and negation for integral Hackenbush positions behaves just like addition and negation does for the integers. The next step is to extend this property to the dyadic numbers. In preparation for this we first prove the following lemma.

Lemma 4.8. *Let $a_1, \ldots, a_n$ be numbers, each of which is either 0 or has the form $\pm 2^k$ for some integer k. If $a_1 + a_2 + \ldots + a_n = 0$, then $\bullet a_1 + \bullet a_2 + \ldots + \bullet a_n \equiv \bullet 0$.*

Proof. We proceed by induction on n. If $n = 1$, there is just one number a_1, so we must have $a_1 = 0$, and thus $\bullet a_1 \equiv \bullet 0$ as desired. For the inductive step, let $a_1, \ldots, a_n$ be a list with $n \geq 2$ satisfying the assumptions, and assume the lemma is true for any list with $< n$ terms. If all of the numbers in our list are integers, then Theorem 4.4 implies that $\bullet a_1 + \ldots + \bullet a_n = \bullet(a_1 + \ldots + a_n) = \bullet 0$ and we are done. Otherwise, choose k to be the largest number so that one of $a_1, \ldots, a_n$ is equal to $\pm\frac{1}{2^k}$. The ordering of the numbers in

[2]Technically speaking, we are redefining some integer positions here. For instance 7 is an integer, but also a dyadic rational so we are redefining $\bullet 7 = \bullet 4 + \bullet 2 + \bullet 1$. Fortunately, this definition agrees with our earlier one, so all is well.

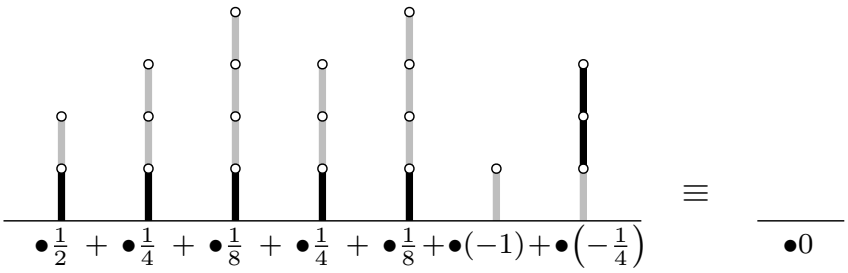


Figure 4.13: A Hackenbush position equivalent to $\bullet 0$.

our list does not matter, so by reordering we may assume $a_n = \pm\frac{1}{2^k}$. If all of the terms in $a_1, \ldots, a_{n-1}$ can be expressed as fractions with denominator 2^{k-1} then it is not possible for $a_1 + \ldots + a_n = 0$. So, there must be another term in $a_1, \ldots, a_{n-1}$ that is equal to $\pm\frac{1}{2^k}$. By reordering, we may assume $a_{n-1} = \pm\frac{1}{2^k}$. Define $b = a_{n-1} + a_n$ and note that either $b = 0$ or $b = \pm\frac{1}{2^{k-1}}$. Now our lemma applies by induction to the sequence $a_1, \ldots, a_{n-2}, b$. This gives the equation

$$\bullet a_1 + \ldots + \bullet a_{n-2} + \bullet b \equiv \bullet 0.$$

If a_n and a_{n-1} have opposite signs, then $b = 0$ and $\bullet a_{n-1} + \bullet a_n \equiv \bullet 0 \equiv \bullet b$ which combines with the above equation to give the desired result. On the other hand, if a_n and a_{n-1} have the same sign, then Lemma 4.5 implies that $\bullet a_{n-1} + \bullet a_n \equiv \bullet b$ and again we may combine this with the above equation to get the desired result. □

Since each dyadic position $\bullet q$ is a sum of positions of the form $\bullet(\pm 2^d)$, the previous lemma is all we need to establish the behavior of dyadic positions under sums. For instance, the proof that the equation in Figure 4.14 holds true follows from the equation in Figure 4.13.

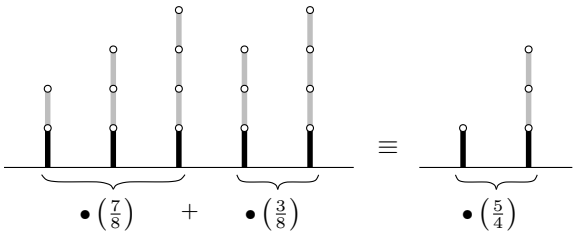


Figure 4.14: Adding two dyadic positions

Theorem 4.9. *If p, q are dyadic numbers, then*

1. $-(\bullet p) \equiv \bullet(-p)$
2. $(\bullet p) + (\bullet q) \equiv \bullet(p + q)$

Proof. The first part of this theorem follows from the definitions of negation and our dyadic positions. For the second part, apply the previous lemma to obtain $\bullet p + \bullet q + (-\bullet(p + q)) \equiv \bullet 0$. Then adding $\bullet(p + q)$ to both sides of this equation yields the desired result. □

It is rather straightforward to determine the type of a dyadic position. For a positive rational number $q > 0$ the position $\bullet q$ is a sum of positions of type L and therefore $\bullet q$ is type L. Conversely, the position $\bullet(-q)$ is a sum of positions of type R, so $\bullet(-q)$ will also be type R. This gives us the following

Observation 4.10. *For a dyadic number q we have*

$$\bullet q \text{ is type } \begin{cases} L & if \quad q > 0 \\ P & if \quad q = 0 \\ R & if \quad q < 0. \end{cases}$$

The above observation and previous theorem give us an appealing extension to our previous interpretation of advantage. Namely, we may now consider a position α with $\alpha \equiv \bullet q$ as one which offers an advantage of q for Louise. Then adding two positions which give Louise an advantage of p and q yields a position which gives Louise an advantage of $p + q$.

4.3 The Simplicity Principle

Chapter 3 introduced the MEX Principle, which gave us a recursive procedure to determine the nimber equivalent of any position in an impartial game. This section introduces an analogous principle for impartial games called the Simplicity Principle. This new principle provides a procedure to determine a dyadic position equivalent to a given one under certain assumptions.

The Proof

While the available moves for both players are the same in impartial games, in Hackenbush they are generally quite different. Accordingly, we will use the position notation introduced in Section 2.1 to denote Hackenbush positions (and more generally positions in partizan games). As in the figure, the available moves for Louise are to the left of the bar, and those for Richard are to the right.

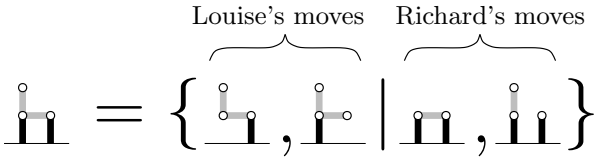


Figure 4.15: Position notation

Our proof of the Simplicity Principle involves some basic properties about positions of the form $\bullet\frac{1}{2^k}$. Using the above position notation with dyadic positions, for every $k > 0$ we have

$$\bullet\tfrac{1}{2^k} = \left\{\bullet 0 \mid \bullet 1, \bullet\tfrac{1}{2}, \ldots, \bullet\tfrac{1}{2^{k-1}}\right\}.$$

So, if a player moves $\bullet\frac{1}{2^k}$ to $\bullet c$, then either Richard moved and c is at least $\frac{1}{2^k}$ larger than $\frac{1}{2^k}$, or Louise moved and c is $\frac{1}{2^k}$ smaller. This principle immediately yields the following.

Lemma 4.11. *Let $c = \frac{n}{2^k}$ with $k \geq 1$ and suppose a player moves the position $\bullet c$ to the new position $\bullet c'$.*

1. *If Louise moved, then $c' \leq c - \frac{1}{2^k}$.*

2. *If Richard moved, then $c' \geq c + \frac{1}{2^k}$.*

We are now ready to introduce the Simplicity Principle that we will use to prove that certain positions in partizan games are equivalent to dyadic positions.

Theorem 4.12 (The Simplicity Principle)**.** *Consider a position in a partizan game given by $\gamma = \{\alpha_1, \ldots, \alpha_m \mid \beta_1, \ldots, \beta_n\}$ and suppose*

$$\alpha_i \equiv \bullet a_i \text{ for } 1 \leq i \leq m.$$
$$\beta_j \equiv \bullet b_j \text{ for } 1 \leq j \leq n.$$

If there do not exist a_i and b_j with $a_i \geq b_j$, then $\gamma \equiv \bullet c$ where c is the oldest number larger than all of $a_1, \ldots, a_m$ and smaller than all of $b_1, \ldots, b_n$.

Proof. Assume first that both players have available moves and (by possibly reordering) that $a_m \leq \ldots \leq a_1$ and $b_1 \leq \ldots \leq b_n$. Then c is the oldest number in the interval (a_1, b_1). To simplify, assume further that $c = \frac{\ell}{2^k}$ where $\ell, k \geq 1$ (the other cases are similar). Since both $c + \frac{1}{2^k}$ and $c - \frac{1}{2^k}$ are older than c, we have:

$$c - \tfrac{1}{2^k} \leq a_1 < c < b_1 \leq c + \tfrac{1}{2^k}.$$

To prove that $\gamma \equiv \bullet c$, it suffices to show that $\gamma + \bullet(-c) \equiv \bullet 0$ (then adding $\bullet c$ to both sides yields our desired equation). We then need to prove that the position $\gamma + \bullet(-c)$ is type P. That is, for $\gamma + \bullet(-c)$, Richard has a winning strategy when Louise plays first, and Louise has a winning strategy when Richard plays first.

Assume Louise plays first and moves the γ component to α_i. The new position $\alpha_i + \bullet(-c)$ is equivalent to $\bullet a_i + \bullet(-c)$. Since $a_i - c \leq a_1 - c < 0$, Richard has a winning strategy from here. Suppose next that Louise plays first and moves in the $\bullet(-c)$ component. In this case, Lemma 4.11 implies that the new position will be equivalent to $\gamma + \bullet(-c')$ where $(-c') \leq (-c) - \frac{1}{2^k}$. So, now Richard can move γ to β_1 bringing the game to a position equivalent to $\bullet b_1 + \bullet(-c')$. Since $b_1 - c' \leq b_1 - c - \frac{1}{2^k} \leq 0$ this new position is either of type R or P and in either case, Richard has a winning strategy.

Similarly, if Richard plays first and moves in γ, the new position is equivalent to $\bullet b_j + \bullet(-c)$ and since $b_j - c > 0$, Louise has a winning strategy from here. If Richard plays first and moves $\bullet(-c)$ to $\bullet(-c')$, then Louise can move γ to α_1 bringing the full position to one equivalent to $\bullet a_1 + \bullet(-c')$. Since $(-c') \geq (-c) + \frac{1}{2^k}$, we have $a_1 - c' \geq a_1 - c + \frac{1}{2^k} \geq 0$, so playing second from here, Louise has a winning strategy. □

Example: Here we use the Simplicity Principle to find a dyadic position equivalent to a chair-shaped Hackenbush position.

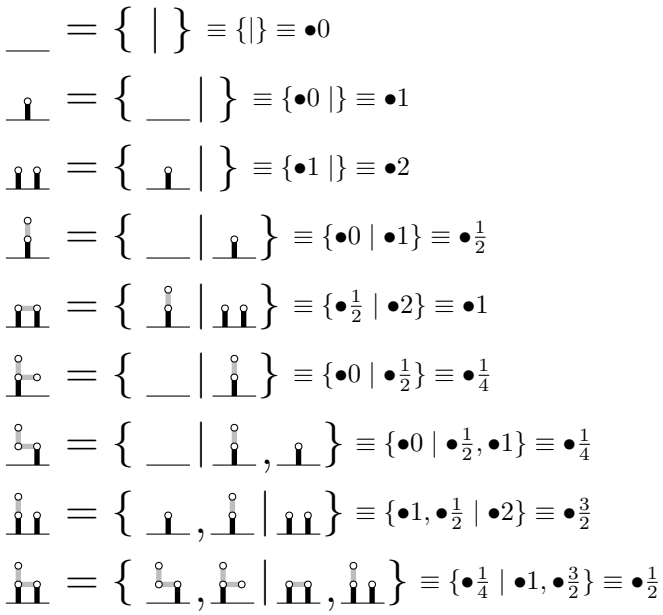


Figure 4.16: Applying the Simplicity Principle in Hackenbush

Applying The Simplicity Principle

In the previous chapter, we established the MEX Principle, which can be used (recursively) to show that a position in an arbitrary impartial game is equivalent to a nimber. So even though the MEX Principle is based on Nim, it can be more broadly applied. The Simplicity Principle acts similarly. Although dyadic positions are Hackenbush positions, the Simplicity Principle can be used (recursively) to show that many positions in other partizan games are equivalent to dyadic positions. Let's do a small example from Domineering.

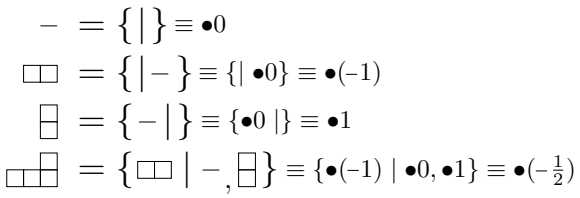


Figure 4.17: Domineering positions equivalent to dyadic positions

The Simplicity Principle is also the key to understanding Cut-Cake. To start, note that a $1 \times n$ piece in Cut-Cake is equivalent to $\bullet(n-1)$ since in this position Louise has $n-1$ moves — which she can make in any order—, but Richard has no available moves. Similarly an $n \times 1$ piece in Cut-Cake is equivalent to $\bullet(-n+1)$. Next we apply the Simplicity Principle to analyze a small position in this game, frequently taking advantage of our knowledge of sums.

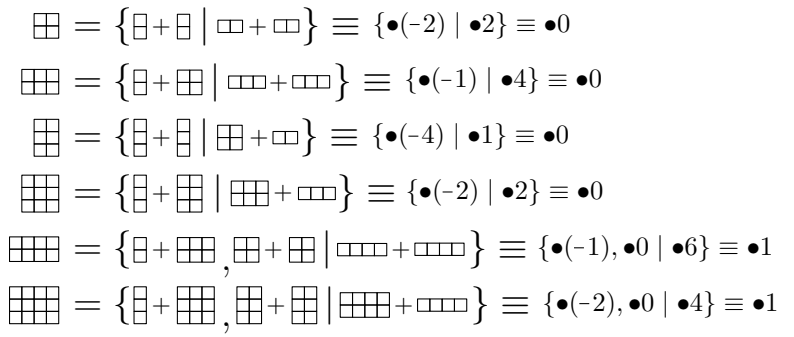


Figure 4.18: Cut-Cake positions equivalent to dyadic positions

Since we understand how dyadic positions behave under sums, we can also evaluate sums of positions which are equivalent to dyadic positions. For instance if α, β are positions equivalent to $\bullet a, \bullet b$, then $\alpha + \beta \equiv \bullet a + \bullet b$. Let's do an example.

Example 4.13. By earlier analysis, the position in the figure below is equivalent to $\bullet\frac{3}{4}$, so it is type L and gives Louise an advantage of $\frac{3}{4}$.

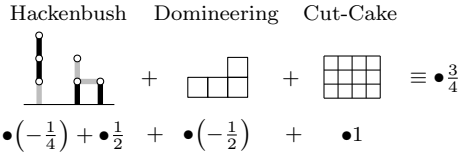


Figure 4.19: Sums of positions

The Big Picture

In the great world of normal-play games, we have now established two important sets of positions: the nimbers and the dyadic positions. We can use the MEX Principle to show that any position in an impartial game is equivalent to a nimber. Likewise, we can use the Simplicity Principle to show that some positions in partizan games are equivalent to dyadic positions. Our theory in both cases also allows us to understand sums. If two positions are equivalent to nimbers, say $*a$ and $*b$, then their sum will be equivalent to $*(a \oplus b)$. On the other hand, if two positions are equivalent to dyadic positions $\bullet a$ and $\bullet b$, their sum will be equivalent to $\bullet(a + b)$.

Nimbers	Dyadic Positions
MEX Principle to show equivalence	Simplicity Principle to show equivalence
Nim-sum to add: $*a + *b \equiv *(a \oplus b)$	Usual sum to add: $\bullet a + \bullet b \equiv \bullet(a + b)$
Applies to **all** impartial games	Applies to **some** partizan games

As we have mentioned, the Simplicity Principle does not apply to every position in a partizan game. For the position

$$\gamma = \{\alpha_1, \ldots, \alpha_m \mid \beta_1, \ldots, \beta_n\} \equiv \{\bullet a_1, \ldots, \bullet a_m \mid \bullet b_1, \ldots, \bullet b_n\}$$

the Simplicity Principle is only applicable when $\max\{a_1, \ldots, a_m\} < \min\{b_1, \ldots, b_n\}$. As a result, there is a wide world of partizan-game positions that are not equivalent to dyadic positions.

Earlier in this section, we succeeded in using the Simplicity Principle to show that a certain position in Domineering is equivalent to $\bullet(-\frac{1}{2})$. That calculation worked since the Simplicity Principle applied at each step. However, there are other positions in Domineering that behave differently. For instance, consider a 2×2 array as in Figure 4.20. Any move by Louise brings the game to a 1×2 array which is equivalent to $\bullet 1$, while any move by Richard brings the game to a 2×1 array which is equivalent to $\bullet(-1)$.

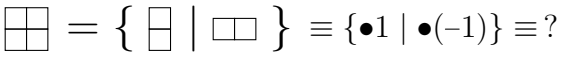


Figure 4.20: A position in Domineering

Our analysis demonstrates that this 2×2 position is equivalent to $\{\bullet 1 \mid \bullet(-1)\}$. Yet, since the maximum of the set of numbers on the left hand side is greater than the minimum of those on the right, the Simplicity principle does not apply here. What distinguishes this position from any dyadic position is the fact that each player would prefer to move first — this is a position of type N.

In fact, there is a vast and vibrant theory of partizan games featuring a wide variety of games and positions, some fascinating theoretical results, and numerous tantalizing unsolved problems. We encourage the interested reader to continue with either the brilliant classic *Winning Ways* or the modern update *Lessons in Play*.

Index