https://doi.org/10.1090//stml/080/01

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.

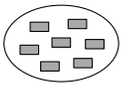


Figure 1.1. A 7-brick position in Pick-Up-Bricks

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.

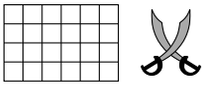


Figure 1.2. A 4×6 position in Chop

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 northeast corner). The last player to eat a nonpoison square wins.

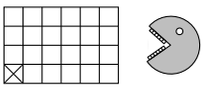


Figure 1.3. A 4×6 position in Chomp

Game 1.4 (Hex)**.** This is a game played on the board pictured in Figure 1.4.[1] Similar to the widely familiar game Tic-Tac-Toe, on each turn a player marks an empty hexagon. One player uses $*$ as his 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.

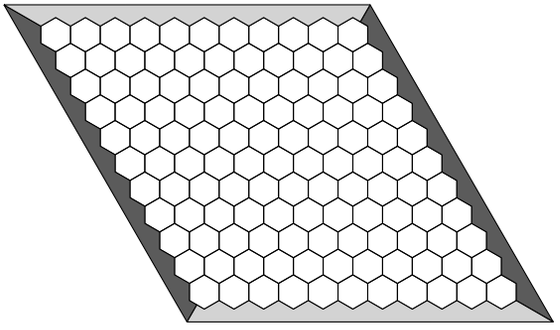


Figure 1.4. The board for Hex

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.

[1] A large version of this game board (as well as the pictured positions in Chop and Chomp) can be found at the end of this book and also online at `www.ams.org/bookpages/stml-80`.

Definition 1.5. A *combinatorial game* is a 2-player game played between **L**ouise (for **L**eft) and **R**ichard (for **R**ight).[2] The game consists of the following:

(1) A set of possible *positions*. These are the states of the game (e.g. 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. 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. Later in Chapter 7, we will broaden our horizons and introduce some of these variations. For now, though, we restrict ourselves to combinatorial games.

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

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

begin by seeing how to model the play of any combinatorial game in this manner.

Modeling 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 Richard or Louise gets two adjacent squares marked with his or her symbol, then he or she wins. If all squares get marked without this happening, the games ends in a draw.

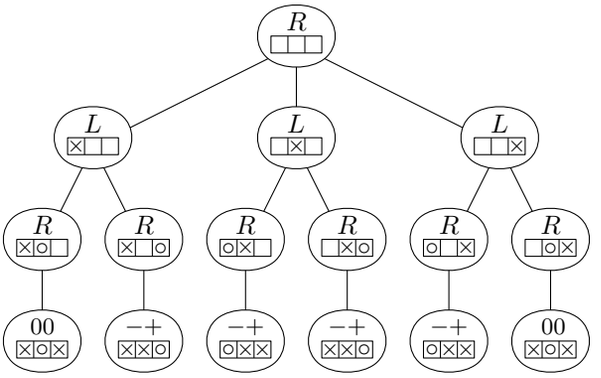


Figure 1.5. A game tree for Tic

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 the 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.

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

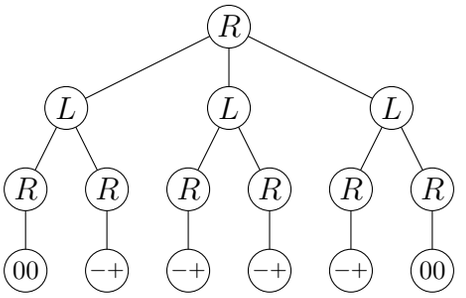


Figure 1.6. A W-L-D game tree for Tic

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 game tree in Figure 1.8, we have depicted a strategy for Richard by boldfacing the edges indicating his choices.

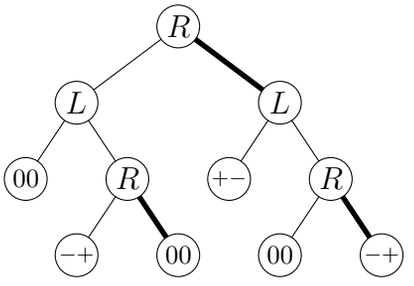


Figure 1.8. A strategy for Richard

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 labeled *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 labeled *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.

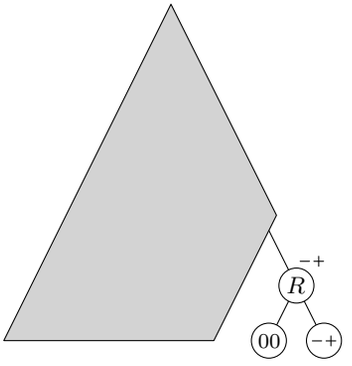


Figure 1.9. An easy decision for Richard

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 Backwards)**.** 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.

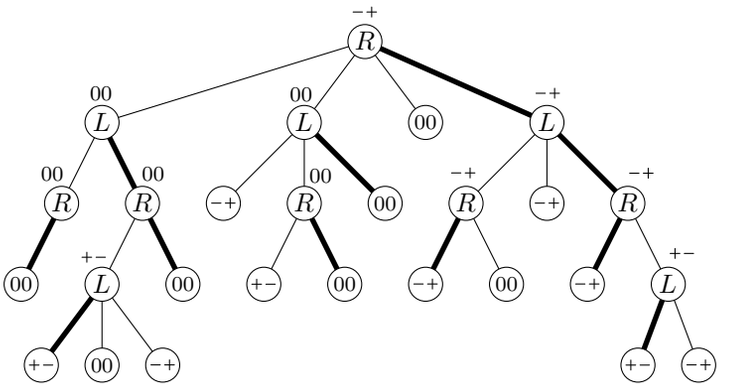


Figure 1.10. Rational decision-making

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 backwards 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 $n \geq 0$.

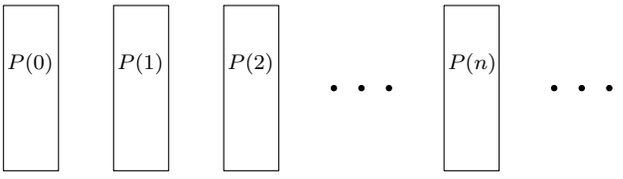


Figure 1.11. A proof by induction

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 $P(n)$ is also true.[3] 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.

[3]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.

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

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

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

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

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 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 (e.g. 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 nonnegative 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 $N_1, N_2, \ldots, N_\ell$.

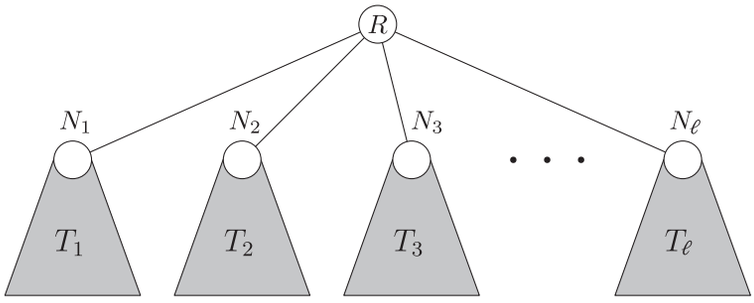


Figure 1.12. The first move in a game tree

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

Case 1. *At least one of $T_1, \ldots, T_\ell$ 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 $T_1, \ldots, T_\ell$ 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 $T_1, \ldots, T_\ell$ 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 $T_1, \ldots, T_\ell$ is type $-+$, each of Louise's strategies $\mathcal{L}_1, \ldots, \mathcal{L}_\ell$ 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 Backwards 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[4]. 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 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

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

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 nonsquare position. Note that the second player can always move a nonsquare 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 a multiple of 3. 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 3. To start, the first player may remove either one or two bricks to bring the position to a multiple of 3. 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 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 Figure 1.13.

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 his or her 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. □

In Chapter 9, Exercise (11), we will show 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. Assuming Hex cannot end in a draw (this fact is proved in Exercise (11) of Chapter 9), 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.

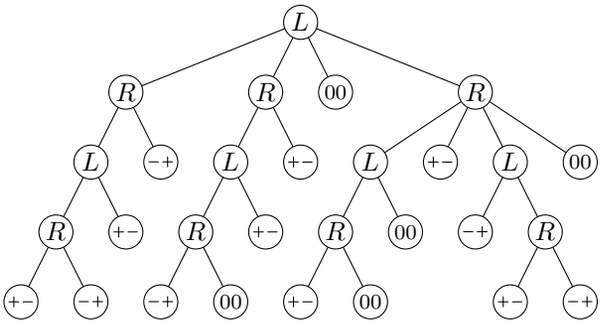
There is one slight complication we ignored in the above argument, but it isn't hard to fix. Namely, it might be the case that at some point the strategy $\mathcal{S}$ we are following in the pretend game instructs us to mark the hexagon h with $*$. Since h is already occupied by a $*$, this is not a 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. □

Exercises

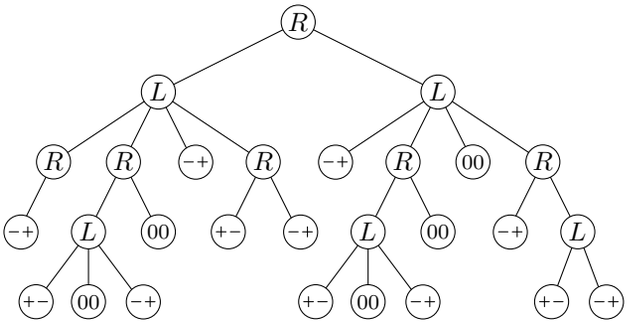
(1) For each game below, construct a game tree with Louise moving first from the indicated starting position.
   (a) Tic starting from a blank board.
   (b) Pick-Up-Bricks starting with 4 bricks.
   (c) Chop starting from a 2×3 board.

(2) In each W-L-D game tree, use Procedure 1.8 to find a winning strategy for one of the players or a drawing strategy for both.

(a)

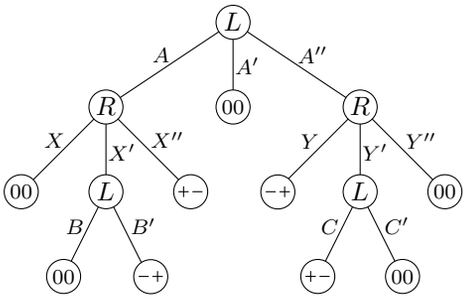


(b)

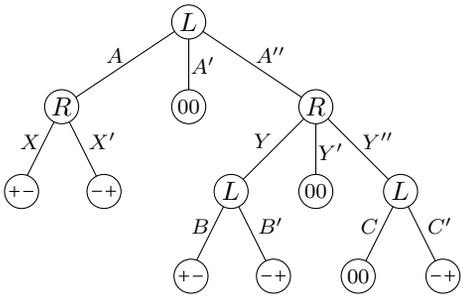


(3) For each position below, construct a game tree with Louise moving first. Then use Procedure 1.8 to find a winning strategy for one of the players.
   (a) A 5-brick position in Pick-Up-Bricks.
   (b) The Chomp position  .

(4) Consider the labeled W-L-D game tree below. To describe a strategy for Louise, we can indicate the label corresponding to each of her choices. For instance, $AB'C$ and $A''BC'$ are strategies for Louise. Similarly, XY' and $X''Y$ are strategies for Richard.
   (a) Find all of Louise's strategies.
   (b) Find all of Richard's strategies.
   (c) Find a pair of strategies, one for Louise and one for Richard, that will produce outcome $+-$. Repeat for the outcomes $-+$ and 00.



(5) As in the previous problem, we may describe strategies in the game tree below using labels so $AB'C$ is a strategy for Louise and XY'' is a strategy for Richard.
   (a) Find all of Louise's strategies.
   (b) Find all of Richard's strategies.
   (c) Which of Louise's strategies are drawing?
   (d) Which of Richard's strategies are drawing?

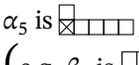
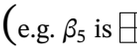


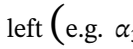
(6) Determine if each of the following games is a combinatorial game. If a given game is not a combinatorial game, explain why not.
   (a) Connect-Four.
   (b) Battleship.
   (c) Backgammon.
   (d) Go.

(7) For every pair m, n of positive integers, determine the depth of the game tree for
   (a) an m-brick position in Pick-Up-Bricks,

(b) an $m \times n$ position in Chop, and
(c) an $m \times n$ position in Chomp.

(8) Prove the following formulas by induction on n:
(a) $1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$.
(b) $1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$.

(9) For every positive integer n let α_n be the two-row Chomp position where the top row has a single square and the bottom row has n squares (e.g. α_5 is ). Let β_n be the position in Chop given by a $2 \times n$ array (e.g. β_5 is ). Prove that the game trees for the positions α_n and β_n have the same number of nodes for every $n \geq 1$.

(10) For any pair m, n of positive integers, define $\alpha_{m,n}$ to be the "L" shaped position in Chomp consisting of a column of m squares with the poison square at the bottom and a row of n squares with the poison square on the left (e.g. $\alpha_{3,6}$ is ). For all possible values of m and n find a winning strategy for either the first or second player.

(11) Let T be a W-L-D game tree with the property that every time a player has a choice to make there are exactly two options. If t is the number of terminal nodes and b is the number of branch nodes (i.e. nonterminal nodes), prove that $t = b + 1$.

(12) Consider a game tree T where Louise has n_2 nodes where she has a choice between 2 options, n_3 nodes where she has a choice between 3 options, ..., n_k nodes where she has a choice between k options (and no nodes where she chooses among more than k options). What is the formula for the total number of strategies that Louise has in T?

(13) Let T be a W-L-D game tree and assume that every time Louise has a decision to make in T she has exactly two options, and also assume that none of Richard's choices can bring the game to a terminal position. Let ℓ be the number of nodes marked L and let n be the total number of nodes. Prove that $n = 3\ell$ if Louise makes a choice at the root node, and otherwise $n = 3\ell + 1$.

(14) If N and N' are two nodes of a game tree and a player with a choice at N can move to N', then N and N' are connected by an edge (in our drawings, this edge is realized by a line segment). Prove every game tree has exactly one more node than edge.

(15) Let a and b be positive integers and let T be a W-L-D game tree with the property that whenever Louise has a choice to make she has exactly a options and whenever Richard has a choice to make he has exactly b options. Suppose that n is the total number of nodes, ℓ is the number of nodes marked L, and r is the number of nodes marked R. Prove that $a\ell + br + 1 = n$.

(16) For every integer $n \geq 1$ let c_n be the number of nodes in the game tree for a $1 \times n$ position in Chop. Find (and prove) a formula for c_n for every $n \geq 1$.

(17) The Fibonacci Sequence is an infinite sequence that starts off 0, 1, 1, 2, 3, 5, 8, 13, 21, ... It is defined by the following rules: $f_0 = 0$, $f_1 = 1$, and $f_{n+2} = f_{n+1} + f_n$ for every $n \geq 0$.
   (a) Let φ, ψ be the solutions to the equation $x^2 = x + 1$ where $\varphi > 0$ (here φ is the Golden Ratio). Prove by induction that $f_n = \frac{\varphi^n - \psi^n}{\sqrt{5}}$ for every $n \geq 0$.
   (b) For every real number x, let $[x]$ denote the closest integer to x. Prove that $f_n = \left[\frac{\varphi^n}{\sqrt{5}}\right]$ holds for every $n \geq 0$.
   (c) Let p_n be the number of nodes in a game tree for a Pick-Up-Bricks position with n bricks. Find (and prove) a formula for p_n.

(18) Consider a Chomp position that consists of just two rows, with the bottom row of length n (including the poison square) and the other row of length m and assume $m \leq n$. Determine which player has a winning strategy for all possible values of m and n and prove your answer. (Hint: Consider the special case $n = m + 1$.)

(19) The game Kayles is played with a $1 \times n$ array, each square of which is either empty or is marked by a P indicating that it contains a bowling pin. On each turn we think of a player as bowling a ball toward this line-up of bowling pins. Each player has perfect aim and may choose to knock down any one pin or any two consecutive pins (i.e. erase the P from either one box or from two adjacent boxes). The first player to clear the board wins. For every positive integer n find a winning strategy for either the first or second player when the starting position is a $1 \times n$ array where every square contains a P.

(20) Consider playing d-dimensional Tic-Tac-Toe on a board with dimensions $\underbrace{n \times n \times \cdots \times n}_{d}$. To win, a player needs to get all n boxes in some line marked with his or her symbol ($\times$ or $\circ$). Prove that for every $n, d \geq 1$ the second player does not have a winning strategy.

(21*) The game SOS is played on a $1 \times n$ array. On his turn a player may choose any empty square and mark either an "S" or an "O" in it. If a player manages to get the letters "SOS" appearing in three consecutive squares and he is the first one to achieve this, then he wins. If the game ends without this arrangement, it is a draw. Prove that from each of the following starting positions, the first player has a winning strategy.

(a) A 1×4 array where the leftmost square is marked "S" and the other three squares are empty.

(b) A blank 1×7 array.

(c) A blank $1 \times n$ array where $n \geq 7$ is odd.