Thomas de Mol

Synchronized Cherries

Bachelor thesis

July 13, 2021

Thesis supervisors: M.J.H. van den Bergh
W.A. Kosters

Leiden University
Mathematical Institute
Leiden Institute of Advanced Computer Science

Abstract

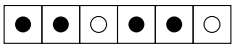
Cherries is a two-player game in which the players repeatedly remove cherries of their own color from the front or back of a segment of black and white cherries. Played as a combinatorial game (the players alternate taking turns), there is a simple method for determining the winner of any Cherries game. However, if we play Cherries as a synchronized game (the players pick their cherries at the same time), this is not the case. In this thesis, we propose a method for determining the winner of any Synchronized Cherries game by reducing it to a simpler game we call Synchronized Stack Cherries. We also present an algorithm based on this approach and analyze its complexity.

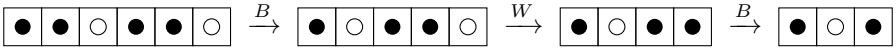
Contents

1 Introduction

Synchronized games are games in which each player simultaneously picks an action, without knowing what the other players will do. Synchronized games are most often discussed in the context of probabilities. For example, in the synchronized game *Rock paper scissors*, we consider each action to result in a one in three chance of winning, a one in three chance of losing, and a one in three chance of a draw. Using such an approach, we usually try to find a Nash equilibrium of a game in order to assign it a value. For some synchronized games, however, a probabilistic approach does not seem very useful. An example of such a game is the game *Cherries*.

Cherries is a two-player game played on one or more *segments* of black and white cherries. The players repeatedly remove cherries of their own color from the front or back of a segment. If a player has no moves available when it is their turn, they lose the game. Cherries can be played both as a synchronized game, where the players pick which cherry they want to remove at the same time, or as a *combinatorial game*, where the players alternate taking turns.

Example 1.1. Consider the Cherries game  , consisting of one segment. If we play this game combinatorially and the black player starts, the game may play out as follows:



At this point it is white's turn to move, but both outer cherries are black, so white loses.

We will see that it is relatively simple to determine the winner of any combinatorial Cherries game if both players play perfectly. For the synchronized case, however, this is not as straightforward. The goal of this thesis is to find a method for determining the winner of any Synchronized Cherries game.

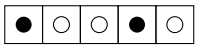
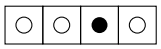
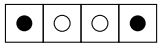
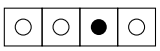
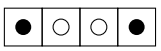
We will start in Section 2 by providing a concise introduction to the theory of combinatorial games, and use this to analyze the combinatorial version of Cherries in Section 3. In Section 4, we will properly define what a synchronized game is and discuss some of the difficulties that arise with this definition. We will also examine a subset of combinatorial games that naturally give rise to synchronized versions of these games. Section 5 contains our main results about Synchronized Cherries; we will propose a method for decomposing any Cherries game into a so-called *Stack Cherries* game and give an extensive overview of the theory of Synchronized Stack Cherries. We will apply this knowledge in Section 6 to design an algorithm for determining the winner of any Cherries game. Lastly, in Section 7 we will briefly examine some Cherries variants. Section 8 concludes.

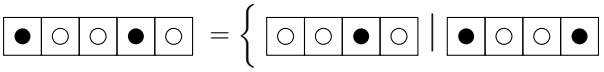
This thesis was written for the bachelor programs Mathematics and Computer Science at Leiden University, under the supervision of Mark van den Bergh (MI) and Walter Kosters (LIACS).

2 Combinatorial Games

In this section, we introduce some concepts and theorems from the field of combinatorial game theory based on [ANW19], to help us analyze the game of Cherries. The first thing we need to do to start analyzing games is properly defining what a game is. In combinatorial game theory, we look at two-player games that do not involve randomness of any kind and in which the players have perfect information about the state of the game. The players alternately take turns moving, until the player whose turn it is has no legal moves left. In this thesis, we will only consider *normal play*, where the last player to make a move wins. We will refer to the two players as Left and Right. The left player is associated with the color black and is referred to as female, while the right player gets the color white and is referred to as male.

Definition 2.1. We define a combinatorial game G as a set of left options and a set of right options. The set of left options is the, possibly empty, finite set of combinatorial games to which Left can play by making a move on the game G, and similarly for the right options. We use the notation $G = \{\mathcal{G}^L \mid \mathcal{G}^R\}$, where $\mathcal{G}^L$ is the set of left options and $\mathcal{G}^R$ the set of right options.

Example 2.2. Consider the combinatorial Cherries game $G =$  . If it is the left (black) player's turn to move, she can remove the first cherry, resulting in the game  . If it is the right (white) player's turn to move, he can remove the last cherry, resulting in the game  . Thus, $\mathcal{G}^L = \left\{ \right.$  $\left. \right\}$ and $\mathcal{G}^R = \left\{ \right.$  $\left. \right\}$. In practice, we often omit the braces around the left and right options of a game, leaving us with



Note that this definition is recursive, since we define a game as two sets of other games. This recursion does not have to go on forever, since the sets of left and right options may be empty. In fact, in this thesis, we will only be looking at games where this recursion ends at some point, or in other words games that always end after a finite amount of moves. Such games are called *short games*. We give the game with no left and right options the name zero: $0 = \{\emptyset \mid \emptyset\}$, or more concise, $0 = \{ \mid \}$. We introduce the concept of the *birthday* of a game to concretely define the finiteness of this recursion.

Definition 2.3. We define the *birthday* of the game zero to be 0. For any other game $G = \{\mathcal{G}^L \mid \mathcal{G}^R\}$, we recursively define the birthday of G as the maximum birthday of any game in $\mathcal{G}^L$ or $\mathcal{G}^R$ plus one.

In other words, we can think of the birthday of a game G as the length of the longest sequence of left and right moves from G until we have reached the empty

game 0. It follows that the birthday of a Cherries segment is equal to the length n of the segment, since we have only reached the game 0 after all n cherries have been removed.

For games with a finite birthday, if both players play perfectly, the winner can already be determined if we know which player moves first.

Theorem 2.4 (Fundamental Theorem of Combinatorial Games)**.** *Let G be a combinatorial game with finite birthday and suppose that Left moves first. Then either Left has a move that allows her to win no matter what Right does, or for every move by Left there is a responding move by Right such that he can always win, but not both.*

Proof. For every left option G^L of G, either Right has a winning move playing first or Left has a winning response to every move by Right by induction on the birthday of G^L. If for every left option G^L of G Right has a winning move playing first on G^L, then this means Right always has a winning move playing second on G. Otherwise, there exists some left option G^L of G such that Left has a winning response to every move by Right on G^L, which means that Left has a winning move on G. □

This theorem allows us to split all combinatorial games into four categories, based on which player can win when Left moves first and which player can win when Right moves first. We call this category the *outcome class* of a game, denoted by $o(G)$. We label these four classes $\mathcal{L}, \mathcal{N}, \mathcal{P}$ and $\mathcal{R}$. The class $\mathcal{L}$ consists of games where the $\mathcal{L}$eft player can always win, no matter who moves first. The class $\mathcal{N}$ contains the games where the first player to move, or the $\mathcal{N}$ext player, can always win. The class $\mathcal{P}$ contains the games where the second player to move, or the $\mathcal{P}$revious player, can always win. Lastly, the class $\mathcal{R}$ consists of the games where the $\mathcal{R}$ight player can always win. This information is summarized in Table 1.

Outcome classes		When Right moves first	
		Left Wins	Right Wins
When Left moves first	Left Wins	$\mathcal{L}$	$\mathcal{N}$
	Right Wins	$\mathcal{P}$	$\mathcal{R}$

Table 1: The four possible outcome classes of a combinatorial game

In combinatorial game theory, we are often interested in talking about multiple games played next to each other, which we will call a *sum* of games. For example, if we have a Cherries game that consists of multiple segments, we can think of this game as a sum of multiple independent components (the segments) and when it is their turn, the players pick a component and perform a move on it. This mindset of thinking about games as sums of independent components will turn out to be a very powerful in the analysis of many combinatorial games.

Definition 2.5. Let $G = \{\mathcal{G}^L \mid \mathcal{G}^R\}$ and $H = \{\mathcal{H}^L \mid \mathcal{H}^R\}$ be combinatorial games. Then we define the sum of G and H as

$$G + H = \{\mathcal{G}^L + H, G + \mathcal{H}^L \mid \mathcal{G}^R + H, G + \mathcal{H}^R\}.$$

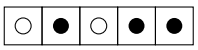
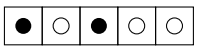
This definition might look complicated at first, but it encapsulates exactly what we described above. This definition also contains the new concept of the sum of a set of games and a single game. For example, $\mathcal{G}^L + H$, which is just shorthand notation for $\{G^L + H : G^L \in \mathcal{G}^L\}$, the set containing each game in $\mathcal{G}^L$ summed with H. So this definition tells us that the left options of $G + H$ are all the games of the form $G^L + H$ where G^L is a left option of G and all the games of the form $G + H^L$ where H^L is a left option of H. In other words, the left player picks the component G or H, makes a move on it and leaves the other component intact.

Next, we will introduce the concept of the negation of a game. Intuitively, we can think of taking the negation of a game as swapping the roles of the two players. This means that we swap the left and right options of the game and do the same for its children. This gives us the following recursive definition.

Definition 2.6. Let $G = \{\mathcal{G}^L \mid \mathcal{G}^R\}$ be a combinatorial game. We define the negation of G as

$$-G = \{-\mathcal{G}^R \mid -\mathcal{G}^L\}$$

Example 2.7. If we swap the roles of the two players in a Cherries game, this means that Left may now take white cherries and Right may now take black cherries from the front or back of a segment. Equivalently, this is equal to the game where the colors of the cherries have switched. Thus, taking the negation of a Cherries segment corresponds to flipping the color of every cherry. For example,

$-$  $=$ 

Currently we are only interested in determining which player has a winning strategy on a given game and not by how much they can win. In our mindset of thinking of games as sums of independent components, this means that we would say two games are "equal" if they always behave the same as components in some larger game. More precisely, if for any game containing the component G, replacing G with H does not change the outcome of the game, we say G and H are equal.

Definition 2.8. Let G, H be combinatorial games. We say $G = H$ if $o(G+X) = o(H + X)$ for all combinatorial games X.

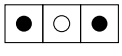
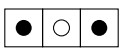
Lemma 2.9. *The equality of combinatorial games in Definition 2.8 is an equivalence relation. Furthermore, addition and negation are invariant under this equivalence relation.*

Theorem 2.10. *The equivalence classes of combinatorial games under equality form an abelian group with group operation $+$ and inverse operation $-$.*

This theorem basically tells us that the addition, negation and equality we just defined satisfy all the basic properties we would expect them to have. For example, $G + H = H + G$ and $G - G = 0$ for all combinatorial games G, H.

Trying to apply our definition of equality directly seems like a very tedious task: how do we check that the outcome class of $G+X$ is the same as that of $H+X$ for *all* games X, knowing that there are infinitely many games? Fortunately, there is another method for showing equality of games that is much more practical.

Theorem 2.11. *Let G, H be combinatorial games. Then $G = H$ if and only if $G - H \in \mathcal{P}$.*

Example 2.12. Let $G =$  and let $H =$  . Playing the game $G - H =$  $+$, we can verify that the second player to move can always win. Thus $G - H \in \mathcal{P}$ and therefore  $=$ 

In the same vein as the equality, we also want to introduce an inequality on games to indicate when one game is "better" than another for Left or Right. In combinatorial game theory, we use the convention that "greater than" means "better for the left player". With this in mind, we introduce a partial order on the four outcome classes satisfying $\mathcal{R} < \mathcal{N} < \mathcal{L}$ and $\mathcal{R} < \mathcal{P} < \mathcal{L}$. Note that with this order, only $\mathcal{N}$ and $\mathcal{P}$ are incomparable, since one outcome class is not always better for Left than the other.

Definition 2.13. Let G, H be combinatorial games. We say $G \leq H$ if $o(G+X) \leq o(H+X)$ for all combinatorial games X.

Just like the equality, this inequality also behaves like we would expect it to:

Theorem 2.14. *The inequality of combinatorial games from Definition 2.13 is a partial order and respects addition.*

One of the simplest types of games are the integer games, which are defined as follows.

Definition 2.15. Let $n > 0$ be an integer. We define the integer games n and $-n$ as

$$n = \{n - 1 \mid\}$$
$$-n = \{\mid -n + 1\}$$

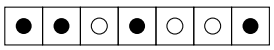
So, for positive integers n, we can think of the game n as a game where Left has n free moves while Right has none and similarly we can think of the game $-n$ as a game where Left has no moves and Right has n free moves.

The following theorem about integer games will be very helpful for the analysis of combinatorial Cherries:

Theorem 2.16 (Simplest number theorem)**.** *Let $G = \{a \mid b\}$ where a and b are integer games such that $a + 2 \leq b$. Then $G = c$ where c is the integer with the smallest absolute value satisfying $a < c < b$.*

Proof. The condition $a + 2 \leq b$ guarantees there is an integer c satisfying $a < c < b$. So let c be the integer with the smallest absolute value satisfying $a < c < b$. We will show that $c - G \in \mathcal{P}$. Suppose Left moves first on $c - G$. If she plays on the component $-G$ the resulting game is $c - b < 0$ and thus Right wins. If $c > 0$, she can also play on the component c, resulting in the game $c - 1 - G$. Right responds by playing to $c - 1 - a$. Suppose that $a < c - 1$. This means that $c - 1$ is also an integer satisfying $a < c - 1 < b$ and $c - 1$ has a smaller absolute value than c. This is a contradiction, so $a \geq c - 1$ must hold. It follows that $c - 1 - a \leq 0$ is a win for Right playing second. A similar argument show that if Right moves first on $c - G$, then Left wins. We conclude that $c - G \in \mathcal{P}$ and thus that $G = c$. □

3 Combinatorial Cherries

In this section, we will see how we can determine the outcome class and an optimal strategy for any Combinatorial Cherries game. We can look at a Cherries segment as a sequence of *blocks* of consecutive black or white cherries. For example,  starts with a black block of length 2, followed by a white block of length 1 and so on. We say a block is *internal* if it is not the first or last block in the segment. We define the sign of a black cherry to be $+1$ and the sign of white cherry to be -1.

If G is a Cherries segment that solely consists of n black cherries, then $G = n$ since Left has n free moves and Right has none. Similarly, a game that only consists of n white cherries is equal to $-n$. Any Cherries position that contains both black and white cherries also turns out to be equal to an integer.

Theorem 3.1. *Let G be a Cherries segment containing both black and white cherries. Let m be the length of the first block of cherries and n the length of the last block. Let $\ell, r \in \{-1, +1\}$ be the signs of the first respectively last blocks. If the segment contains an internal block of length greater than 1, let $x, y \in \{-1, +1\}$ be the signs of the leftmost respectively rightmost internal blocks of length greater than 1. Otherwise, let $x = y = 0$. Then*

$$G = \ell(m-1) + r(n-1) + \frac{\ell + r}{2} + \frac{x + y}{2}$$

Proof. We will prove this theorem using induction on the length of the segment of cherries. For the base case, we look at segments of cherries that consist of exactly two blocks, as these games are the smallest games containing both black and white cherries. Let G be a segment of Cherries that starts with a block of m black cherries followed by a block of n white cherries. In this case, $\ell = 1$, $r = -1$, $x = y = 0$, so we want to show that $G = m - n$. If Left removes the first black cherry of G, by induction we are left with the game $(m-1) - n = m - n - 1$. Similarly, if Right removes the last white cherry we are left with the game $m - n + 1$. Thus, using the simplest number theorem we find that $G = \{m - n - 1 \mid m - n + 1\} = m - n$.

For the induction step, let G be a segment of Cherries that consists of at least three blocks. Define $v = \ell(m-1) + r(n-1) + \frac{\ell+r}{2} + \frac{x+y}{2}$. Suppose that the first block is black ($\ell = 1$). If the first block has length greater than 1 and left removes the first cherry in the segment, the variable m gets decreased by 1 while the other variables remain the same, so the resulting game has value $v - 1$ by induction. If the first block has length 1, the second block also has length 1 and left removes the first cherry in the segment, only the variable ℓ changes from $+1$ to -1 and since $m = 1$, this means that the value of the resulting game is again $v - 1$ by induction. Lastly, if the first block has length 1, the second block has length greater than 1 and left removes the first cherry in the segment, the variable ℓ again changes from $+1$ to -1 and the value of m gets increased from

1 to at least 2. If the second block was the only inner block with length greater than 1, x and y go from -1 to 0. If not, y remains the same while x either stays as -1 or changes to $+1$ depending on the color of the second leftmost inner block of length greater than 1. In both cases, $\frac{x+y}{2}$ either stays the same or gets increased by 1. All in all, the value of $\ell(m-1)$ decreases from 0 to at most -1, the value of $\frac{\ell+r}{2}$ gets decreased by 1 and $\frac{x+y}{2}$ stays the same or gets increased by 1. So the total value of the components gets decreased by at least 1 and the resulting game is at most $v-1$ by induction.

So we see that any move made by Left is to an integer with value at most $v-1$ and we call a move by Left optimal if it is to the game $v-1$. Similarly, moves by Right are to integers with value at least $v+1$ and we call a move by Right optimal if it is to the game $v+1$. If the first and last cherries are both black, $\ell = r = 1$, so we see that $v \geq 0$. If at least one of the two moves for left is optimal, this means that $G = \{v-1 \mid\} = v$. If both moves are not optimal, this means that $m = n = 1$ and $x = y = -1$, so $v = 0$. So if Left starts, she has to move to a negative game and Right wins, and if Right starts, he has no moves available and Left wins. Thus, $G = 0 = v$. The same argument holds when the first and last cherries are both white. Lastly, assume the first cherry is black and the last cherry is white ($\ell = 1$ and $r = -1$). If the moves for both players are optimal, $G = \{v-1 \mid v+1\} = v$. If only the move for left is optimal, then $n = 1$ and $y = 1$, so $v \geq 0$. Now Left can move to $v-1$ and right to at least $v+1$, so by the simplest number theorem $G = v$. The case when only the move for Right is optimal is symmetrical. If neither moves are optimal, $m = n = 1$, $x = -1$ and $y = 1$, so $v = 0$. Now Left can only move to a negative game while Right can only move to a positive game, so $G = 0 = v$.

□

Using Theorem 3.1 and the arithmetic properties of integer games, we can calculate the value and thus the outcome of any sum of Cherries games. The proof of this theorem also outlines an optimal strategy for playing a game of Cherries, since Left wants to make a move that maximizes the value of the resulting game, while Right wants to minimize it. So for Left, taking a black cherry that is not immediately followed by a white inner block of length greater than 1 is always optimal, since it only decreases the value of the game by 1. If all outer black cherries are immediately followed by a white inner block of length greater than 1, Theorem 3.1 tells us that $G \leq 0$, so for any move by Left, Right will have a winning response.

Example 3.2. Consider the following sum of Cherries segments

$G =$  $+$ $+$

By Theorem 3.1, $G = 2 + 0 - 1 = 1$, so left can win moving first or second. The only winning starting move for left is taking the leftmost cherry from the first component, since the resulting game is equal to 0, again by Theorem 3.1.

4 Synchronized Games

So far, we have been playing games sequentially; first one player makes a move, then the other player makes a move and so on. Now we will look at synchronized games, where both players pick a move at the same time.

Definition 4.1. A synchronized game is a tuple $G = (G^L, G^R, G^S)$. Here G^L is a sequence of m synchronized games, the solo left options of G, G^R is a sequence of n synchronized games, the solo right options of G and G^S is a $m \times n$ matrix of synchronized games representing the synchronized moves of G.

When playing a synchronized game $G = (G^L, G^R, G^S)$, Left and Right pick integers $i \in \{1, \dots, m\}$ and $j \in \{1, \dots, n\}$ respectively, independently of each other. This results in the new position G^S_{ij}.

If G^S is an empty matrix (there are no more solo left options or no more solo right options), we call the game G *decided*. For a decided game, if $|G^L| > 0$, we say left wins, if $|G^R| > 0$, we say right wins and if $|G^L| = |G^R| = 0$, we call the game a draw. We define the birthday of a synchronized game similar to that of a combinatorial game.

Definition 4.2. The *birthday* of a synchronized game G is defined recursively as 1 plus the maximum birthday of any element in G^L, G^R or G^S. The birthday of the game where $|G^L| = |G^R| = 0$ is defined as 0.

In this thesis, we will only consider synchronized games with finite birthday, allowing us to prove statements using induction on the birthday of the game with the game without any options as the base case. We again call this empty game zero: $0 = ((), (), ())$.

The idea behind addition of synchronized games is the same as with combinatorial games: The players pick a component to move on and leave the other component intact. If both players move on the same component, a synchronized move is performed on this component; if the players move on different components, solo moves are performed.

Definition 4.3. Let G and H be synchronized games. We define the addition of G and H by $G + H = (X^L, X^R, X^S)$, where X^L is the concatenation of the sequences $G^L + H$ and $G + H^L$, X^R is the concatenation of the sequences $G^R + H$ and $G + H^R$, and X^S is the $|X^L| \times |X^R|$ matrix of synchronized games defined by

$$X^S_{ij} = \begin{cases} G^S_{ij} + H & \text{if } i \le |G^L| \text{ and } j \le |G^R| \\ G^L_i + H^R_{j-|G^R|} & \text{if } i \le |G^L| \text{ and } j > |G^R| \\ G^R_j + H^L_{i-|G^L|} & \text{if } i > |G^L| \text{ and } j \le |G^R| \\ G + H^S_{i-|G^L|, j-|G^R|} & \text{if } i > |G^L| \text{ and } j > |G^R| \end{cases}$$

Definition 4.4. Let G be a synchronized game. We define the negation of G as $-G = (-G^R, -G^L, -(G^S)^\intercal)$.

For integers $n > 0$, we recursively define the synchronized game $n = ((n-1), (), ())$ and for integers $n < 0$ we define $n = ((), (n+1), ())$. With the above definition of addition we can verify that, for integers $m, n \geq 0$, the sum of synchronized games $m + n$ refers to the same game as the synchronized game corresponding to the integer $m + n$. However, when $m < 0$ and $n > 0$ this is not the case, as we will see in the next subsection.

4.1 Equality of Synchronized Games

Just like with combinatorial games, we need to introduce the concept of outcome classes of synchronized games to define equality and inequality. This is, however, not as straightforward as in the combinatorial case. Since the players pick moves at the same time, independently of each other, they cannot factor in the move the other player will pick when making their decision. Take, for example, a synchronized game G with $G^S = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$. In this game, if both players pick the same move index (a move on the diagonal), we see that Left wins; if the players pick different moves, Right wins. So neither Left nor Right has a move that guarantees them a win, but the game also never ends in a draw. This means that this game has no clear winner. For now, we will group all these games without a clear winner into one class, leaving us with the following three outcome classes:

- The outcome class $\mathcal{L}$, containing games G where Left has a move i such that $G^S_{ij} \in \mathcal{L}$ for all j (a winning move for Left).
- The outcome class $\mathcal{R}$, containing games G where Right has a move j such that $G^S_{ij} \in \mathcal{R}$ for all i (a winning move for Right.
- The outcome class $\mathcal{U}$, containing games where neither player has a winning move.

Note that the symbols $\mathcal{L}$ and $\mathcal{R}$ are used both for the combinatorial games and the synchronized games where left or right has a winning strategy, but it should always follow from the context which of these we are referring to. We denote the outcome class of a synchronized game G by $o(G)$. This brings us to the definition of equality of synchronized games.

Definition 4.5. Let G, H be synchronized games. We say $G = H$ if $o(G+X) = o(H+X)$ for all synchronized games X.

With the order $\mathcal{R} < \mathcal{U} < \mathcal{L}$ on the outcome classes, we also get our definition of inequality.

Definition 4.6. Let G, H be synchronized games. We say $G \geq H$ if $o(G+X) \geq o(H+X)$ for all synchronized games X.

We need to be a bit careful with these definitions, since many nice properties that hold for combinatorial equality do not hold for the synchronized case, as can be seen in the following example.

Example 4.7. If G_c is a combinatorial game, $G_c - G_c = 0$ always holds by Theorem 2.10. For synchronized games, this is not the case. Take for example the synchronized games $G = 1$ and $X = ((-2), (-2), (2))$. We have $o(0 + X) = o(X) = \mathcal{L}$, since the only synchronized move from X is to the game 2. But on the game $1 - 1 + X$, if Left moves on the component 1 and Right on the component X, the resulting game is $-1 - 2$, so Right wins. If Left moves on the component X and Right on the component -1, the resulting game is $1 - 2$, so Right wins again. This means that Left has no move on $1 - 1 + X$ that guarantees her to win, so $o(1 - 1 + X) \neq \mathcal{L} = o(0 + X)$ and thus $1 - 1 \neq 0$. □

Conversely, sometimes synchronized equality of games holds even if we would not want it to.

Example 4.8. Let G, H be synchronized games with

$$G^S = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

$$H^S = \begin{pmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{pmatrix}$$

Since the players in a synchronized game pick their moves independently from each other, we can think of G as a game where Left and Right both have equal chances of winning, and H as a game where Left has a $\frac{1}{3}$ chance to win and Right a $\frac{2}{3}$ chance. But using our definition of equality, we can show that $G = H$.

Indeed, let X be a synchronized game. Suppose $o(G + X) = \mathcal{L}$. This means that Left has a winning move on the component G or X. If Left has a winning move i on G, this means that $o(G^S_{ij} + X) = \mathcal{L}$ for all right moves j of G and that $o(G^L_i + X^R_j) = \mathcal{L}$ for all right moves j of X. It follows that $o(1 + X) = o(-1 + X) = \mathcal{L}$ and that $o(0 + X^R_j) = \mathcal{L}$ for all right moves j of X. So, playing on the game $H + X$, if Left picks any move i on H we have $o(H^S_{ij} + X) = o(1 + X) = \mathcal{L}$ or $o(H^S_{ij} + X) = o(-1 + X) = \mathcal{L}$ for all right moves j of H and $o(H^L_i + X^R_j) = o(0 + X^R_j) = \mathcal{L}$ for all right moves j of X. Thus, this is a winning move for Left and $o(H + X) = \mathcal{L}$. In the other case, Left has a winning move i on X, so $o(G^R_j + X^L_i) = o(0 + X^L_i) = \mathcal{L}$ and $o(G + X^S_{ij}) = \mathcal{L}$ for all j. By induction on the birthday of X, $o(H + X^S_{ij}) = o(G + X^S_{ij}) = \mathcal{L}$ and $o(H^R_j + X^L_i) = o(0 + X^L_i) = \mathcal{L}$ for all j. So in both cases we see that $o(H + X) = \mathcal{L} = o(G + X)$.

Using the same argument, we can show that $o(G + X) = \mathcal{L} \iff o(H + X) = \mathcal{L}$ and $o(G + X) = \mathcal{R} \iff o(H + X) = \mathcal{R}$. It now also follows that $o(G + X) = \mathcal{U} \iff o(H + X) = \mathcal{U}$. Thus, $G = H$.

We see that most of these problems have to do with the outcome class $\mathcal{U}$. Since this class contains games where the winner is based on chance, saying that $o(G) = o(H)$ does not necessarily mean that the outcome when playing out these games will always be the same if both players play perfectly. Despite this, this definition of equality and inequality will work well for our needs. One simple but important property our synchronized equality does have is the following:

Lemma 4.9. *Let $G = (G^L, G^R, G^S)$ be a synchronized game such that $|G^L| > 0$ and $|G^R| = 0$. Then $G > 0$.*

Proof. Let X be a synchronized game. We will show that $o(X+G) \geq o(X)$ using induction on the birthday of X. For the base case, let X be a decided game. If X only has left options, we have $o(X) = \mathcal{L}$, and since G also only has left options, we conclude that $o(X+G) = \mathcal{L}$. If X only has right options, $o(X) = \mathcal{R}$, so we always have $o(X+G) \geq o(X)$. Lastly, if X has no options we have $X = 0$, so $o(X+G) = o(G) = \mathcal{L} > \mathcal{U} = o(X)$.

For the induction step, let X be non-decided. If $o(X) = \mathcal{L}$, there is a move i for Left such that $o(X^S_{ij}) = \mathcal{L}$ for all j. Since G has no right options, all synchronized moves of $X+G$ are of the form $X^S_{ij} + G$ if Left again picks move i on the component X. By induction these games are all wins for Left, so $o(X+G) = \mathcal{L}$. Next, suppose that $o(X) = \mathcal{U}$. If $o(X+G) = \mathcal{R}$, there is a move j for Right such that $o(X^S_{ij}+G) = \mathcal{R}$ for all i. By induction, $o(X^S_{ij}) \leq o(X^S_{ij}+G) = \mathcal{R}$ for all i, so $o(X) = \mathcal{R}$. This is a contradiction, so $o(X+G) \neq \mathcal{R}$ and thus $o(X+G) \geq o(X)$. Lastly, if $o(X) = \mathcal{R}$, there is nothing to show.

We have now shown that $o(X+G) \geq o(X)$ for all X, meaning that $G \geq 0$. Lastly, $o(G+0) = o(G) = \mathcal{L} > \mathcal{U} = o(0)$, and thus $G > 0$. □

Intuitively, this lemma tells us that a synchronized game G that only has left options can never hurt Left in the context of any synchronized game X, since she can always decide to ignore the component G if the moves on it are bad, and that it sometimes even provides a strict advantage for Left, for example when $X = 0$.

We will see the proof technique used in this lemma return often for showing inequalities between games; To prove that $G \geq H$, we only have to show that $o(H+X) = \mathcal{L} \implies o(G+X) = \mathcal{L}$ and that $o(G+X) = \mathcal{R} \implies o(H+X) = \mathcal{R}$ for all synchronized games X.

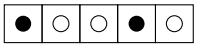
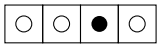
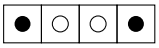
4.2 Synchronizing Combinatorial Games

The definitions of combinatorial and synchronized games are very different, but it can be interesting to try to construct a natural-seeming synchronized version of some combinatorial game and analyze it.

For a combinatorial game G, we want each pair of left and right options (G^L, G^R) to correspond to some synchronized move that represents executing both moves.

For many combinatorial games, defining a synchronized version seems fairly straightforward. Take for example the game chess. If Left wants to move one of her pieces somewhere and Right wants to move one of his pieces somewhere, the corresponding synchronized move would be to move both pieces to their desired destinations at the same time. But we quickly run into edge cases with this definition; what if both players want to move a piece to the same location? We could, for example, capture both pieces, or we could only allow Left's piece to move, but there does not seem to be one correct answer. Ideally, we want the synchronized move corresponding to G^L and G^R to be some position that is both reachable from G^L and from G^R. This gives rise to the following definition:

Definition 4.10. Let G be a combinatorial game. We call G *strongly separable* if, for all positions H of G, for all left options $H^L = \{\mathcal{H}^{LL} \mid \mathcal{H}^{LR}\}$ and right options $H^R = \{\mathcal{H}^{RL} \mid \mathcal{H}^{RR}\}$ of H, we have $\mathcal{H}^{LR} \cap \mathcal{H}^{RL} \neq \emptyset$.

Example 4.11. Let G be a combinatorial Cherries segment. Then every position H of G is again a Cherries segment. If the first and last cherries of H share the same color, this means that H either has no left options or no right options and thus satisfies the condition of being strongly separable trivially. If the first and last cherries of H have different colors, there is a left option H^L and a right option H^R of H where the first or last cherry has been removed. The game H^S, where the first and last cherries of H have both been removed, is now a right option of H^L and a left option of H^R. For example, if $H =$  , the only left option is $H^L =$  and the only right option is $H^R =$  . The game $H^S =$  where the first and last cherries have been removed is a right option of H^L and a left option of H^R. We conclude that Cherries segments are strongly separable.

An example of a combinatorial game that is not strongly separable is Nim. The game Nim is played with several piles of coins and a move for either player consists of picking a pile and removing any amount of coins. If H is a combinatorial Nim game that consists of one pile, then both Left and Right can choose to remove every coin from this pile resulting in the games $H^L = 0$ and $H^R = 0$. Since the game 0 has no left or right options, the intersection of the right options of H^L and the left options of H^R is empty, meaning that H is not strongly separable.

Lemma 4.12. *Let G and H be strongly separable combinatorial games. Then $G + H$ and $-G$ are also strongly separable.*

Proof. Any position of $G + H$ is of the form $G' + H'$, where G' is a position of G and H' is a position of H. A left option of $G' + H'$ is of the form $G'^L + H'$ or $G' + H'^L$ and a right option of $G' + H'$ is of the form $G'^R + H'$ or $G' + H'^R$. For a left option $G'^L + H'$ and a right option $G'^R + H'$, the intersection of the right options of G'^L and the left options of G'^R contains a game G'^S since G is strongly separable. So the intersection of the right options of $G'^L + H'$ and

the left options of $G'^R + H'$ contains the game $G'^S + H'$ and thus is not empty. For a left option $G'^L + H'$ and a right option $G' + H'^R$, the intersection of the right options of $G'^L + H'$ and the left options of $G' + H'^R$ contains the game $G'^L + H'^R$ and thus is not empty. The other cases are analogous. Thus, $G + H$ is strongly separable.

Next, we will look at $-G$. Positions of this game are of the form $-G'$ where G' is a position of G. For all left options $-G'^L$ and right options $-G'^R$ of $-G'$, the game G'^L is a right option of G' and G'^R is a left option of G' by the definition of the negation of combinatorial games. Since G is strongly separable, the intersection of the left options of G'^L and the right options of G'^R contains some game G'^S. Again by the definition of negation, the intersection of the right options of $-G'^L$ and the left options of $-G'^R$ now contains the game $-G'^S$ and thus is not empty. We conclude that $-G$ is strongly separable. □

For a strongly separable game G, it seems sensible to define the synchronized move corresponding to some left option G^L and right option G^R to be some sort of synchronized version of a game in the non-empty intersection of the right options of G^R and the left options of G^L, which brings about the following definition:

Definition 4.13. Let G_c be a strongly separable combinatorial game and G a synchronized game. We say G is a *synchronized version* of G_c if there are bijections $g^L \colon \{1, \ldots, |G^L|\} \to \mathcal{G}_c^L$ and $g^R \colon \{1, \ldots, |G^R|\} \to \mathcal{G}_c^R$ such that

1. G_i^L is a synchronized version of $g^L(i)$ for $1 \leq i \leq |G^L|$
2. G_j^R is a synchronized version of $g^R(j)$ for $1 \leq j \leq |G^R|$
3. G_{ij}^S is a synchronized version of some game in the intersection of the right options of $g^L(i)$ and the left options of $g^R(j)$ for $1 \leq i \leq |G^L|$ and $1 \leq j \leq |G^R|$

Lemma 4.14. *Let G_c, H_c be strongly separable combinatorial games and let G, H be synchronized versions of G_c and H_c respectively. Then $G + H$ is a synchronized version of $G_c + H_c$ and $-G$ is a synchronized version of $-G_c$.*

Proof. Let g^L, g^R be the left and right option bijections between G and G_c and h^L, h^R the left and right option bijections between H and H_c. Define $f^L \colon \{1, \ldots, |G^L| + |H^L|\} \to (\mathcal{G}_c^L + H_c) \cup (G_c + \mathcal{H}_c^L)$ with

$$f^L(i) = \begin{cases} g^L(i) + H & \text{if } i \leq |G^L| \\ G + h^L(i - |G^L|) & \text{if } i > |G^L| \end{cases}$$

Define $f^R \colon \{1, \ldots, |G^R| + |H^R|\} \to (\mathcal{G}_c^R + H_c) \cup (G_c + \mathcal{H}^R)$ analogously. Using induction and the definition of addition, we can verify that f^L and f^R are left

and right option bijections between $G + H$ and $G_c + H_c$ satisfying all three conditions of Definition 4.13. Thus, $G + H$ is a synchronized version of $G_c + H_c$.

Similarly for $-G$, define $f^L \colon \{1, \ldots, |G^R|\} \to -\mathcal{G}_c^R$ with $f^L(i) = -g^R(i)$ and $f^R \colon \{1, \ldots, |G^L|\} \to -\mathcal{G}_c^L$ with $f^R(i) = -g^L(i)$. Then, for all $1 \leq i \leq |G^R|$, $(-G)_i^L = -(G_i^R)$ is a synchronized version of $-g^R(i) = f^L(i)$ by induction and the definition of negation. The proof for condition 2 of Definition 4.13 is analogous. And lastly, for all $1 \leq i \leq |G^R|$ and $1 \leq j \leq |G^L|$, $(-G)_{ij}^S = -(G_{ji}^S)$ is a synchronized version of some game in the intersection of the right options of $-g^R(i) = f^L(i)$ and the left options of $-g^L(i) = -f^R(i)$, again by induction and the definition of negation. We conclude that $-G$ is a synchronized version of $-G_c$. □

Example 4.15. In Example 4.11 we saw that Cherries is a strongly separable game. Furthermore, for segments with differently colored outer cherries we saw that the the intersection of the right options of their single left option and the left options of their single right option contains the game where both outer cherries have been removed. This naturally leads to a recursive definition of a synchronized version of a Cherries segment; solo options correspond to synchronized versions of the segment where a single outer cherry has been removed, synchronized options correspond to synchronized versions of the segment where the cherries of both solo options have been removed. By Lemma 4.14 this also extends to the definition of a synchronized version for any sum of Cherries segments.

The outcome of a strongly separable combinatorial game can sometimes tell us a lot about the outcome of its synchronized version.

Theorem 4.16. *Let G_c be a strongly separable combinatorial game and G be a synchronized version of G_c. Then*

1. *If $G_c \in \mathcal{L}$, then $G \in \mathcal{L}$.*
2. *If $G_c \in \mathcal{R}$, then $G \in \mathcal{R}$.*
3. *If $G_c \in \mathcal{P}$, then G can be an element of $\mathcal{L}$, $\mathcal{R}$ or $\mathcal{U}$.*
4. *G_c cannot be an element of $\mathcal{N}$.*

Proof.

1. If $G_c \in \mathcal{L}$, there is a left option G_c^L of G_c that is a win for Left moving second. If G_c has no right options, G also has no right options. Since G does have left options, this means that $G \in \mathcal{L}$. Now suppose G_c does have right options. Playing on G, if Left picks the option i corresponding to the combinatorial option G_c^L, then for any move j from Right G_{ij}^S is a synchronized version of G_c^{LR} for some right option G_c^{LR} of G_c^L by the definition of a synchronized version of a game. Since G_c^L is a win for Left moving second, G_c^{LR} is a win for Left moving first. Using item 4, G_c^{LR}

cannot be an element of $\mathcal{N}$. So G_c^{LR} must be an element of $\mathcal{L}$, and using induction it follows that Left has a winning strategy on G_{ij}^S. Thus, $G \in \mathcal{L}$.

2. Symmetric to 1.

3. We will show three examples of combinatorial games in $\mathcal{P}$ with synchronized versions in $\mathcal{L}$, $\mathcal{R}$ and $\mathcal{U}$.

   - If G_c is the combinatorial game $\{-1 \mid \}$, then $G_c \in \mathcal{P}$ since the second player to move wins. The synchronized version of G_c is $G = ((-1), (), ())$. This game only has left options, so $G \in \mathcal{L}$
   - If G_c is the combinatorial game $\{ \mid 1\}$, then again $G_c \in \mathcal{P}$. The synchronized version of G_c is $G = ((), (1), ())$. This game only has right options, so $G \in \mathcal{R}$
   - If G_c is the combinatorial game $0 \in \mathcal{P}$, then $G = ((), (), ()) \in \mathcal{U}$.

4. If $G_c \in \mathcal{N}$, there is a left option G_c^L of G_c that is a win for Left moving second and a right option G_c^R of G_c that is a win for Right moving second. Since G_c is strongly separable, the intersection of the right options of G_c^L and the left options of G_c^R contains at least one element, say G_c^S. G_c^S must be a win for Left moving first, because G_c^L is a win for Left moving second. But G_c^S must also be a win for Right moving first, because G_c^R is a win for Right moving second. This is a contradiction, so G_c cannot be an element of $\mathcal{N}$.

□

5 Synchronized Cherries

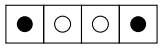
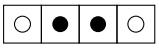
In this section we will analyze Cherries played as a synchronized game. The first task in tackling this problem is properly defining the synchronized options of this game. The obvious choice for the synchronized option corresponding to Left removing some black cherry and Right removing some white cherry would be the game where both of these cherries have been removed. In Example 4.15 we saw that defining the synchronized options like this results in a synchronized version of the combinatorial Cherries game satisfying Definition 4.13. This allows us to use Theorem 4.16 to help us determine the outcome class of the corresponding synchronized game.

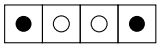
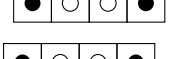
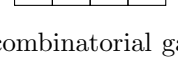
Example 5.1. Consider the synchronized Cherries game

$G =$  $+$

By Theorem 3.1, the corresponding combinatorial game is equal to $1+0=1 \in \mathcal{L}$. So Theorem 4.16 tells us that G is also a win for Left and any winning move on the corresponding combinatorial game is also a win on the synchronized game.

The only case where Theorem 4.16 does not help us in determining the winner of a synchronized Cherries game is when the corresponding combinatorial game is an element of $\mathcal{P}$. We can show that in this case all three synchronized outcome classes can actually occur. From Theorem 3.1 we know that , , $\in \mathcal{P}$, but

 $\in \mathcal{U}$,  $\in \mathcal{L}$,  $\in \mathcal{R}$.

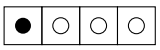
The position  turns out to be quite interesting. Since this segment starts and ends with a black cherry, Lemma 4.9 tells us that this segment can only help Left in the context of any synchronized game, but the advantage it provides seems to be infinitesimally small;  has combinatorial value 0, so no matter how many copies of  we have, if we add one to the game, the corresponding combinatorial game will have value -1 and thus the synchronized game will be a win for Right. More generally, any synchronized Cherries segment G seems to be infinitesimally close to its combinatorial integer value a, in the sense that for any $n \in \mathbb{N}_{>0}$ the synchronized game $n \cdot G - n \cdot a - 1$ is a win for Right since the corresponding combinatorial game has value -1 and similarly $n \cdot G - n \cdot a + 1$ is a win for Left.

So how do we deal with this? A good first step would be to look at what intuitively feels like a good strategy for playing synchronized Cherries. Just like

with the combinatorial version, it seems to make sense to pick cherries that reveal the least amount of cherries for the other player, since we are trying to make them run out of moves. More specifically, moves that do not reveal any of the enemy's cherries seem to be optimal in any situation. We refer to such cherries as *free*.

Example 5.2. The most obvious example of a free cherry is an outer cherry that is directly followed by another cherry of that same color. For example, the first cherry in  is free because removing it does not reveal any white cherries.

Another example of a free cherry is the first cherry of  . Although it is followed by a cherry with a different color, this white cherry is already revealed so removing the first cherry does not reveal any new white cherries.

More generally, we also consider a black cherry only followed by some amount of white cherries to be free. For example, we consider the first cherry of  to be free. Although it is followed by a differently colored cherry that is not an outer cherry, this cherry is still part of an outer block and therefore we consider it revealed from the start.

The following lemma justifies the connotation of optimality that the word *free* brings with it:

Lemma 5.3. *Let G be a sum of synchronized Cherries segments. Number the outer black cherries $b_1, b_2, ..., b_\ell$ and number the outer white cherries $w_1, w_2, ..., w_k$. Let $G \setminus A$ be the game where all cherries in the set A have been removed from G. If b_1 is free, then $o(G \setminus \{b_1, w_j\}) \geq o(G \setminus \{b_i, w_j\})$ for all $1 < i \leq \ell$ and $1 \leq j \leq k$.*

Proof. Let $1 < i \leq \ell$ and $1 \leq j \leq k$. Suppose that $G \setminus \{b_1, w_j\} \in \mathcal{R}$. This means there is some white cherry w such that $G \setminus \{b_1, w_j, b, w\} \in \mathcal{R}$ for all outer black cherries b of $G \setminus \{b_1, w_j\}$. In particular, we have $G \setminus \{b_1, w_j, b_i, w\} \in \mathcal{R}$. Note that w must either be revealed from the start or by removing w_j since b_1 is free. In the game $G \setminus \{b_i, w_j\}$, the cherry b_1 is still free, so by induction we may assume Left will take this cherry next. The white cherry w is also revealed in $G \setminus \{b_i, w_j\}$ since we saw that it is revealed from the start or by removing w_j. Right decides to take this cherry, resulting in the game $G \setminus \{b_i, w_j, b_1, w\} \in \mathcal{R}$. Thus, $G \setminus \{b_i, w_j\} \in \mathcal{R}$.

Similarly, suppose that $G \setminus \{b_i, w_j\} \in \mathcal{L}$. This game still contains the free cherry b_1, so by induction we have $G \setminus \{b_i, w_j, b_1, w\} \in \mathcal{L}$ for all white outer cherries w of $G \setminus \{b_i, w_j\} \in \mathcal{L}$. In the game $G \setminus \{b_1, w_j\}$, Left decides to take the cherry b_i and white takes some cherry w. The outer white cherries of $G \setminus \{b_1, w_j\}$ are a subset of the outer white cherries of $G \setminus \{b_i, w_j\}$ since b_1 is free, so the resulting game is $G \setminus \{b_1, w_j, b_i, w\} \in \mathcal{L}$. We conclude that $G \setminus \{b_1, w_j\} \in \mathcal{L}$. □

This lemma tells us that in a game of synchronized Cherries, both players will always take a free cherry when available. It follows that $\boxed{\bullet}\boxed{\circ} = 0$ in the context of any synchronized Cherries game, since we may assume that both players will instantly take their free cherry from the component $\boxed{\bullet}\boxed{\circ}$. Similarly, it also follows that $\boxed{\bullet}\boxed{\circ}\boxed{\circ} = \boxed{\circ}$ in the context of any Cherries game. If we now apply this to the Cherries segment $\boxed{\bullet}\boxed{\circ}\boxed{\circ}\boxed{\bullet}$ we discussed earlier, we get

$$\begin{aligned}\boxed{\bullet}\boxed{\circ}\boxed{\circ}\boxed{\bullet} &= \Big(\Big(\boxed{\circ}\boxed{\circ}\boxed{\bullet}, \boxed{\bullet}\boxed{\circ}\boxed{\circ}\Big), (), ()\Big)\\ &= \Big(\Big(\boxed{\bullet}\boxed{\circ}\boxed{\circ}\Big), (), ()\Big)\\ &= \Big(\Big(\boxed{\circ}\Big), (), ()\Big)\end{aligned}$$

In other words, Left can start off by taking an outer cherry, after which we are left with a game that is equal to a single white cherry in the context of any synchronized Cherries game.

To better model this situation, we introduce a variant of Cherries called *Stack Cherries*. The only difference with regular Cherries is that cherries may only be taken from the front of a Stack Cherries segment and not the back. We denote a Stack Cherries segment by a triangle pointing to the front of the segment, indicating that cherries may only be taken from that side. For example,

$$\triangleright\boxed{\bullet}\boxed{\circ}\boxed{\circ}\boxed{\bullet}\boxed{\circ} = \Big(\Big(\triangleright\boxed{\circ}\boxed{\circ}\boxed{\bullet}\boxed{\circ}\Big), (), ()\Big)$$

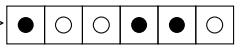
So in the context of any Cherries game, we get

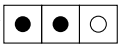
$$\boxed{\bullet}\boxed{\circ}\boxed{\circ}\boxed{\bullet} = \Big(\Big(\boxed{\circ}\Big), (), ()\Big) = \Big(\Big(\triangleright\boxed{\circ}\Big), (), ()\Big) = \triangleright\boxed{\bullet}\boxed{\circ}$$

So the Cherries segment $\boxed{\bullet}\boxed{\circ}\boxed{\circ}\boxed{\bullet}$ can be reduced to the simpler Stack Cherries segment $\triangleright\boxed{\bullet}\boxed{\circ}$. Other more complex Cherries segments also seem to be modelled well by a sum of simpler Stack Cherries segments. For this reason we will first try to fully understand Stack Cherries before moving on to the more complex regular Cherries.

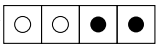
5.1 Stack Cherries

In this section we will give an extensive overview of the theory of synchronized Stack Cherries. We will see that this game, which might look trivial at first

glance, actually has some very interesting properties. We start off by introducing some notation we will use throughout this section. If G is a Stack Cherries segment, the game G_k represents the Stack Cherries segment where the front k cherries have been removed. We will also occasionally use the notation $G[a...b]$ to refer to the Stack Cherries segment starting at the a-th cherry of G and ending with the b-th cherry. For example, if $G = \triangleright$  , we have

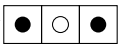
$$G_3 = \triangleright$$ 

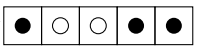
and

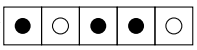
$$G[2...5] = \triangleright$$ 

Using Lemma 4.9, we know that $G > 0$ for any Stack Cherries segment G starting with a black cherry and that $H < 0$ for any Stack Cherries segment H starting with a white cherry. We extend this idea by introducing a *lexicographic order* on Stack Cherries segments. To compare two different segments lexicographically, we first pad the back of the shorter segment with empty squares until they have the same length. A square with a black cherry is greater than an empty square, which is greater than a square with a white cherry. We then look at the leftmost square where the two segments differ and we say the segment that has a greater square at this position is lexicographically greater than the other segment. If the Stack Cherries segment G is lexicographically greater than the Stack Cherries segment H, we notate this by $G \succ H$.

Example 5.4. Consider the following Stack Cherries segments:

$$G = \triangleright$$ 

$$H = \triangleright$$ 

$$J = \triangleright$$ 

Here, G is lexicographically greater than H, since the first position the two differ is at index 3, where G has a black cherry and H a white cherry.

Similarly, G is lexicographically less than J, since the first position the two differ is at index 4, where G has an "empty square" and H a black cherry.

Note that this lexicographic order is a total order, and thus by transitivity H is also lexicographically less than J.

Theorem 5.5. *Let G and H be Stack Cherries segments. If G is lexicographically greater than H, then $G > H$.*

Proof. If G starts with a black cherry and H starts with a white cherry, by Lemma 4.9 we have $G > 0 > H$. Similarly, if G or H is the empty segment this

lemma also tells us that $G > H$. So the only case left to check is when G and H both start with a cherry of the same color.

Assume without loss of generality that G and H both start with a black cherry. Let X be a synchronized game. We will show that $o(X+G) \geq o(X+H)$ using induction on the birthday of X. For the base case, let X be a game without right options. Since G and H also do not have right options, this means that $o(X+G) = o(X+H) = \mathcal{L}$.

For the induction step, let X be a synchronized game with at least one right option. If $X + H \in \mathcal{L}$, Left has a winning move on $X + H$. If this move is on the component X, there is an i such that $X^S_{ij} + H \in \mathcal{L}$ for all j. By induction it follows that $X^S_{ij} + G \in \mathcal{L}$ for all j, and since right has no options on G, this means that $X + G \in \mathcal{L}$. If Left's winning move is on the Stack Cherries segment H, then $X^R_j + H_1 \in \mathcal{L}$ for all j. Since G is lexicographically greater than H and they both start with a cherry of the same color, G_1 is lexicographically greater than H_1. Thus, by induction $X^R_j + G_1 \in \mathcal{L}$ for all j and therefore $X + G \in \mathcal{L}$.

If $X + G \in \mathcal{R}$, Right has a winning move on $X + G$. Since Right has no moves on G, this means there is a j such that $X^S_{ij} + G \in \mathcal{R}$ for all i and $X^R_j + G_1 \in \mathcal{R}$. By induction, $X^S_{ij} + H \in \mathcal{R}$ for all i, and since G_1 is lexicographically greater than H_1 we also have $X^R_j + H_1 \in \mathcal{R}$. So $X + H \in \mathcal{R}$.

So $o(X+G) \geq o(X+H)$ for all X, meaning that $G \geq H$. Lastly, since G is lexicographically greater than H we have $o(G-H) = \mathcal{L}$ and $o(H-H) = \mathcal{U}$, so strict equality does not hold. We conclude that $G > H$. □

Lemma 5.6. *Let G be any Stack Cherries segment and let H be a Stack Cherries segment that starts with a black cherry. Let GH be the concatenation of the segments G and H. Then $G + H \geq L$.*

Proof. Let X be a synchronized game. We will show that $o(X+G+H) \geq o(X+L)$ using induction on the birthday of G and X. If G is the empty segment, this means that $G = 0$ and $L = H$, so $G + H = L$.

Now let G contain at least one cherry. If $X + L \in \mathcal{L}$, Left has a winning move on the component X or L. If Left has a winning move on L, this means that $X^R_j + L_1 \in \mathcal{L}$ for all j. By induction, $X^R_j + G_1 + H \in \mathcal{L}$. Since H starts with a black cherry, this means that $X + G + H \in \mathcal{L}$. If Left has a winning move i on X, then $X_{ij} + L \in \mathcal{L}$ for all j. By induction, $X_{ij} + G + H \in \mathcal{L}$ for all j. If L starts with a white cherry, we also have $X^L_i + L_1 \in \mathcal{L}$, so again using induction we also have $X^L_i + G_1 + H \in \mathcal{L}$. Thus, $X + G + H \in \mathcal{L}$.

If $X + G + H \in \mathcal{R}$, Right has a winning move on the component X or G, since H starts with a black cherry. If Right has a winning move on G, by induction $X^L_i + L_1 \leq X^L_i + G_1 + H \in \mathcal{R}$ for all i. Since G starts with a white cherry, L also starts with a white cherry, so $X + L \in \mathcal{R}$. If Right has a winning move j on X, by induction $X^S_{ij} + L \leq X^S_{ij} + G + H \in \mathcal{R}$. If L starts with a black cherry, G also starts with a black cherry and by induction $X^R_j + L_1 \leq X^R_j + G_1 + H \in \mathcal{R}$. Thus, $X + L \in \mathcal{R}$.

We conclude that $o(X+G+H) \geq o(X+L)$ for all synchronized games X, so $G+H \geq L$. □

In other words, this lemma tells us that blocking a segment that starts with a black cherry behind another segment can only hurt Left. In particular, if G is a Stack Cherries segment where the $(k+1)$-st cherry is black, we find that $G[1...k] + G_k \geq G$

Theorem 5.7. *Let G and H be Stack Cherries segments of length at least k where the first k cherries of both segments match. If $G > H$, then $G_i + H \geq G + H_i$ for all $1 \leq i \leq k$.*

Proof. We will first look at the case $i = k$. G and H cannot both have exactly length k, since that would mean that $G = H$. We may assume without loss of generality that G has length greater than k, since otherwise we could take the negation of G and H. Since $G > H$ and G and H only match for the first k cherries, the cherry at position $k+1$ in G must be black. Lemma 5.6 now tells us that $G_k + G[1...k] \geq G$. If H has length exactly k, this means that $H = G[1...k]$ and $H_k = 0$, so $G_k + H = G_k + G[1...k] \geq G = G + H_k$. If H has length greater than k, the cherry at position $k+1$ in H must be white, since G and H do not match at position $k+1$. So Lemma 5.6 tells us that $H \geq H[1...k] + H_k$. It follows that

$$G_k + H \geq G_k + H[1...k] + H_k = G_k + G[1...k] + H_k \geq G + H_k$$

Now we will look at the case $1 \leq i < k$. Let X be a synchronized game. We will show that $o(X + G_i + H) \geq o(X + G + H_i)$ using induction on the birthday of X, on k and on $k - i$. In the previous paragraph we handled the case $k - i = 0$, and the case $k = 0$ is trivial. First, suppose $X + G + H_i \in \mathcal{L}$, meaning Left has a winning move on the component X, G or H_i.

- If Left has a winning move i on X, then by induction $X^S_{ij} + G_i + H \geq X^S_{ij} + G + H_i$ for all j. If the first cherry of H is white, this also means the first cherry of G is white, so $X^L_i + G_1 + H_i \in \mathcal{L}$ since i is a winning move for Left. By induction on k, $X^L_i + G_i + H_1 \in \mathcal{L}$. Similarly, if the first cherry of G_i is white, this also means the first cherry of H_i is white, so $X^L_i + G + H_{i+1} \in \mathcal{L}$. By induction on $k - i$, $X^L_i + G_{i+1} + H \in \mathcal{L}$. Thus, $X + G_i + H \in \mathcal{L}$.
- If Left has a winning move on G, then by induction $X^R_j + G_i + H_1 \geq X^R_j + G_1 + H_i \in \mathcal{L}$ for all j. If the first cherry of G_i is white, by induction on k also $X + G_{i+1} + H_1 \geq X + G_1 + H_{i+1} \in \mathcal{L}$. Thus, $X + G_i + H \in \mathcal{L}$.
- Lastly, if Left has a winning move on H_i, then by induction $X^R_j + G_{i+1} + H \geq X^R_j + G + H_{i+1} \in \mathcal{L}$ for all j. If the first cherry of H is white, by induction on k also $X + G_{i+1} + H_1 \geq X + G_1 + H_{i+1} \in \mathcal{L}$. Thus, $X + G_i + H \in \mathcal{L}$.

In all cases, we see that $X + G_i + H \in \mathcal{L}$. If $X + G_i + H \in \mathcal{R}$, we can use a symmetric argument to show that $X + G + H_i \in \mathcal{R}$. We conclude that $o(X + G_i + H) \geq o(X + G + H_i)$ for all synchronized games X. Thus, $G_i + H \geq G + H_i$ for $1 \leq i \leq k$. □

This theorem tells us that, playing on a sum of Stack Cherries segments, we may assume that Left will move first on a segment starting with a black cherry that is lexicographically greatest and that Right will move first on a segment starting with a white cherry that is lexicographically smallest.

With this, we can determine the outcome of any sum of Stack Cherries games by repeatedly playing on the lexicographically greatest component for Left and the lexicographically smallest component for Right until one of the players has no moves left. This is already much better than the brute force method of determining the outcome, which entailed going through an exponential amount of games. But this method is still not very insightful for determining the total value of a sum of Stack Cherries segments, so we will keep going.

To prove more specialized results about Stack Cherries, our current definition of equality is not sufficient. We for example want to show that for any Stack Cherries segment G, adding $G - G$ to a sum of Stack Cherries segments does not change the outcome, but in Section 4 we saw that $G - G = 0$ does not hold in general. To deal with this we introduce a restricted version of equality specifically for Stack Cherries games.

Definition 5.8. Let G, H be synchronized games. We say $G =_{SC} H$ if $o(G + X) = o(H + X)$ for any sum of Stack Cherries segments X.

The condition for this equality is less strict than that of general equality. So if we have $G = H$, then $G =_{SC} H$ also holds. We analogously define the inequality operator $\geq_{SC}$.

Lemma 5.9. *Let G be a Stack Cherries segment. Then $G - G =_{SC} 0$.*

Proof. Let X be a sum of Stack Cherries segments. We will show that $o(G - G + X) = o(X)$ using induction on the birthday of G and X. Suppose that $G - G + X \in \mathcal{L}$. If Left has a winning move by removing the first cherry from G, this means that $G_1 - G_1 + X \in \mathcal{L}$. By induction on the birthday of G, $o(X) = o(G_1 - G_1 + X) = \mathcal{L}$. Similarly, if Left has a winning move on $-G$ we can again show that $o(X) = \mathcal{L}$. Lastly, if Left has a winning move i on the component X, then $G - G + X^S_{ij} \in \mathcal{L}$ for all j, so by induction on the birthday of X we have $o(X^S_{ij}) = o(G - G + X^S_{ij}) = \mathcal{L}$ for all j. Thus, we also have $o(X) = \mathcal{L}$.

Conversely, suppose that $X \in \mathcal{L}$ with winning move i. If in the game $G - G + X$ the lexicographically smallest segment starting with a white cherry is in X, we may assume that Right will play on this segment with move number j. By induction on the birthday of X, $o(G - G + X^S_{ij}) = o(X^S_{ij}) = \mathcal{L}$ since i is a winning move for Left on X. Thus, $G - G + X \in \mathcal{L}$. If in the game $G - G + X$ the

lexicographically smallest segment starting with a white cherry is G, we may assume that Right will play on this segment. By induction on the birthday of G, $o(G_1 - G_1 + X) = o(X) = \mathcal{L}$, so playing on the segment $-G$ is a winning move for Left and $G - G + X \in \mathcal{L}$. The same argument holds when the lexicographically smallest segment is $-G$.

We have now shown that $G - G + X \in \mathcal{L} \iff X \in \mathcal{L}$. A symmetric argument shows that $G - G + X \in \mathcal{R} \iff X \in \mathcal{R}$. We conclude that $G - G =_{SC} 0$. $\square$

Using Lemma 4.9, we know that for all non-empty Stack Cherries segments G, either $G > 0$ or $G < 0$ holds (and the empty segment is equal to 0). This allows us to naturally define the absolute value of a Stack Cherries segment:

$$|G| = \begin{cases} G & \text{if } G \geq 0 \\ -G & \text{if } G < 0 \end{cases}$$

So taking the absolute value of a Stack Cherries segment corresponds to flipping the color of every cherry if the first one is white. For Stack Cherries segments G, H we now say G is *(lexicographically) stronger* than H if $|G|$ is lexicographically greater than $|H|$, or equivalently if $|G| > |H|$. Similarly, we say G is *(lexicographically) weaker* than H if $|G| < |H|$. We will take a closer look at a subset of Stack Cherries segments that satisfy some interesting properties.

Definition 5.10. Let G be a Stack Cherries segment of length n. We say G is a *basis element* if $|G| < |G_k|$ for all $0 < k < n$.

So to show that a Stack Cherries segment G is a basis element, we need to check that G is lexicographically weaker than all of its strict, non-empty suffixes. Since the players want to move on the strongest component starting with their color, this means that on a basis element G, every move after the initial move is "better" than this initial move. What this means in practice, is that once the first cherry of a basis element has been taken, the rest will generally follow soon after.

Example 5.11. The games $\triangleright[\bullet]$, $\triangleright[\circ|\bullet|\bullet]$ and $\triangleright[\bullet|\circ|\circ|\bullet]$ are all basis elements. The game $G = \triangleright[\bullet|\circ|\bullet|\bullet]$ is not a basis element, since

$$|G_1| = \left|\triangleright[\circ|\bullet|\bullet]\right| = \triangleright[\bullet|\circ|\circ] < |G|$$

The following supporting lemmas will help us prove an important theorem about basis elements.

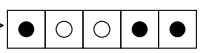
Lemma 5.12. *Let G be a basis element such that the $(\ell+1)$-st cherry of $|G|$ is white for some $\ell > 0$. Then $|G|$ and $|G_\ell|$ cannot match for more than ℓ cherries.*

Proof. We will prove this statement using contradiction. Assume $|G|$ and $|G_\ell|$ match for at least $\ell + 1$ cherries. If the second cherry of $|G|$ is white, this means the second cherry of $|G_\ell|$ is also white. Since the $(\ell + 1)$-st cherry of $|G|$ is white, G and G_ℓ start with different color cherries. So the $(\ell + 2)$-nd cherry of $|G|$, corresponding to the second cherry of $|G_\ell|$, is black. G is a basis element, so $|G| < |G_\ell|$ must hold. This means that the $(\ell + 2)$-nd cherry of $|G_\ell|$ must also be black.

Now suppose the second cherry of $|G|$ is black. Since $|G|$ and $|G_\ell|$ match for the first $\ell + 1$ cherries and the $(\ell + 1)$-st cherry of $|G|$ is white, the $(\ell + 1)$-st cherry of $|G_\ell|$ must also be white. Since G and G_ℓ start with different colors, this means that the $(2\ell + 1)$-st cherry of $|G|$ must be black. Now the second cherry of $|G_{2\ell}|$ must also be black, since otherwise $|G| < |G_{2\ell}|$ would not hold. Equivalently, this means that the $(\ell + 2)$-nd cherry of $|G_\ell|$ is white, just like the $(\ell + 2)$-nd cherry of $|G|$.

Thus, in both cases the $|G|$ and $|G_\ell|$ match for the $(\ell + 2)$-nd cherry. Using the same argument, they now also have to match for the $(\ell + 3)$-rd cherry, for the $(\ell + 4)$-th cherry and so on. We conclude that $|G|$ and $|G_\ell|$ match for infinitely many cherries, meaning that G is infinitely long. This is a contradiction, since we only consider Stack Cherries segments with finite length. □

Example 5.13. There are no basis elements starting with

$G = \triangleright$  $\ldots$

since the third cherry of $|G|$ is white and $|G|$ and $|G_2|$ match for at least 3 cherries. □

Lemma 5.14. *Let G be a basis element of length n. Then $|G_\ell| \geq |G[1...\ell]|$ for all $0 < \ell < n$.*

Proof. Suppose the $(\ell + 1)$-st cherry of $|G|$ is black. This means that $|G| > |G[1...\ell]|$. Since G is a basis element, it follows that $|G_\ell| > |G| > |G[1...\ell]|$.

Now suppose the $(\ell + 1)$-st cherry of $|G|$ is white. If $|G|$ and $|G_\ell|$ do not match for the first ℓ cherries, then $|G_\ell| > |G[1...\ell]|$ since the mismatch happens before the end of $G[1...\ell]$. If $|G|$ and $|G_\ell|$ do match for the first ℓ cherries, they cannot match at the $(\ell + 1)$-st cherry be Lemma 5.12. Since the $(\ell + 1)$-st cherry of $|G|$ is white, this means that the $(\ell + 1)$-st cherry of $|G_\ell|$ is either black or the end of the segment. In both cases, it follows that $|G_\ell| \geq |G[1...\ell]|$. □

Theorem 5.15. *Let G, H be basis elements with $G > H > 0$. Then $G >_{SC} k \cdot H$ for all $k \in \mathbb{N}_{\geq 1}$*

Proof. By Lemma 5.9, it suffices to show that $G - k \cdot H >_{SC} 0$. The base case $k = 1$ is true by assumption. Now let $k > 1$ and X be a sum of Stack Cherries games. Consider playing on the game $G - k \cdot H + X$. If the lexicographically

strongest component starting with a white cherry is a H, Right will play on this component. Left decides to move on the component G, resulting in the game $G_1 - H_1 - (k-1) \cdot H + X$. Since G and H are both basis elements and $G > H$, all strict, non-empty suffixes of G and H are lexicographically stronger than H. And since H was the lexicographically strongest component starting with a white cherry in the original game, this means that Right will immediately take a cherry from a suffix of G or H if it starts with a white cherry. Since $G > H > 0$, at least one of G_1 and $-H_1$ has to start with a black cherry. So while G_i and $-H_i$ start with different colors, we just saw that Right will play on the component starting with a white cherry and Left decides to play on the other component.

Repeating this process, eventually we will reach the game $G_\ell - H_\ell - (k-1) \cdot H + X$ where either both G_ℓ and $-H_\ell$ start with a black cherry or one of them is the empty segment and the other starts with a black cherry. If G_ℓ starts with a black cherry, it follows that $G_\ell - H_\ell \geq G_\ell > G$ since G is a basis element. If G_ℓ does not start with a black cherry, it means that G_ℓ is the empty segment and $-H_\ell$ starts with a black cherry. Since both players have been playing on G and $-H$ each turn, it follows that G is the game consisting of the first ℓ cherries of H. By Lemma 5.14, $|H_\ell| \geq |G| = G$ and therefore $G_\ell - H_\ell = 0 + |H_\ell| \geq G$. In both cases we see that $G_\ell - H_\ell \geq G$, so by induction on k we find that $G_\ell - H_\ell - (k-1) \cdot H + X \geq G - (k-1) \cdot H + X >_{SC} X$. Thus, $o(G - k \cdot H + X) \geq o(G_\ell - H_\ell - (k-1) \cdot H + X) \geq o(X)$.

Next, we look at the case where the lexicographically strongest component starting with a white cherry is in X with move number j. We need to show that $o(G - k \cdot H + X) \geq o(X)$. First, suppose $X \in \mathcal{L}$ with winning move i. By induction on the birthday of X, $o(G - k \cdot H + X_{ij}^S) \geq o(X_{ij}^S) = \mathcal{L}$. Since j is the best move for Right, we conclude that $G - k \cdot H + X \in \mathcal{L}$. Next, if $G - k \cdot H + X \in \mathcal{R}$, a winning move for Right is move j on the component X, since this is the lexicographically strongest component starting with a white cherry. By induction, it follows that $\mathcal{R} = o(G - k \cdot H + X_{ij}^S) \geq o(X_{ij}^S)$ for all i. Thus, $X \in \mathcal{R}$.

We conclude that $G - k \cdot H \geq_{SC} 0$. Lastly, again by induction on k and using the same strategy as before we find that $o(G - k \cdot H) \geq o(G_\ell - H_\ell - (k-1) \cdot H) \geq o(G - (k-1) \cdot H) = \mathcal{L}$ while $o(0) = \mathcal{U}$. Thus, strict equality does not hold, leaving us with $G - k \cdot H >_{SC} 0$. □

This theorem tells us that all basis elements are infinitely far away from each other. This means that working with sums of basis elements is very simple; using Lemma 5.9, we first cancel out every pair $(G, -G)$ of positive and negative basis elements with the same absolute value, so that we are left with a game in which every basis element does not occur both as a positive version and as a negative version. If we are now left with the game 0, this means the game is a draw. Otherwise, Theorem 5.15 tells us that the winner of the game is the player whose color the strongest remaining basis element starts with.

Example 5.16. Consider the sum of basis elements

$G = \triangleright[\bullet|\circ] + \triangleright[\circ|\bullet] + \triangleright[\circ|\bullet] + \triangleright[\bullet|\circ|\circ|\bullet] + \triangleright[\bullet|\circ|\circ|\circ]$

By Lemma 5.9, $G = \triangleright[\circ|\bullet] + \triangleright[\bullet|\circ|\circ|\bullet] + \triangleright[\bullet|\circ|\circ|\circ]$. The lexicographically strongest component in this game is $\triangleright[\circ|\bullet]$, so using Theorem 5.15 we conclude that $G \in \mathcal{R}$.

Since these basis elements are so nice to work, the next thing we will try to do is somehow decompose any Stack Cherries segment into a sum of basis elements.

Lemma 5.17. *Let G be a Stack Cherries segment of length n that is not a basis element. Let $0 < i < n$ such that $|G_i| \leq |G_k|$ for all $0 \leq k < n$ and let $0 \leq j < i$ such that $|G[1...i]_j| \leq |G[1...i]_k|$ for all $0 \leq k < i$. Then $|G_i| \leq |G[1...i]_j|$. Furthermore, if the $(i+1)$-st and $(j+1)$-st cherries of G have different colors, this inequality is strict.*

Proof. First suppose $|G_i|$ and $|G[1...i]_j|$ are not prefixes of each other. This means that the two mismatch before reaching the end of either segment, and since $|G_i| < |G_j|$ it follows that $|G_i| < |G[1...i]_j|$.

Next, suppose that $|G_i|$ is a strict prefix of $|G[1...i]_j|$. So $|G_i|$ and $|G_j|$ match for $n-i$ cherries. Since G_i has length $n-i$ it follows that $G[1...i]_j$ has length at least $n-i+1$. Since $|G_i| < |G_j|$ and $|G_i|$ and $|G_j|$ match for the first $n-i$ cherries, it follows that the $(n-i+1)$-st cherry of $|G_j|$ and therefore also the $(n-i+1)$-st cherry of $|G[1...i]_j|$ must be black. We conclude that $|G_i| < |G[1...i]_j|$.

Lastly, suppose that $|G[1...i]_j|$ is a strict prefix of $|G_i|$. So $|G_i|$ and $|G_j|$ match for at least $i-j$ cherries. Since $G[1...i]_j$ has length $i-j$ it follows that G_i has length at least $i-j+1$. If the $(i-j+1)$-st cherry of $|G_i|$ would be black, this would mean that the $(i-j+1)$-st cherry of $|G_j|$ also has to be black since $|G_i| < |G_j|$. So $|G_i|$ and $|G_j|$ match for at least $i-j+1$ cherries, their $(i-j+1)$-st cherries are the same color as their first cherries (black) and $|G_i| < |G_j|$. Thus, we also have $|(G_i)_{i-j}| < |(G_j)_{i-j}| = |G_i|$. This is a contradiction, since $|G_i| \leq |G_k|$ for all $0 \leq k < n$. We conclude that the $(i-j+1)$-st cherry of $|G_i|$ must be white and thus that $|G_i| < |G[1...i]_j|$.

We have now shown that if $|G_i| \neq |G[1...i]_j|$, then $|G_i| < |G[1...i]_j|$. In other words, we have $|G_i| \leq |G[1...i]_j|$. All that is left to show now is that $|G_i| = |G[1...i]_j|$ cannot hold when the $(i+1)$-st and $(j+1)$-st cherries of G have different colors. Suppose that $|G_i| = |G[1...i]_j|$. Since $|G_i| \leq |G_j|$ and $|G_i|$ and $|G_j|$ match for the first $n-i$ cherries at which point we have reached the end of $|G_i|$, the $(n-i+1)$-st cherry of $|G_j|$ must be black. In other words, the first and $(n-i+1)$-st cherries of G_j have the same color. Since $|G_i|$ is equal to $|G[1...i]_j|$, they must also have the same length, and thus $n-i = i-j$. We conclude that the $(j+1)$-st cherry of G has the same color as the $(n-i+1+j) = (i+1)$-st cherry. □

Theorem 5.18 (Stack Cherries Backwards Decomposition)**.** *Let G be a Stack Cherries segment of length n and let $0 \leq i < n$ such that $|G_i| \leq |G_j|$ for all $0 \leq j < n$. Then $G =_{SC} G[1...i] + G_i$.*

Proof. Assume without loss of generality that the first cherry of G is black. If $i = 0$, there is nothing to show, so assume that $0 < i < n$. First suppose that G_i also starts with a black cherry. If G and G_i do not match for the first i cherries, then $G[1...i] > G_i$ since $G > G_i$. If G and G_i match for the first $i+1$ cherries, it follows from $G > G_i$ that $G_i > (G_i)_i = G_{2i}$. But the first cherry of G_{2i} has the same color as the first cherry of G_i which is black, and therefore $|G_i| = G_i > G_{2i} = |G_{2i}|$. This is a contradiction, since G_i is the weakest suffix of G. So G and G_i cannot match for the first $i+1$ cherries. Thus, if G and G_i match for the first i cherries, they must mismatch at position $i+1$. The $(i+1)$-st cherry of G is the first cherry of G_i and therefore black, so the $(i+1)$-st cherry of G_i must be the end of the segment or white. In both cases, this means that $G[1...i] \geq G_i$. Thus, in any synchronized game $G[1...i] + G_i + X$ we may assume that Left will play on $G[1...i]$ before G_i. By induction on the birthday of G, we conclude that $G[1...i] + G_i = ((G[2...i] + G_i), (), ()) = ((G_1), (), ()) = G$.

Next, we will consider the case where G_i starts with a white cherry. By induction on the birthday of G, we can repeatedly apply this theorem and Lemma 5.17 on $G[1...i]$ to obtain a sequence of basis elements $B_1 \geq B_2 \geq \ldots \geq B_\ell$ satisfying $G[1...i] =_{SC} B_1 + B_2 + \ldots + B_\ell$. Since G starts with a black cherry, Lemma 5.17 tells us that B_1 starts with a black cherry and is strictly stronger than G_i and any B_j that starts with a white cherry. Now let X be any sum of Stack Cherries segments. Playing on $B_1 + B_2 + \ldots + B_\ell + G_i + X$, if the strongest segment starting with a white cherry W is G_i or a B_j, then B_1 is a basis element starting with a black cherry that is strictly stronger than the basis element W, which is again stronger than the other segments in the game starting with a white cherry. Thus, Theorem 5.15 tells us that $B_1 + B_2 + \ldots + B_\ell + G_i + X >_{SC} 0$. In other words, if the best move for Right is to play on G_i or any B_j, Left can win anyway, so we may ignore these moves. By induction we conclude that

$$\begin{aligned} G[1...i] + G_i &=_{SC} B_1 + B_2 + \ldots + B_\ell + G_i \\ &=_{SC} (((B_1)_1 + B_2 + \ldots + B_\ell + G_i), (), ()) \\ &=_{SC} ((G_1), (), ()) = G \end{aligned}$$

□

As we mentioned in the proof, repeatedly applying this theorem allows us to decompose any Stack Cherries segment into a unique sum of basis elements. With this, we can view any Stack Cherries segment as a concatenation of basis elements:

$$G = \rhd\boxed{B_1}\boxed{B_2}\boxed{\ldots}\boxed{B_\ell} =_{SC} \rhd\boxed{B_1} + \rhd\boxed{B_2} + \ldots + \rhd\boxed{B_\ell}$$

Furthermore, by Lemma 5.17 we have $|B_1| \geq |B_2| \geq \ldots \geq |B_\ell|$ and equality can only hold if B_i and B_{i+1} start with the same color.

Example 5.19. We will decompose the Stack Cherries segment

$$G = \triangleright \boxed{\bullet}\boxed{\bullet}\boxed{\circ}\boxed{\bullet}\boxed{\bullet}\boxed{\circ}\boxed{\circ}$$

into basis elements. We see that the suffix $G_4 = \triangleright \boxed{\bullet}\boxed{\circ}\boxed{\circ}$ is the lexicographically weakest out of all non-empty suffixes, so Theorem 5.18 tells us that $G =_{SC} \triangleright \boxed{\bullet}\boxed{\bullet}\boxed{\circ}\boxed{\bullet} + \triangleright \boxed{\bullet}\boxed{\circ}\boxed{\circ}$, noting that G_4 is a basis element. In the next iteration, we decompose $\triangleright \boxed{\bullet}\boxed{\bullet}\boxed{\circ}\boxed{\bullet}$ into $\triangleright \boxed{\bullet}\boxed{\bullet} + \triangleright \boxed{\circ}\boxed{\bullet}$ and in the final iteration we decompose $\triangleright \boxed{\bullet}\boxed{\bullet}$ into $\triangleright \boxed{\bullet} + \triangleright \boxed{\bullet}$. We conclude that

$$G =_{SC} \triangleright \boxed{\bullet} + \triangleright \boxed{\bullet} + \triangleright \boxed{\circ}\boxed{\bullet} + \triangleright \boxed{\bullet}\boxed{\circ}\boxed{\circ}$$

where each component is a basis element.

One thing to note is that, if we have a Stack Cherries segment G where the weakest suffix is G_i for some $i > 0$, then the weakest suffix of G_1 is still G_i. This means that if we have a Stack Cherries segment $G = \triangleright \boxed{B_1}\boxed{B_2}\boxed{\ldots}\boxed{B_\ell}$ with basis element decomposition $\triangleright \boxed{B_1} + \triangleright \boxed{B_2} + \ldots + \triangleright \boxed{B_\ell}$, that after $\ell - 1$ iterations of Theorem 5.18 we find that $G_1 = \triangleright \boxed{(B_1)_1}\boxed{B_2}\boxed{\ldots}\boxed{B_\ell} =_{SC} \triangleright \boxed{(B_1)_1} + \triangleright \boxed{B_2} + \ldots + \triangleright \boxed{B_\ell}$. This shows us that the impact of a move on a Stack Cherries segment only depends on the leftmost basis element in its decomposition. In other words, if we have another Stack Cherries segment $H = \triangleright \boxed{A_1}\boxed{A_2}\boxed{\ldots}\boxed{A_k}$ with basis element decomposition $\triangleright \boxed{A_1} + \triangleright \boxed{A_2} + \ldots + \triangleright \boxed{A_k}$ such that $A_1 = B_1$, then $G_1 + H = G + H_1$. We therefore give this leftmost basis element a special name.

Definition 5.20. Let G be a Stack Cherries segment. We define the *main basis element* of G as the lexicographically strongest basis element in its basis element decomposition.

If the decomposition of a Stack Cherries segment consists of many basis elements, determining the main basis element of that segment using Theorem 5.18 can be tedious, since this method works from back to front while the main basis element is the leftmost basis element in its decomposition. To more easily determine the main basis element of a segment we will introduce an equivalent Stack Cherries decomposition method that works from front to back. Before this, we first introduce a lemma that will help us with the proof of its correctness.

Lemma 5.21. *Let G be a Stack Cherries segment of length n and let $0 \leq i < n$ such that $|G_i| \leq |G_j|$ for all $0 \leq j < n$. Then, for $0 \leq j, k < i$, we have*

$$G_j > G_k \iff G[1...i]_j > G[1...i]_k$$

Proof. Define $H = G[1...i]$. Assume without loss of generality that the $(i+1)$-st cherry of G is black. Assume that $G_j > G_k$. First suppose that $j < k$. If G_j and G_k do not match for the first $i - k$ cherries, then $H_j > H_k$ since the mismatch happens before the end of either segment. If G_j and G_k do match for the first $i - k$ cherries, then the $(i - k + 1)$-st cherry of G_j must be black since $G_j > G_k$ and the $(i - k + 1)$-st cherry of G_k is $G_k[i - k + 1] = G[i + 1] = \texttt{black}$. Thus, H_j and H_k match for the first $i - k$ cherries, after which H_j is followed by a black cherry while we have reached the end of H_k. It follows that $H_j > H_k$.

Now suppose that $j > k$. If G_j and G_k do not match for the first $i - j$ cherries we can again conclude that $H_j > H_k$ since the mismatch happens before the end of either segment. Now suppose G_j and G_k do match for the first $i - j$ cherries. Since $G_j > G_k$, it follows that $G_i = G_{j+(i-j)} > G_{k+(i-j)}$. If the first cherry of G_{k+i-j}, or equivalently the $(i-j+1)$-st cherry of G_k, would be black, this would mean that $|G_i| > |G_{k+i-j}|$. This is a contradiction, since we assume that G_i is the lexicographically weakest suffix of G. Thus, the $(i - j + 1)$-st cherry of G_k must be white. So H_j and H_k match for the first $i - j$ cherries, after which we have reached the end of H_j and H_k is followed by a white cherry, so we conclude that $H_j > H_k$.

We have now shown that $G_j > G_k \implies H_j > H_k$. By flipping j and k, the same argument shows that $G_j < G_k \implies H_j < H_k$. We also note that $G_j = G_k \iff j = k \iff H_j = H_k$. We conclude that $G_j > G_k \iff G[1...i]_j > G[1...i]_k$. □

In other words, removing the lexicographically weakest suffix from a Stack Cherries segments preserves the order of the longer suffixes.

Theorem 5.22 (Stack Cherries Forwards Decomposition)**.** *Let G be a non-basis Stack Cherries segments of length n and let $0 < i < n$ be the smallest index such that $|G_i| \leq |G|$. Then $G = G[1...i] + G_i$*

Proof. Let $0 < k < n$ such that $|G_k| < |G_j|$ for all $0 < j < n$. If $G[1...k]$ is a basis element, Lemma 5.21 tells us that $|G_j| > |G|$ for all $0 \leq j < k$ and thus that $i \geq k$. Since $|G_k| < |G|$, we conclude that $i = k$. Thus, by Theorem 5.18, $G = G[1...i] + G_i$.

Next, suppose $G[1...k]$ is not a basis element. By Lemma 5.21, $i < k$ is also the smallest index such that $|G[1...k]_i \leq |G[1...k]|$. By induction on the birthday of G, it follows that $G[1...k] = G[1...i] + G[1...k]_i = G[1...i] + G[1 + i...k]$. Since G_k is the weakest suffix of G, it is also the weakest suffix of G_i. Theorem 5.18 now tells us that $G_i = G[1 + i...k] + G_k$. Thus,

$$G = G[1...k] + G_k = G[1...i] + G[1 + i...k] + G_k = G[1...i] + G_i.$$

□

5.2 Cherries Decomposition

Now that we have a solid method for determining the winner of any Stack Cherries game, we would like to find a method for reducing any game of Cherries to a game of Stack Cherries. In this subsection we will propose such a method and lay the foundation for a possible proof of its correctness.

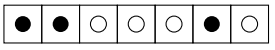
For a Cherries segment G, we use the notation $G_{(i,j)}$ to refer to the Cherries segment where the front i cherries and the back j cherries have been removed, and we again use the notation $G[a...b]$ to refer to the Cherries segment starting at the a-th cherry of G and ending with the b-th cherry. A new piece of notation is $\overline{G}$, which we use to refer to the reverse of G. Note that this notation will mostly be used in the context of lexicographical comparisons, since reversing a Cherries segment does not change anything about its value.

We will see that some of the theorems and proof techniques we used to analyze Stack Cherries can also be applied to regular Cherries in some modified way. One of these theorems is the following, which very much resembles Theorem 5.7:

Theorem 5.23. *Let G be a Cherries segment of length greater than k such that $G \succ \overline{G}$. Suppose that G and $\overline{G}$ match for the first k cherries. Then $G_{(i,0)} \geq G_{(0,i)}$ for all $1 \leq i \leq k$.*

Proof. The proof of this theorem is also very similar to that of Theorem 5.7. We will again first look at the case $i = k$. Since G and $\overline{G}$ match for the first k cherries and $G \succ \overline{G}$, it follows that the $(k+1)$-st cherry from the start of G is black and the $(k+1)$-st cherry from the end of G is white. By quickly checking all the cases, we conclude that $G_{(0,k)} + X \in \mathcal{L} \implies G_{(k,0)} + X \in \mathcal{L}$ and $G_{(k,0)} + X \in \mathcal{R} \implies G_{(0,k)} + X \in \mathcal{R}$ for all synchronized games X. Thus, $G_{(k,0)} \geq G_{(0,k)}$

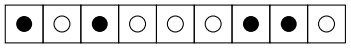
Next, we consider the case $1 \leq i < k$. Again, using the same induction argument from Theorem 5.7, we conclude that $G_{(0,i)} + X \in \mathcal{L} \implies G_{(i,0)} + X \in \mathcal{L}$ and $G_{(i,0)} + X \in \mathcal{R} \implies G_{(0,i)} + X \in \mathcal{R}$ for all synchronized games X. Thus, $G_{(i,0)} \geq G_{(0,i)}$. □

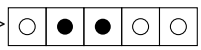
So, just like the players will always pick the segment with the highest absolute value in a game of Stack Cherries, on a segment of regular Cherries where both ends have the same color, the player of that color will always pick the side from which the segment reads the lexicographically strongest. Obtaining a similar result for segments with differently colored endings is not as straightforward. In fact, results as general as this that we would expect to be true are not actually always true. Take for example the Cherries segment $G =$  .
Since the segment read from front to back is lexicographically stronger than the segment read from the back to front, we would expect that removing the first and last cherry will benefit Left. But with $X = ((1), (1), (-1))$, we see that $o(G + X) = \mathcal{U}$, while $o(G_{(1,1)} + X) = \mathcal{R}$, which means that $G_{(1,1)} \geq G$ does not hold.

However, we still expect this result to be true if we restrict X to be a sum of Cherries segments. With this in mind, we propose a method for decomposing a Cherries segment into Stack Cherries segments, based on repeatedly taking cherries from the lexicographically strongest side.

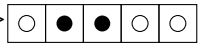
Definition 5.24. Let G be a Cherries segment. Rotate G such that $|G| \succeq |\overline{G}|$. Let $0 \leq i \leq n$ be the smallest index such that $|G_{(i,0)}| \prec |\overline{G_{(i,0)}}|$ or such that $|G_{(i,0)}| = |\overline{G_{(i,0)}}|$ and $G_{(i,0)}$ and $\overline{G_{(i,0)}}$ do not start with the same colors. Then we define $\mathrm{dec}_1(G) = \rhd G[1...i]$. We define $\mathrm{dec}_2(G)$ by repeating this process on $G_{(i,0)}$, and so on. Finally, we define $\mathrm{dec}(G) = \sum_i \mathrm{dec}_i(G)$.

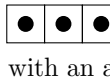
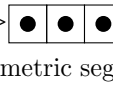
This definition looks quite complicated but can be described in words as follows: Given a Cherries segment G, pick the side from which the segment is lexicographically strongest and start taking cherries from this side until this side is no longer the lexicographically strongest. The cherries that have been removed so far make up $\mathrm{dec}_1(G)$. We then repeat this process to determine $\mathrm{dec}_2(G)$, $\mathrm{dec}_3(G)$ and so on. We are finished when there are no cherries left or when the remaining segment H is *anti-symmetric*, which means that $|H| = |\overline{H}|$ and H and $\overline{H}$ start with different colors. At this point, all future iterations will yield $\mathrm{dec}_i(G) = 0$.

Example 5.25. Consider the Cherries segment $G =$  . We see that G is lexicographically stronger front to back than back to front so we take the front cherry from G. Next, we see that $G_{(1,0)}$ is still stronger front to back than back to front so we again take the front cherry. At this point we are left with $G_{(2,0)}$, which is stronger from back to front, so we stop this iteration and conclude that $\mathrm{dec}_1(G) = \rhd G[1...2] = \rhd$  .

In the next iteration we see that $|G_{(2,0)}| \prec |\overline{G_{(2,0)}}$ and $|G_{(2,1)}| \prec |\overline{G_{(2,1)}}$ so we take two cherries from the back. At this point we have $|G_{(2,2)}| = |\overline{G_{(2,2)}}|$, but $G_{(2,2)}$ starts and ends with the same color so we keep going from this side and take the next cherry. Next, we see that $|G_{(2,3)}| \prec |\overline{G_{(2,3)}}$ and $|G_{(2,4)}| \prec |\overline{G_{(2,4)}}$ so we again take two cherries from the back, Finally, we are left with the anti-symmetric segment $G_{(2,5)} =$  so we stop this iteration and conclude that $\mathrm{dec}_2(G) = \rhd$ 

Since we are now left with an anti-symmetric segment, we see that all future iterations stop immediately, and thus $\mathrm{dec}_3(G) = \mathrm{dec}_4(G) = \ldots = 0$. Thus,

$$\mathrm{dec}(G) = \sum_i \mathrm{dec}_i(G) = \mathrm{dec}_1(G) + \mathrm{dec}_2(G) = \rhd$$  $+ \rhd$ 

If we have a Cherries segment that only consists of one color, all cherries will be taken in the first iteration. For example, $\mathrm{dec}\left(\right.$  $\left.\right) = \rhd$  . In all other cases, we can show that we will end up with an anti-symmetric segment

instead of the empty segment after some amount of iterations. Indeed, if G is a Cherries segment starting with one black cherry followed by a positive amount of white cherries, then $|G| \preceq |\overline{G}|$ and thus the front cherry of G will not be taken when determining $\text{dec}(G)$. So if G is a Cherries segment containing black and white cherries, we will never be able to take all cherries when determining $\text{dec}(G)$, so we must end up with an anti-symmetric segment after some amount of iterations.

We will now prove some basic properties of this decomposition function, to help us get closer to a proof of its correctness.

Lemma 5.26. *Let G be a Cherries segment with $\rhd\!\lhd G \neq \rhd\!\lhd\overline{G}$. Then $|\text{dec}_1(G)| \geq |\text{dec}_2(G)|$.*

Proof. We may assume without loss of generality that $|G| \succeq |\overline{G}|$, since we could reverse G if this was not the case. If $\text{dec}_2(G) = 0$ the result follows immediately. So assume $\text{dec}_2(G) \neq 0$, which also means that $\text{dec}_1(G) \neq 0$. In this case, G can be written in the form $\boxed{A}\boxed{X}\boxed{B}$ such that A, X, B are sequences of cherries with $\rhd\!A = \text{dec}_1(G)$ and $\rhd\!\overline{B} = \text{dec}_2(G)$.

If $|\rhd\!A|$ and $|\rhd\!\overline{B}|$ are not prefixes of each other, this means that they mismatch before reaching the end of either segment. Since $|G| \geq |\overline{G}|$ it follows that $|\rhd\!A| > |\rhd\!\overline{B}|$ and thus that $|\text{dec}_1(G)| \geq |\text{dec}_2(G)|$.

Next, consider the case where $|\rhd\!A|$ is a prefix of $|\rhd\!\overline{B}|$. Let $|\rhd\!A|$ have length n. Assume that $|\rhd\!A| < |\rhd\!\overline{B}|$. This means that the $(n+1)$-st cherry of $|\rhd\!\overline{B}|$ is black. Since $|G| \succeq |\overline{G}|$ it also follows that the $(n+1)$-st cherry of $|\rhd\!\lhd G|$ is black. So $|G|$ and $|\overline{G}|$ match for at least $n+1$ cherries and their $(n+1)$-st cherry is black, which means that if we remove the first n cherries and the last n cherries from G, the segment should still be lexicographically stronger from front to back than from back to front, or in other words $|G_{(n,n)}| \succeq |\overline{G_{(n,n)}}|$. But $G_{(n,n)}$ is of the form $\boxed{X}\boxed{B_{(0,n)}}$ and so by the definition of dec we have $|G_{(n,n)}| \preceq |\overline{G_{(n,n)}}|$. We conclude that $|G_{(n,n)}|$ is lexicographically equal to $|\overline{G_{(n,n)}}|$ and thus that $|G|$ is lexicographically equal to $|\overline{G}|$. But by assumption we know that $\rhd\!\lhd G \neq \rhd\!\lhd\overline{G}$ and we also have $\rhd\!\lhd G \neq -\rhd\!\lhd\overline{G}$ since equality would mean that G is anti-symmetric and $\text{dec}_1(G)$ would be 0. This is a contradiction, so $|\rhd\!A| \geq |\rhd\!\overline{B}|$ must hold. We conclude that $|\text{dec}_1(G)| \geq |\text{dec}_2(G)|$. Lastly, the proof for when $|\rhd\!\overline{B}|$ is a prefix of $|\rhd\!A|$ is analogous. □

Note that by repeatedly applying Lemma 5.26 while removing cherries from the lexicographically strongest side, we can conclude that $(|\text{dec}_i(G)|)_{i\geq 1}$ is a non-increasing sequence of Stack Cherries segments. If $\text{dec}_1(G)$ has length n and we remove $1 \leq k < n$ cherries from lexicographically strongest side of G resulting in the game H, then $\text{dec}_1(H) = \text{dec}_1(G)_k$ and $\text{dec}_i(H) = \text{dec}_i(G)$ for $i \geq 2$ by definition of our decomposition. Combining this with Lemma 5.26, it follows that all the suffixes of $\text{dec}_i(G)$ must be lexicographically stronger than

$\text{dec}_{i+1}(G)$, for all i and all Cherries segments G. With this, we can prove an even stronger result about the non-increasing nature of $(|\text{dec}_i(G)|)_{i\geq 1}$

Lemma 5.27. *Let G be a Cherries segment such that $\text{dec}_1(G) > 0$ and $\text{dec}_3(G) < 0$. Then the main basis element of $|\text{dec}_1(G)|$ is strictly greater than that of $|\text{dec}_3(G)|$.*

Proof. Assume without loss of generality that $|G| \succ |\overline{G}|$. We will prove this statement using contradiction. By Lemma 5.26, the main basis element of $|\text{dec}_1(G)|$ is greater than or equal to the main basis element of $|\text{dec}_3(G)|$. So assume that the two main basis elements are both equal to some basis element B. Since all the suffixes of $\text{dec}_1(G)$ must be lexicographically stronger than $\text{dec}_3(G)$, it follows that the basis element decomposition of $|\text{dec}_1(G)|$ may only contain the basis element B. In other words, $|\text{dec}_1(G)|$ is the concatenation of some positive amount of B's. Since $|\text{dec}_1(G)| \geq |\text{dec}_2(G)| \geq |\text{dec}_3(G)|$, it also follows that the main basis element of $|\text{dec}_2(G)|$ is B. The same argument now tells us that $|\text{dec}_2(G)|$ is also the concatenation of some positive amount of B's. This amount is less than or equal to the amount of B's in $|\text{dec}_1(G)|$, since $|\text{dec}_1(G)| \geq |\text{dec}_2(G)|$ must hold.

So, we now know that G is of the form $\boxed{nB}\boxed{-B}\boxed{X}\boxed{\overline{\pm mB}}$, where nB represents the concatenation of n B's, $\pm mB$ represents either mB or $-mB$, X is some sequence of cherries and $m \leq n$. Here, $\text{dec}_1(G) = nB$ and $\text{dec}_2(G) = \pm mB$. If we remove the first $n - m$ copies of B from the front, we are left with $\boxed{mB}\boxed{-B}\boxed{X}\boxed{\overline{\pm mB}}$. Since this segment is still lexicographically stronger from front to back than from back to front and $-B$ starts with a white cherry, we conclude that $\boxed{-B}\boxed{X}$ is lexicographically stronger from back to front than from front to back. This is a contradiction by the definition of our decomposition, since we have removed $\text{dec}_1(G)$ and $\text{dec}_2(G)$ and thus $\boxed{-B}\boxed{X}$ should be lexicographically stronger from front to back than from back to front. □

It follows from Lemma 5.26 and Lemma 5.27 that, from the component $\text{dec}(G)$ in a sum of Stack Cherries games, we may assume that the players will only take cherries from $\text{dec}_1(G)$ or $\text{dec}_2(G)$. Indeed, if $\text{dec}_n(G)$ is the strongest segment starting with one player's color for some $n > 2$, then by Lemma 5.26 and Lemma 5.27 $\text{dec}_1(G)$ will have a strictly stronger main basis element. So, using basis element decomposition we find that this game is a win for the other player. Thus, if playing on $\text{dec}_n(G)$ for some $n > 2$ seems like the best option, this game is a win for the other player anyways, so if we ignore such options the outcome class of the game will not change.

Definition 5.28. Let G, H be synchronized games. We say $G =_C H$ if $o(G + X) = o(H + X)$ for any sum of Cherries and Stack Cherries segments X.

Conjecture 5.29. *Let G be a Cherries segment. Then $G =_C \text{dec}(G)$.*

There are two things we would still need to show for a possible proof of this conjecture: First, if we remove a cherry from a Cherries segment G on the lexicographically weakest side, this should change the value of its decomposition by at least as much as taking a cherry from the lexicographically strongest side. Second, if we have a Cherries segment G where $\mathrm{dec}_1(G)$ and $\mathrm{dec}_2(G)$ have the same main basis element, then removing a Cherry from the side of $\mathrm{dec}_2(G)$ should change the value of its decomposition by the same amount as removing a cherry from the side of $\mathrm{dec}_1(G)$ does.

If we were to prove these two statements, a proof for Conjecture 5.29 would follow from Lemma 5.26 and Lemma 5.27 and the theory of Stack Cherries.

In an attempt to further convince the reader of the correctness of this conjecture, we can programmatically show that it is correct up to a specific point. We managed to check that the decomposition works for all Cherries games containing at most 20 cherries. More precisely, for all sums of Synchronized Cherries segments containing in total at most 20 cherries, the outcome class of this game is equal to the outcome class of the sum of the Stack Cherries decompositions of all the segments. Here, we determined the outcome classes of the Cherries games directly using the definition of the outcome classes and a dynamic programming approach, and we determined the outcome classes of the Stack Cherries decompositions by simulating the optimal Stack Cherries strategy from Theorem 5.7.

This decomposition also gives us a good explanation of the infinitesimal behaviour we observed all the way back at the start of this section: the basis element decomposition of a Cherries segment G with combinatorial value n contains the basis element n times plus possibly some other weaker basis elements. By Theorem 5.15, these weaker basis elements are infinitely weaker. Thus, G is "infinitesimally close" to n.

6 Algorithms for Synchronized Cherries

In the previous section, we have shown how to assign a value to any Cherries or Stack Cherries position in the form of Stack Cherries basis elements. Together with Theorem 5.15, which tells us that basis elements are infinitely larger than any smaller basis element, this gives us a very clear measure of how valuable a Cherries or Stack Cherries position is for either player. More specifically, this can help us answer the important question of which player, if any, has a winning strategy on a sum of Cherries and Stack Cherries games.

It is not hard to find a theoretically correct method for determining the winner of any synchronized game; by simply trying out all possible options you will find out if some move always results in a win for one player. The difficulty lies in finding an algorithm that does this efficiently. In this section, we will propose an algorithm for determining the winner of any sum of Cherries and Stack Cherries games using our knowledge from the previous section, and we will analyze its complexity.

The algorithm consists of three phases. In the first phase, we decompose the Cherries segments into Stack Cherries segments based on Definition 5.24. Next, we decompose the Stack Cherries positions into basis elements using Theorem 5.18. Finally, we determine the winner of the resulting sum of basis elements using Theorem 5.15.

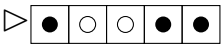
6.1 Stack Cherries Decomposition

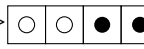
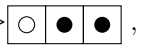
We will start by discussing the second phase of the algorithm: decomposing a Stack Cherries segment into basis elements. Using Theorem 5.18, we see that one way to achieve this is by repeatedly finding the lexicographically weakest suffix and splitting the segment at that point. This means that lexicographically comparing suffixes is an important part of this algorithm. A data structure that can help us with this is the *suffix array*. A suffix array is an array containing all suffixes of a string in lexicographic order. More specifically, a suffix array contains the starting indices of the suffixes in lexicographic order. In this thesis, we are using one-based indexing for arrays, just like we have been treating (Stack) Cherries segments as one-based.

Example 6.1. Consider the word CHERRIES with alphabetical order. The suffixes of this word are CHERRIES, HERRIES, ERRIES, RRIES, RIES, IES, ES and S. We say the suffix CHERRIES has starting index 1, HERRIES has starting index 2 and so on. If we sort these from lexicographically smallest to greatest like in a dictionary, we get CHERRIES, ERRIES, ES, IES, HERRIES, RIES, RRIES and S. So the suffix array of CHERRIES is $[1, 3, 7, 6, 2, 5, 4, 8]$.

We can also create a suffix array for a Stack Cherries segment using the lexicographic order we introduced in the previous section. Consider the Stack Cherries

segment



If we order its suffixes from smallest to greatest we get  ,  ,  , and . So the suffix array of this Stack Cherries segment is $[2, 3, 1, 5, 4]$

Suffix arrays were developed as a space efficient alternative to suffix trees [MM93], storing information about the suffixes of a string in an array instead of in a tree. Suffix trees provide an efficient solution to many string related problems, such as searching for a pattern within a string [Gus97]. Suffix trees have also found success in the field of computational biology, where the strings represent long sequences of DNA, for example [KS99]. All of these suffix tree applications can also be implemented using suffix arrays, using the same time complexity and generally using less space [AKO04].

If we were to naively create a suffix array by filling an array with the suffixes of a string of length n and applying a general sorting algorithm, this would have time complexity $O(n^2 \log n)$: general sorting algorithms require $O(n \log n)$ comparisons and lexicographically comparing two suffixes has worst-case complexity $O(n)$. Using intrinsic properties of suffixes, algorithms have been devised that reduce this complexity all the way down to $O(n)$ [LLH18].

From Lemma 5.21, we know that removing the lexicographically weakest suffix from a Stack Cherries segment preserves the order of the suffixes with a lower starting index. More specifically, suppose we have a suffix array of a Stack Cherries segment G and the lexicographically weakest suffix of G has starting index i. If we remove the entries from the array representing suffixes with starting index greater than or equal to i, the resulting array will still be a suffix array for the segment $G[1...i]$. We use this in Algorithm 1 to efficiently decompose Stack Cherries segments into basis elements.

Theorem 6.2. *Algorithm 1 is correct and has time complexity $O(n)$.*

Proof. Since S is a suffix array of G, in lines 2–5 we set r and ℓ such that $G[S[r]...n]$ is the lexicographically smallest suffix starting with a black cherry, if any, and $G[S[\ell]...n]$ is the lexicographically greatest suffix starting with a white cherry, if any. This means that either $G[S[r]...n]$ or $G[S[\ell]...n]$ is the lexicographically weakest suffix of G. In lines 9–19, we determine which of these is the lexicographically weakest, add this suffix to D and decrease m by the length of this suffix to indicate the new end of the segment. By Theorem 5.18, G is now equal to $G[1...m] + G_m$.

By Lemma 5.21, S is also a suffix array of $G[1...m]$ if we ignore the values greater than or equal to m. Thus, after lines 20–23, $G[S[\ell]...m]$ and $G[S[r]...m]$ will again be the lexicographically greatest suffix starting with a white cherry and the lexicographically smallest suffix starting with a black cherry of $G[1...m]$, respectively.

Algorithm 1: Decompose a Stack Cherries segment into basis elements

Input: A Stack Cherries segment G of length n

Output: An array of basis elements D such that $G =_{SC} \sum_i D[i]$

```
1  Let S be a suffix array of G
2  r ← 1
3  while r ≤ n and G[S[r]] = White do
4      r ← r + 1        // r is the index of the weakest positive suffix
5  ℓ ← r − 1            // ℓ is the index of the weakest negative suffix
6  m ← n
7  i ← 1
                                                    // Iteration counter
8  while m > 0 do
       /* Determine s, the weakest suffix index in G[0...m − 1]      */
9      if ℓ < 1 then
10         s ← S[r]
11     else if r > n then
12         s ← S[l]
13     else
14         if |G[S[r]...m]| < |G[S[l]...m]| then
15             s ← S[r]
16         else
17             s ← S[l]
18     D[i] ← G[s...m]
19     m ← s − 1
       /* Update ℓ, r to the new weakest suffixes in G[1...m]        */
20     while ℓ ≥ 1 and S[ℓ] ≥ m do
21         ℓ ← ℓ − 1
22     while r ≤ n and S[r] ≥ m do
23         r ← r + 1
24     i ← i + 1
```

So, in the next iteration of the while loop, we will find the lexicographically weakest suffix of $G[1...m]$, remove it and add it to the list. We continue until we have gone through the entire segment, or equivalently until $m = 0$. At this point, we conclude that D is a decomposition of G into basis elements.

Line 1 of this algorithm has complexity $O(n)$ since we can create a suffix array in linear time. Lines 2–5 also have complexity $O(n)$ since the loop body will be executed at most n times. Lastly, the while loop on lines 8–24. In line 14, we compare two suffixes lexicographically, which costs time linear in the length of the shorter suffix. We then decrease m by the length of one of these suffixes, so the complexity of line 14 is at most linear in the amount m gets decreased. Since we decrease m from n to 0, this means that over the entire execution, line 14 will have complexity $O(n)$. The while-conditions of lines 20 and 22 can be true at most n times since we do not decrease ℓ beyond 0 and do not increase r beyond $n + 1$, so lines 20–23 also have time complexity $O(n)$ over the entire execution. Lastly, all other operations on lines 8–24 take constant time and will be executed at most n times since we decrease m by at least 1 each iteration. We conclude that this algorithm has time complexity $O(n)$.

□

6.2 Outcome of a Sum of Basis Elements

After performing the algorithm from the previous section on all of our Stack Cherries segments, we are left with a sum of basis elements. The only thing that is left to do now is apply Lemma 5.9 and Theorem 5.15 to determine the outcome of this game. From these two theorems it follows that the winner of any sum of basis elements is the player that has the most copies of the basis element with the largest absolute value. In case of a tie, we move on to the basis element with the next largest absolute value.

We would like our algorithm for this step to have time complexity $O(n)$, since the time complexity of the entire algorithm is equal to the time complexity of its slowest step and the previous step had complexity $O(n)$ (here, n is the sum of lengths of all the basis elements). A useful data type that will help us achieve this is the *dictionary*. Dictionaries allow the user to store data as key-value pairs. A common implementation of the dictionary makes use of a hash table, since they allow for lookup and insertion with average time complexity $O(1)$ [CLRS09]. However, since we will be using strings as keys, calculating the hash of a key or comparing two keys in the case of a collision will have time complexity $O(s)$, where s is the length of the string. Thus, the total time complexity for lookup or insertion will be $O(s)$.

In Algorithm 2, we use a dictionary as a counter for how many positive copies compared to negative copies there are for each basis element.

Algorithm 2: Determine the outcome of a sum of basis elements

Input: An array of basis elements D

Output: The outcome c of the sum of games in D

```
1  T ← {}                          // Initialize T as an empty dictionary
2  for G ∈ D do
3      if |G| ∉ T then
4          T[|G|] ← 0              // Add key |G| if it does not exist yet
5      if G > 0 then
6          T[|G|] ← T[|G|] + 1
7      else
8          T[|G|] ← T[|G|] − 1
9  m ← 0
10 c ← 𝒰
11 for G ∈ D do
12     if |G| > m and T[|G|] ≠ 0 then
13         m ← |G|
14         if T[|G|] > 0 then
15             c ← ℒ
16         else
17             c ← ℛ
```

Theorem 6.3. *Algorithm 2 is correct and has time complexity $O(n)$.*

Proof. In the first for loop, we count the amount of positive and negative copies for each basis element. If we encounter a positive element $G > 0$, we increase the counter $T[|G|]$ by one and if we encounter a negative element $G < 0$ we decrease the counter $T[|G|]$ by one. It follows that after this for loop $T[|G|]$ contains the amount of occurrences of $|G|$ minus the amount of occurrences of $-|G|$ for each basis element G in D.

Next, we will use the variable m to store the basis element with the highest absolute value and a non-zero counter. If the counter for this element is positive, Lemma 5.9 and Theorem 5.15 tells us that $o\left(\sum_i D[i]\right) = \mathcal{L}$, and if the counter is negative that $o\left(\sum_i D[i]\right) = \mathcal{R}$. Lastly, if there are no basis elements with a non-zero counter, $o\left(\sum_i D[i]\right) = \mathcal{U}$ by Lemma 5.9. Thus, at the end of the algorithm, the variable c contains the outcome class of $\sum_i D[i]$.

In the first for loop, for a basis element $G \in D$ with length s, the loop body will have time complexity $O(s)$ if we implement the dictionary T using a hash table. So the entire loop will have time complexity $O(n)$, where n is the sum of lengths of all elements in D. Similarly, the second for loop body also has complexity $O(s)$ for a basis element $G \in D$ with length s, since lexicographically comparing $|G|$ and m has worst-case complexity $O(s)$. So the second for loop also has total complexity $O(n)$. We conclude that the entire algorithm has time complexity

$O(n)$. □

Combining Algorithm 1 and Algorithm 2, we see that we can determine the winner of any sum of Stack Cherries segments in only $O(n)$ time, where n is the total amount of cherries in all the segments.

6.3 Cherries Decomposition

Lastly, we propose Algorithm 3 for decomposing any Cherries segment into a sum of Stack Cherries segments based on Definition 5.24. Note that the correctness of this algorithm relies on the correctness of Conjecture 5.29.

Theorem 6.4. *Algorithm 3 is correct and has time complexity $O(n^2)$.*

Proof. Algorithm 3 precisely describes the construction of $\mathrm{dec}(G)$ from Definition 5.24: While G is not empty and not anti-symmetric, repeatedly pick cherries from the side that is lexicographically strongest and add them to Stack Cherries segments. So, at the end of the algorithm, $D[i] = \mathrm{dec}_i(G)$ for all indices i of D, and $\mathrm{dec}_j(G) = 0$ for all indices j greater than the size of D. In other words, $\sum_i D[i] = \mathrm{dec}(G)$. By Conjecture 5.29, we conclude that $\sum_i D[i] =_C G$.

Lines 1–9 have time complexity $O(n)$, since we are doing a constant amount of constant-time operations and one linear-time operation (lexicographically comparing $|G|$ and $|\overline{G}|$). The loop body of lines 11-24 will be executed at most n times, since we either increase ℓ by 1 or decrease r by 1 each iteration, and we will stop at the latest when $\ell = r$. The while-condition check and loop body both have time complexity $O(n)$, since they consist of a constant amount of constant-time operations and linear-time operations. Thus, the entire while loop has time complexity $O(n^2)$. Lastly, the part consisting of lines 25-28 has time complexity $O(n)$, since it involves of copying and possibly reversing an array of length at most n. We conclude that Algorithm 3 has time complexity $O(n^2)$. □

If Conjecture 5.29 is correct, combining Algorithm 1, Algorithm 2 and Algorithm 3 gives us an $O(n^2)$ method for determining the winner of any Synchronized Cherries game.

Algorithm 3: Decompose a Cherries segment into Stack Cherries segments

Input: A Cherries segment G of length n

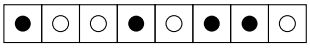
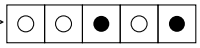
Output: An array of Stack Cherries segments D such that $G =_C \sum_i D[i]$

$\ell \leftarrow 1$
$r \leftarrow n$
if $|G| \succeq |\overline{G}|$ **then**
    $s \leftarrow$ `left` `// The side from which we are taking cherries`
    $t \leftarrow 1$ `// The position we started from this iteration`
else
    $s \leftarrow$ `right`
    $t \leftarrow n$
$i \leftarrow 1$ `// Iteration counter`
while $\ell < r$ **and** $G[\ell...r] \neq -\overline{G[\ell...r]}$ **do**
    if $s =$ `left` **and** $|G[\ell...r]| \prec |\overline{G[\ell...r]}|$ **then**
        $D[i] \leftarrow \triangleright G[t...l-1]$
        $s \leftarrow$ `right`
        $t \leftarrow r$
        $i \leftarrow i+1$
    else if $s =$ `right` **and** $|G[\ell...r]| \succ |\overline{G[\ell...r]}|$ **then**
        $D[i] \leftarrow \triangleright \overline{G[r+1...t]}$
        $s \leftarrow$ `left`
        $t \leftarrow l$
        $i \leftarrow i+1$
    if $s =$ `left` **then**
        $\ell \leftarrow \ell + 1$
    else
        $r \leftarrow r - 1$
`/* Add the cherries from the final iteration to the result */`
if $s =$ `left` **then**
    $D[i] \leftarrow \triangleright G[t...l-1]$
else
    $D[i] \leftarrow \triangleright \overline{G[r+1...t]}$

7 Cherries Variants

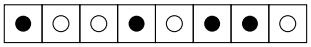
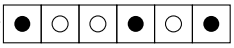
In this section, we will briefly go over the main results for some variants of Cherries. We will only write out a complete proof for the more difficult theorems.

7.1 Plus-Minus Stack Cherries

In Section 5, we conjectured that every Cherries segment can be decomposed into Stack Cherries segments, but not that all moves on a Cherries segment are equally as good as some move on a Stack Cherries segment. Take for example the Cherries segment $G =$  . By Conjecture 5.29, $G = 0$ and $G_{(1,0)} = \triangleright$  . In other words, taking the first cherry from G is equivalent to increasing the value of the game by $\triangleright$  . But we cannot seem to find any Stack Cherries segment that has this same property. In an attempt to better understand how such a move compares to other moves on (Stack) Cherries segments, we introduce a Stack Cherries variant called *Plus-Minus Stack Cherries*.

Definition 7.1. Let $G > 0$ be a Stack Cherries segment. We define the Plus-Minus Stack Cherries segment $\pm G$ by

$$\pm G = ((G_1), (-G_1), (0))$$

We can use these Plus-Minus Stack Cherries segments to model anti-symmetric Cherries segments as follows. Say we have an anti-symmetric Cherries segment G starting with a black cherry. If we remove the first cherry from this segment, the resulting game is equal to a sum of Stack Cherries segments by Conjecture 5.29. Using basis element decomposition and recomposition, we know that any sum of Stack Cherries segments is equal to a single segment. So $G_{(1,0)} =_C H$ for some Stack Cherries segment H. By symmetry, $G_{(0,1)} =_C -H$. Lastly, if we remove both outer cherries, the resulting segment is again anti-symmetric and thus $G_{(1,1)} =_C 0$. In other words, we have $G =_C ((H), (-H), (0)) = ((K_1), (-K_1), (0)) = \pm\triangleright K$, where we obtain K by prepending H with a single black cherry. If we apply this to the example from before, we find that making a move on the Cherries segment  is equivalent to making a move on the plus-minus Stack Cherries segment $\pm\triangleright$  .

Lemma 7.2. *Let $\pm G$ be a Plus-Minus Stack Cherries segment. Then $\pm G = 0$ in the context of any sum of Stack Cherries games and Plus-Minus Stack Cherries games.*

Corollary 7.3. *Let G, H be Stack Cherries or Plus-Minus Stack Cherries segments. Then $G =_{SC} H$ if and only if $o(G + X) = o(H + X)$ for any sum of Stack Cherries segments and Plus-Minus Stack Cherries segments X.*

So, just like anti-symmetric Cherries segments, adding a Plus-Minus Stack Cherries segment to a sum of regular Stack Cherries segments does not change the value of the game. More interesting is again the matter of finding an optimal strategy in a mixed game of Stack Cherries segments and Plus-Minus Stack Cherries segments; Given a Stack Cherries segment and a Plus-Minus Stack Cherries segment, which would you rather make a move on? In answering this question, we first come across an interesting lemma that tells us a lot about the structure of the set of basis elements.

Lemma 7.4. *Let $B > 0$ be a basis element. Then the greatest basis element that is smaller than B is the concatenation of B and $-B$.*

Proof. Define C as the concatenation of B and $-B$ and let n be the length of B. We will first show that C is actually a basis element. Since C and B match for the first n cherries, after which C is followed by a white cherry and B has reached the end of the segment, we have $C < B$. B and C both start with black cherries so it follows that $|C| < |B|$.

Let C_i be the weakest suffix of C for some $0 \leq i < 2n$. If $n \leq i < 2n$, we have

$$|C| < |B| \leq |B_{i-n}| = |-B_{i-n}| = |C_i|$$

This is a contradiction since C_i is the weakest suffix of C. If $0 < i < n$ and B_i starts with a white cherry, it follows from $C_i < B_i$ that $|C_i| > |B_i| \geq |B| > |C|$. This is again a contradiction. Lastly, consider the case where $0 < i < n$ and B_i starts with a black cherry. If B and B_i do not match for the first $n - i$ cherries, then the mismatch happens before reaching the end of either segment and thus it follows from $|B| \leq |B_i|$ that $|C| < |B| \leq |C_i|$ also holds. If B and B_i do match for the first $n - i$ cherries, then $B[n - i + 1]$ must be white since $|B| \leq |B_i|$ and the $(n - i + 1)$-st cherry of B_i is the end of the segment. Since $C_i[n - i + 1] = C[n + 1]$ is also white, we conclude that $|C_i|$ and $|C_n| = |B|$ match for at least $n - i + 1$ cherries. But C_i is a basis element since it is the weakest suffix of C, so this is a contradiction by Lemma 5.12. We conclude that $i = 0$ must hold, or in other words, that C is a basis element.

Next, we will show that C is the greatest basis element smaller than B by contradiction. In the previous paragraph we saw that $B > C$. Now assume there is a basis element D such that $B > D > C$. If C and D match for at most $n - 1$ cherries, this means that B and D also mismatch within the first n cherries and B comes out on top. Since the first n cherries of C are equal to the first n cherries of B it follows that $C > D$, which is a contradiction. If C and D match for $n \leq \ell < 2n$ cherries, this means that C_n and D_n mismatch within the first n cherries and D_n comes out on top. C_n and D_n both start with a white cherry because C and D are less than B. So $|C_n| > |D_n|$ and the mismatch happens within the first n cherries. Since $B = |C_n|$ and the first n cherries of D match the first n cherries of B it follows that $|D| > |D_n|$. This is a contradiction, since D is a basis element. Lastly, suppose that C and D match for $2n$ cherries. $D[2n + 1]$ must be black since $D > C$ and $C[2n + 1]$ is the end of the segment. Now

$|D|[1...n+1] = D[1...n+1] = C[1...n+1]$ and $|D_n|[1...n] = -D[n+1...2n] = -C[n+1...2n] = C[1...n]$ and $|D_n|[n+1] = -D[2n+1] = C[n+1]$. Thus, $|D|$ and $|D_n|$ match for at least $n+1$ cherries, which is not possible by Lemma 5.12. Note that C and D cannot match for more than $2n$ cherries since C only has length $2n$. We conclude that there is no basis element D such that $B > D > C$ and therefore that C is the greatest basis element smaller than B. □

Example 7.5. The greatest basis element is $\triangleright$[●] (we can see this by, for example, applying Theorem 5.15: any basis element greater than $\triangleright$[●] would have to be infinitely greater, but with a finite segment this is not possible). By Lemma 7.4, the second greatest basis element is $\triangleright$[●|○], then $\triangleright$[●|○|○|●], then $\triangleright$[●|○|○|●|○|●|●|○], and so on.

If we now, for example, look at the basis element $\triangleright$[●|○|○], this lemma also tells us that there are infinitely many basis elements between it and $\triangleright$[●].

Lemma 7.6. *Let $G > 0$ be a Stack Cherries segment with main basis element B. Let C be the concatenation of B and $-B$. Then $G_1 =_{SC} G + C_1$.*

Proof. Let n be the length of B. In the proof of Lemma 7.4, we saw that $|C_n| = |B| \leq |C_i|$ for all $0 < i < 2n$. So the weakest suffix of C_1 is C_n and the backwards Stack Cherries decomposition tells us that $C_1 =_{SC} C[2...n] + C_n = B_1 - B$. Since B is the main basis element of G, we have $G_1 - G =_{SC} B_1 - B =_{SC} C_1$. Adding G to both sides, we conclude that $G_1 =_{SC} G + C_1$. □

Corollary 7.7. *Let $G > 0$ be a Stack Cherries segment with main basis element B. Let C be the concatenation of B and $-B$. Then, for Left, playing on the component G is equivalent to playing on the component $\pm C$ in the context of any sum of (Plus-Minus) Stack Cherries games, independently of what move Right picks.*

Proof. Let X be a sum of Stack Cherries games and Plus-Minus Stack Cherries games. Suppose we are playing the game $G + \pm C + X$. First, assume Right plays the component X to X_j^R. If Left plays on the component G, the resulting game is $G_1 + \pm C + X_j^R =_{SC} G_1 + X_j^R$. If Left plays on the component $\pm C$, the resulting game is again $G + C_1 + X_j^R =_{SC} G_1 + X_j^R$ by Lemma 7.6.

Next, assume Right plays on the component $\pm C$. If Left plays on the component G, the resulting game is $G_1 - C_1 + X =_{SC} G + X$ by Lemma 7.6. If Left plays on the component $\pm C$, the resulting game is again $G + X$.

Lastly, note that Right has no moves on the component G. We conclude that, for Left, playing on G is equal to playing on $\pm C$ in the context of any sum of (Plus-Minus) Stack Cherries games. □

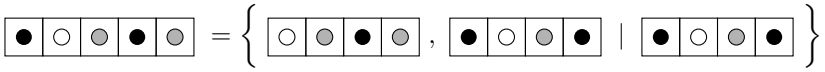
This corollary tells us what the optimal strategy is on any sum of Stack Cherries and Plus-Minus Stack Cherries games:

1. Look at the lexicographically strongest Stack Cherries segment G starting with your color and the lexicographically greatest Plus-Minus Stack Cherries segment $\pm H$.
2. Let B be the main basis element of G and let C be the concatenation of B and $-B$.
3. If C is lexicographically stronger than H, play on the component G. Otherwise, play on the component $\pm H$.

7.2 Gray Cherries

In this subsection, we will take a look at a Cherries variant where, in addition to black and white, cherries may now also be colored gray. Outer gray cherries may be taken by both players. We will first look at the combinatorial version of Gray Cherries.

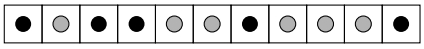
Example 7.8.

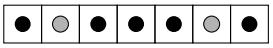


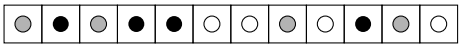
One immediate difference between regular Cherries and Gray Cherries is that Gray Cherries games are not always equal to an integer. Take, for example, the Gray Cherries game  . We have $\in \mathcal{N}$, but all integers are elements of $\mathcal{L}$, $\mathcal{R}$ or $\mathcal{P}$. In combinatorial game theory, the game $= \{0 \mid 0\}$ is often referred to as $*$. Note that $* + * = 0$, since it is a win for the second player.

In the previous sections, we saw that removing a cherry of your own color in a game of regular Cherries makes the game more favorable for the other player. So, intuitively, we expect the players to always prefer taking a gray cherry over taking a cherry of their own color. This means that, if we have two gray cherries right next to each other, both will get taken as soon as possible when revealed, basically canceling out to 0. We now split a Gray Cherries segment up into *sections*, where a section consists of consecutive black or white cherries with gray cherries in between. If we remove all pairs of consecutive gray cherries in a section, we say the *value* of that section is equal to the amount of black/white cherries minus the amount of gray cherries left. If there is an odd amount of gray cherries between two sections, we say that these gray cherries are an *internal separator* of the two sections.

Example 7.9. Consider the following Gray Cherries segment:

$G =$ 

If we remove all pairs of gray cherries, we are left with  , so G consists of a single black section with value 3. Next, consider the following Gray Cherries segment:

$H =$ 

This segment has 1 outer gray cherry, it starts with a black section of value 2, followed by a white section of value 2, followed by a black section of value 1, followed by an internal separator, followed by a last white section of value 1.

Theorem 7.10. *Let G be a Gray Cherries segment consisting of at least 2 sections. Let g be the amount of gray cherries in the outer two blocks. Let m be the value of the first section of cherries and n the value of the last section. Let $\ell, r \in \{-1, +1\}$ be the signs of the first respectively last sections. If the segment contains an internal section of value greater than 1 or an internal separator, let $x \in \{-1, +1\}$ be the sign of the leftmost section that is internal with value greater than 1 or directly follows an internal separator. Similarly, let $y \in \{-1, +1\}$ be the sign of the rightmost section that is internal with value greater than 1 or directly precedes an internal separator. Otherwise, let $x = y = 0$. Then*

$$G = g \cdot * + \ell(m-1) + r(n-1) + \frac{\ell + r}{2} + \frac{x + y}{2}.$$

Proof. Similar to the proof of Theorem 3.1, except that we now say that a move by Left is optimal if it is to the game $v - 1$ or $v + *$ and that a move by Right is optimal if it is to the game $v + 1$ or $v + *$. The desired result then follows from the fact that a move by Left to $v - 1$ results in the same outcome as a move to $v + *$ and similarly for Right. More precisely, for integers a, b with $a \leq b$, we have

$$\{a + * \mid b + *\} = \{a - 1 \mid b + *\} = \{a + * \mid b + 1\} = \{a - 1 \mid b + 1\}.$$

□

Example 7.11. Consider again the Gray Cherries segment H from Example 7.9. Plugging in all the values we found in that example, Theorem 7.10 tells us that $H = 1 + *$.

Lastly, we will very briefly look at the synchronized version of Gray Cherries. The first thing to note is that not all Gray Cherries segments are strongly separable. Take for example the game  $\in \mathcal{N}$. By Theorem 4.16, this game cannot be strongly separable. This stems from the fact that it is not clear what synchronized move should correspond to both players taking the same gray cherry. For now, we will say that the synchronized move corresponding to both players taking the same gray cherry is simply the game where that single cherry has been removed, but other approaches are possible.

Just like in the combinatorial case, we expect the players to prefer taking gray cherries over cherries of their own color. What this means with our definition of the synchronized moves, is that these gray cherries have no impact on the value of the game: as soon as outer gray cherries become available, both players will pick them, resulting in the same game as when those gray cherries were never there. After introducing yet another restricted version of equality, we will write out this theorem.

Definition 7.12. Let G, H be synchronized games. We say $G =_{GC} H$ if $o(G + X) = o(H + X)$ for any sum of Gray Cherries segments X.

Note that "any sum of Gray Cherries segments X" also includes all sums of regular Cherries segments.

Theorem 7.13. *Let G be a Synchronized Gray Cherries segment. Let H be the Synchronized Cherries segment we get by removing all the gray cherries from G. Then $G =_{GC} H$.*

8 Conclusions and Further Research

In this thesis, we considered two main ways of playing the game Cherries. We started off with the combinatorial version of the game, and, using some well-established theorems from combinatorial game theory, we proved that any Combinatorial Cherries game is equal to some integer game.

Next, we introduced our definition of a synchronized game and defined some basic operations on these games. We saw that our definition of equality was more difficult to work with than the equality on combinatorial games, mostly due to the uncertainty of not knowing what move the opponent will pick. We then saw that some combinatorial games, including Cherries, naturally give rise to synchronized versions of these games, and that the outcome class of a combinatorial game often fully determines the outcome class of such a synchronized version.

Section 5 contains our game-theoretical analysis of Synchronized Cherries. We saw that Cherries segments often naturally reduce to Stack Cherries segments, where the players may only take cherries from the front of a segment. As a main result, we showed that Stack Cherries segments can be decomposed into so-called basis elements, which gives us a very good measure of the value of any Stack Cherries game and allows us to easily determine its outcome class. Lastly, we proposed a method for actually reducing any Cherries segment into Stack Cherries segments, but were not yet able to completely prove its correctness.

The theory of Section 5 outlines an efficient three-step method for determining the outcome class of any Cherries game: decompose the Cherries segments into Stack Cherries segments, decompose these Stack Cherries segments into basis elements, and determine the outcome class of the resulting sum of basis elements. Using suffix arrays, we designed an algorithm for the second step that runs in linear time in the length of the segments. We also presented a linear time algorithm for the third step, using hash tables. Combining these two steps gives us a linear time algorithm for determining the outcome class of any Stack Cherries game. Lastly, we presented a quadratic time algorithm for the first step based on our Cherries decomposition conjecture, meaning that the total algorithm has quadratic time complexity.

There are multiple possible directions for future research on the topic of Synchronized Cherries. Most important would be to either finish the proof that our method of decomposing Cherries segments into Stack Cherries segments is correct, or find a counter-example that shows that it is not. We motivated why such a decomposition makes sense, proved some theorems that hint at its correctness and showed that it works for at least all Cherries games containing at most 20 Cherries, but a complete proof is still missing.

Another direction for future research would be to try to find a simpler, explicit winning strategy for playing Synchronized Cherries. Currently, the simplest winning strategy we have implicitly follows from the value we can assign to segments using decomposition: a winning strategy for either player is to always pick a move such that the decomposition of the resulting game is a win for that

player for all possible moves of the opponent, if possible. But this strategy is quite tedious, since one would have to find the basis element decomposition for every possible move to determine if there are any winning moves. Also note that we are using the words "winning strategy" instead of "optimal strategy", since Left having an optimal strategy on a synchronized game G might seem to imply that she has some move such that, for all moves by Right, the resulting game is greater than when she would have picked another move, which is not always the case with Synchronized Cherries (take for example the game $G =$).

Finally, one could try to find a more efficient algorithm for determining the outcome of a Synchronized Cherries game, specifically for the first step: decomposing a Cherries segment into Stack Cherries segments. With a quadratic time complexity, this step is currently the slowest, so improving the time complexity of this step would improve the complexity of the entire algorithm.

References

[AKO04] Mohamed Ibrahim Abouelhoda, Stefan Kurtz, and Enno Ohlebusch. Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms*, 2(1):53–86, 2004.

[ANW19] Michael H. Albert, Richard J. Nowakowski, and David Wolfe. *Lessons in Play: An Introduction to Combinatorial Game Theory*. CRC Press, second edition, 2019.

[BCG01] Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. *Winning Ways for Your Mathematical Plays*, volume 1. Taylor & Francis, second edition, 2001.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, third edition, 2009.

[Con00] John H. Conway. *On Numbers and Games*. CRC Press, second edition, 2000.

[Gus97] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

[KS99] Stefan Kurtz and Chris Schleiermacher. REPuter: Fast computation of maximal repeats in complete genomes. *Bioinformatics*, 15(5):426–427, 1999.

[LLH18] Zhize Li, Jian Li, and Hongwei Huo. Optimal in-place suffix sorting. In *International Symposium on String Processing and Information Retrieval*, pages 268–284. Springer, 2018.

[MM93] Udi Manber and Gene Myers. Suffix arrays: A new method for on-line string searches. *SIAM Journal on Computing*, 22(5):935–948, 1993.

[Sie13] Aaron N. Siegel. *Combinatorial Game Theory*. American Mathematical Society, 2013.