Sipke Castelein

# Order and Chaos

Bachelor thesis

September 13, 2019

Thesis supervisors: M.J.H. van den Bergh MSc
Dr. F.M. Spieksma

Universiteit Leiden
Mathematisch Instituut

## Abstract

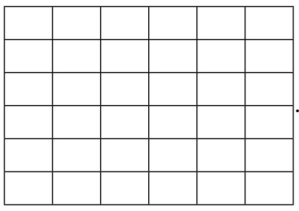
Order and Chaos is a 2-player board game on a $6 \times 6$ board where we have two players: Order and Chaos. Order starts and has to make a row of 5, by playing 2 symbols. Chaos has to prevent that, by playing the same 2 symbols. This thesis consists of two main parts. In the first part we determine a winning strategy for Order, by setting up a lemma, using Monte-Carlo Treesearch and using brute force Depth-First search. In the second part we examine variants of the game, where we slightly change the rules. In the first variant Chaos starts instead of Order. This version has turned out to be winning for Order. In the second variant, Order only wins by creating a row of 5, while a row of 6 is not winning anymore for Order. We can prove that this version is winning for Chaos. The last version we examine, is the variant where Chaos starts and Order does not win with a row of 6. This game is winning for Order.

# Contents

# 1 Introduction

Order and Chaos is a board game for two players. It is a variant of tic-tac-toe and it was invented by Stephen Sniderman. He introduced this game in Games magazine in 1981 [1]. Order and Chaos is played on a $6 \times 6$ board with 6 fields in length and 6 fields in height. Both players have a different goal. Order's goal is to make a row of 5 consecutive identical symbols, and Chaos' goal is to have a full board without a consecutive row of 5 symbols. Every move, both Order and Chaos are able to fill an empty field with either an X or an O. The first player to do a move is Order. This is the classic version of the game. However, it is easy to make some adjustments to this game, so there are variants with slight rule changes.

The board we will be playing on, looks as in (1). In every empty field, the players are permitted to place either X or O.


. (1)

Until now, people have only proved this game to be winning for Order [8], but no specific winning strategy has been constructed and other versions of the game have not been studied. We only know the fact that there exists a winning strategy and that a computer is able to play along this strategy using a script. The gap that we have right now, is that no human, but only the computer is able to generate a winning strategy. Our aim is to determine a winning strategy that is playable for humans and is winning. Furthermore, this game has never been defined in a mathematical way. To do some proper analysis on this game, we need definitions and specific notation.

This is an interesting topic to do research on. Therefore, in the second section, we define Order and Chaos in a mathematical way, so that we can set up some strategies, lemmas and conditions. We will need these in the third chapter, where we set up a lemma that allows us to reduce the size of the problem from a $6 \times 6$ to a $4 \times 4$ board. This lemma has already been used in other research, but it has not been rigorously proved yet. An example of research is Daniël van Gent's work on Order versus Chaos. This has not been published, but it provided a great basis for this thesis. After finishing the proof of this lemma, we have determined a strategy using Monte-Carlo Treesearch, which we will validate afterwards using Depth-First search.

In the fourth section, our aim is to analyse variants of this game. We can modify this game by, for example, changing the size of the board, adding more players or adding more symbols. There are many more features of this game that can be adjusted. We have chosen to study the problem what happens when Chaos starts instead of Order, what happens when a row of 6 is not winning for Order and lastly a combination of both, where Chaos starts and Order does not win with a row of 6. Finally, we will provide a strategy for the winning players and prove that these strategies are winning.

## 2 Definitions and notation

Order and Chaos is a game that is played on a $6 \times 6$ board. We have two players, Order and Chaos. Order's goal is to make a row of length 5 and Chaos' goal is to prevent that from happening. Both players can pick X and O for their move. This means that one move they can play O and the next move X. This is the classic version of Order and Chaos. Order and Chaos is a 2-player, perfect information game that belongs to the class of combinatorial games. More precisely, it is an impartial game, as both players can make the same moves, and it is also finite, since one player can never skip a move and we will always reach an end state [7].
Of course we will study this classic game, but it may also be interesting to look how strategies change as we change the rules, the board or the amount of symbols. In order to do this, we need clear definitions. We will provide these next.

**Definition 2.1.** We define $\mathrm{OvC}_{n\times m}(\ell, s, p)$ an *Order versus Chaos game* as follows:

For $n \leq m$, the parameters $n$ and $m$ are the length and width of the board, respectively. Without loss of generality, we can say that, if $m < n$, $\mathrm{OvC}_{m\times n}(\ell, s, p)$ is the same as $\mathrm{OvC}_{n\times m}(\ell, s, p)$ if we make a rotation of 90°. That means that if $m < n$, we can swap the values ($n_{\text{new}} = m$ and $m_{\text{new}} = n$) and we get an equivalent game $\mathrm{OvC}_{n_{\text{new}}\times m_{\text{new}}}(\ell, s, p)$.
We let $s$ be the amount of symbols a player can choose from . By $S^-$ we define a set of symbols of size $s$ without the blank symbol '$\square$'. Then, we also define $S = S^- \cup \{\square\}$. Order wins if there is a row with a length at least $\ell$ of one identical symbol $s \in S^-$. Note that $\ell \leq \min(n, m)$. Only then we can make horizontal, vertical and diagnoal lines. We define the board $B = (b_{ij})_{ij} \in S^{n\times m}$ as a matrix with element $b_{ij} \in S$, $1 \leq i \leq n$, $1 \leq j \leq m$. Lastly, $p$ is the player (Order or Chaos) that begins the game.

**Notation 2.1.** Let $n \in \mathbb{N}_{\geq 1}$, we say $[n] = \{1, \ldots, n\}$. When $n = 0$, we say $[n] = \{0\}$.

**Definition 2.2.** Let $(i, j) \in [n] \times [m]$, and $s_L = b_{ij} \in S^-$ be given. Then, we define the lengths $L_{\rightarrow}((i,j)), L_{\uparrow}((i,j)), L_{\nearrow}((i,j)), L_{\searrow}((i,j))$ of the *horizontal, vertical, ascending diagonal* and *descending diagonal consecutive row* respectively as follows:

1) We say, that $L_{\rightarrow}((i,j)) = \ell_L$ if there exist $y \in [m]$ and $k \in [\ell_L]$ with $j = y + k - 1$ and $y + \ell_L - 1 \leq m$, such that:
   - $b_{i,y} = b_{i,y+1} = \ldots = b_{i,y+\ell_L-1} = s_L$.
   - When $y \neq m$, $b_{i,y+\ell_L} \neq s_L$.
   - When $y \neq 1$, $b_{i,y-1} \neq s_L$.

   The *extensions* of this row are
   - $b_{i,y-1}$, if $y \neq 1$,
   - $b_{i,y+\ell_L}$, if $y \neq m$.
2) We say, that $L_{\uparrow}((i,j)) = \ell_L$ if there exist $x \in [n]$ and $k \in [\ell_L]$ with $i = x + k - 1$ and $x + \ell_L - 1 \leq n$, such that
   - $b_{x,j} = b_{x+1,j} = \ldots = b_{x+\ell_L-1,j} = s_L$.
   - When $x \neq n$, $b_{x+\ell_L,j} \neq s_L$.
   - When $x \neq 1$, $b_{x-1,j} \neq s_L$.

The *extensions* of this row are

- $b_{x-1,j}$, if $x \neq 1$,
- $b_{x+\ell_L,j}$, if $x \neq n$.

3) We say, that $L_{\searrow}((i,j)) = \ell_L$ if there exist $x \in [n]$, and $k \in [\ell_L]$ with $i = x+k-1$, $j = x+a+k-1$, $x+\ell_L-1 \leq n$, $x+a \geq 1$ and $x+a+\ell_L-1 \leq m$ for $a = j-i$ such that
   - $b_{x,x+a} = b_{x+1,x+a+1} = \ldots = b_{x+\ell_L-1,x+a+\ell_L-1} = s_L$.
   - When $x \neq 1$ and $x+a \neq 1$, $b_{x-1,x+a-1} \neq s_L$.
   - When $x+\ell_L-1 \neq n$ and $x+a+\ell_L-1 \neq m$, $b_{x+\ell_L,x+a+\ell_L} \neq s_L$.

   The *extensions* of this row are
   - $b_{x-1,x+a-1}$, if $x \neq 1$ and $x+a \neq 1$,
   - $b_{x+\ell_L,x+a+\ell_L}$, if $x+\ell_L-1 \neq n$ and $x+a+\ell_L-1 \neq m$.

4) We say, that $L_{\nearrow}((i,j)) = \ell_L$ if there exist $x \in [n]$ and $k \in [\ell_L]$ with $i = x-k+1$, $j = m-x-a+k-1$, $x-\ell_L+1 \geq 1$, $m-x-a \geq 1$ and $m-x-a+\ell_L-1 \leq m$ for $a = m-j-i$ such that
   - $b_{x,m-x-a} = b_{x-1,m-x-a+1} = \ldots = b_{x-\ell_L+1,m-x-a+\ell_L-1} = s_L$.
   - When $x \neq n$ and $m-x-a \neq 1$, $b_{x+1,m-x-a-1} \neq s_L$.
   - When $x-\ell_L+1 \neq 1$ and $m-x-a+\ell_L-1 \neq m$, $b_{x-\ell_L,m-x-a+\ell_L} \neq s_L$.

   The *extensions* of this row are
   - $b_{x+1,m-x-a-1}$, if $x \neq n$ and $m-x-a \neq 1$,
   - $b_{x-\ell_L,m-x-a+\ell_L}$ if $x-\ell_L+1 \neq 1$ and $m-x-a+\ell_L-1 \neq n$.

**Definition 2.3.** Let $(i,j) \in [n] \times [m]$, then the *longest consecutive row* is defined by

$$L((i,j)) = \max_{x \in \{\rightarrow,\uparrow,\nearrow,\searrow\}} \{L_x((i,j))\}.$$

**Notation 2.2.** Let $(i,j) \in [n] \times [m]$. We say that $(i,j)$ are the *coordinates* of $b_{ij}$.

**Notation 2.3.** Let $A \subset [n] \times [m]$. For the board $B$ we say that $B|_A$ are the elements $b_{ij}$ with $(i,j) \in A$.

**Notation 2.4.** Let $(i,j) \in [n] \times [m]$, then

- $b_{i\bullet} = (b_{i,1}, b_{i,2}, \ldots, b_{i,m})$.
- $b_{\bullet j} = (b_{1,j}, b_{2,j}, \ldots, b_{n,j})^\top$.
- $b_{\searrow(i,j)} = (b_{i-k,j-k}, b_{i-k+1,j-k+1}, \ldots, b_{i+l,j+l})^\top$, for $k = \min(i,j)$ and $l = \min(n-i, m-j)$.
- $b_{\nearrow(i,j)} = (b_{i+k,j-k}, b_{i+k-1,j-k+1}, \ldots, b_{i-l,j+l})^\top$, for $k = \min(n-i, j)$ and $l = \min(i, m-j)$.

**Notation 2.5.** Let $A \subset [n] \times [m]$, then $b_{i\bullet}|_A$ is the longest vector of consecutive elements $b_{ij}$ such that $(i,j) \in A$. This is symmetric for $b_{\bullet j}, b_{\searrow(i,j)}$ and $b_{\nearrow(i,j)}$.

**Notation 2.6.** For a vector $v \in S^m$ and a symbol $s \in S^-$, we take $\mathbb{1}_s(v)$ componentwise.

**Example 2.1.** Consider

| | | | | | |
|---|---|---|---|---|---|
| X | | | | | O |
| | X | X | | O | X |
| | | X | O | | |
| | | X | | X | |
| | O | | | X | |
| | | | O | | O |

.

Then $\mathbb{1}_{\mathrm{X}}(b_{\searrow(1,1)}) = \mathbb{1}_{\mathrm{X}}(\mathrm{X}, \mathrm{X}, \mathrm{X}, \square, \mathrm{X}, \mathrm{O}) = (1, 1, 1, 0, 1, 0)$.

**Notation 2.7.** $\vec{1} \in S^m$ is the vector consisting of ones.

**Definition 2.4.** Let $(i, j) \in [n] \times [m]$. Choose $A = \{x_1, \dots, y_1\} \times \{x_2, \dots y_2\}$ with $y_1 - x_1 + 1 = \ell$ and $y_2 - x_2 + 1 = \ell$. We define the *horizontal, vertical, ascending diagonal* and *descending diagonal open lines* of $s \in S^-$ as $P_{\rightarrow}((i, j), s)$, $P_{\uparrow}((i, j), s)$, $P_{\nearrow}((i, j), s)$, $P_{\searrow}((i, j), s)$ respectively as follows:

$$P_{\rightarrow}((i,j),s) = \begin{cases} \mathbb{1}_s(b_{i\bullet}|_A)\vec{1}, & \text{if } \mathbb{1}_{s'}(b_{i\bullet}|_A)\vec{1} = 0, \text{ for all } s' \in S^- \backslash \{s\}, \\ 0, & \text{otherwise;} \end{cases}$$

$$P_{\uparrow}((i,j),s) = \begin{cases} \mathbb{1}_s(b_{\bullet j}|_A)^\top \vec{1}, & \text{if } \mathbb{1}_{s'}(b_{\bullet j}|_A)^\top \vec{1}, \text{ for all } s' \in S^- \backslash \{s\}, \\ 0, & \text{otherwise;} \end{cases}$$

$$P_{\searrow}((i,j),s) = \begin{cases} \mathbb{1}_s(b_{\searrow(i,j)}|_A)^\top \vec{1}, & \text{if } (i,j) = (x_1 + k, x_2 + k) \text{ for } 0 \le k \le \ell - 1 \text{ and} \\ & \mathbb{1}_{s'}(b_{\searrow(i,j)}|_A)^\top \vec{1} = 0, \text{ for all } s' \in S^- \backslash \{s\}, \\ 0, & \text{otherwise;} \end{cases}$$

$$P_{\nearrow}((i,j),s) = \begin{cases} \mathbb{1}_s(b_{\nearrow(i,j)}|_A)^\top \vec{1}, & \text{if } (i,j) = (y_1 - k, x_2 + k) \text{ for } 0 \le k \le \ell - 1 \text{ and} \\ & \mathbb{1}_{s'}(b_{\nearrow(i,j)|_A})^\top \vec{1} = 0, \text{ for all } s' \in S^- \backslash \{s\}, \\ 0, & \text{otherwise.} \end{cases}$$

Note that $A$ can differ every value! We always let $A$ such that the value is maximized under the given restrictions.

**Definition 2.5.** For a point $(i, j) \in [n] \times [m]$ and symbol $s \in S^-$, we define the *maximum feasible line* as

$$P((i,j),s) = \max_{x \in \{\rightarrow, \uparrow, \searrow, \nearrow\}} P_x((i,j),s).$$

**Example 2.2.** For $\text{OvC}_{6\times 6}(5, 2, \text{O})$ we have the following board:

| | | | | | |
|---|---|---|---|---|---|
| X | | | | | O |
| | X | X | | O | X |
| | | X | O | | |
| | | X | | X | |
| | O | | | X | |
| | | | O | | O |

.

Then we have $L((1,1)) = L_{\searrow}((1,1)) = 3$, $P((1,1),\text{X}) = P_{\searrow}((1,1),\text{X}) = 4$, $L((1,6)) = L_{\nearrow}((1,6)) = 3$ and $P((1,6),\text{O}) = P_{\rightarrow}((1,6),\text{O}) = 1$.

**Definition 2.6.** We say $B$ is *in order*, if there exists a $(i,j) \in [n] \times [m]$ with a row $L((i,j)) \geq \ell$.

**Definition 2.7.** We say $B$ is *chaotic*, if for all $(i,j) \in [n] \times [m]$, we have $b_{ij} \neq \square$ and $B$ is not in order.

Order wins if he manages to create an in order board, while Chaos wins if he can create a chaotic board. We call $B$ unstable if $B$ is neither in order nor chaotic, hence there are still empty fields. The game starts with an empty board $E = (e_{ij})_{ij}$, with $e_{ij} = \square$ for all $(i,j) \in [n] \times [m]$.

**Definition 2.8.** We describe *the set of empty fields* $\mathcal{M}$ as

$$\mathcal{M} = \{(i,j) \in [n] \times [m] \mid b_{ij} = \square\}.$$

**Example 2.3.** An Order versus Chaos game $\text{OvC}_{6\times 6}(5, 2, O)$ with $S = \{\text{X}, \text{O}\}$ always begins with board $E$:

$E =$

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

.

**Example 2.4.** This board is in order:

| | | | | | |
|---|---|---|---|---|---|
| O | X | | | | |
| | | X | | O | |
| | X | O | X | | |
| | | | | X | |
| | | | | O | X |
| | O | | | | |

.

Here we see that we have a descending diagonal line of 5 consecutive X's, hence $L((1,2)) = L_{\searrow}((1,2)) = 5 = \ell$. In this case, Order wins the game.

**Example 2.5.** This is an example of a chaotic board:

| | | | | | |
|---|---|---|---|---|---|
| O | X | O | O | X | O |
| O | X | O | X | O | O |
| X | X | O | O | O | X |
| X | O | X | X | X | O |
| O | X | O | X | O | X |
| O | O | X | X | X | O |

.

Here we see that for all $(i, j) \in [6] \times [6]$, $L((i, j)) < 5$ and $b_{ij} \neq \square$, hence Chaos wins this game.

**Definition 2.9.** Let $B$ be stable. We describe a *move* $M_B \colon \mathcal{M} \times S^- \to S^{n\times m}$ defined by $M_B((i, j), s) = B' = (b'_{ij})_{ij}$, and

$$b'_{ab} = \begin{cases} b_{ab} & \text{if } (a, b) \neq (i, j), \\ s & \text{if } (a, b) = (i, j). \end{cases}$$

**Example 2.6.** Let $B =$

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | X | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

, then $M_B((3, 4), \mathrm{X}) =$

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | X | | | |
| | | | X | | |
| | | | | | |
| | | | | | |
| | | | | | |

.

**Definition 2.10.** Let $A \subset [n] \times [m]$. $S^{|A|}$ is the set of possible partition boards $A$.

**Definition 2.11.** A *strategy* is described as $T \colon S^{n\times m} \to \mathcal{M} \times S^-$.

This means that the player looks at the board, and reacts to the condition the board is in. The output elements are coordinates and a symbol. Note that a move is not the same as a strategy. There is a move assigned to each board.

**Definition 2.12.** Let $A \subset [n] \times [m]$. The map $p \colon S^{n\times m} \to S^{|A|}$ defined by $p(B) = B|_A$, is called a priority function.

**Definition 2.13.** Let $T \colon S^{n\times m} \to \mathcal{M} \times S^-$ be a strategy, $A_1 \subset [n] \times [m]$ and $A_2 = [n] \times [m] \backslash A_1$. We say that $T$ *seperates* into $T_1 \colon S^{|A_1|} \to \mathcal{M} \times S^-$ and $T_2 \colon S^{|A_2|} \to \mathcal{M} \times S^-$ if there exists a priority function $p$, and a function $f \colon S^{n\times m} \to \mathcal{M} \times \mathcal{S}^-$ called a *transition function*, such that

$$T(B) = \begin{cases} T_1(B|_{A_1}) & \text{if } p(B) = B|_{A_1} \text{ and } B|_{A_1} \text{ unstable}, \\ T_2(B|_{A_2}) & \text{if } p(B) = B|_{A_2} \text{ and } B|_{A_2} \text{ unstable}, \\ f(B) & \text{otherwise}. \end{cases}$$

We use the notation $T = T_1 \cup T_2$. The function $f$ gives us the strategy if $p(B) = B|_{A_1}$, but $B_{A_1}$ is stable or if $p$ outputs someting different than $B|_{A_1}$ or $B|A_2$, and if $p(B) = B|_{A_2}$, but $B_{A_2}$ is stable or if $p$ outputs someting different than $B|_{A_1}$ or $B|_{A_2}$.

## 3 Classic game

In this section, we want to find a winning strategy for the classic Order and Chaos game. The rules are as follows: we play on a $6 \times 6$ board, Order begins, both players can choose between symbols X and O; Order wins if there is a row of length 5 or longer and Chaos wins if the board is full and there is no row of length 5 or longer. In [8] it was shown by computer that Order has a winning strategy. We prove this rigorously by setting up a lemma in section 3.1, using Monte Carlo Treesearch in section 3.2, analysing the outcome of Monte Carlo Treesearch in section 3.3 and checking it using a brute force search alogrithm in section 3.4. In this section we will go these methods in further detail and show how we got the result.

### 3.1 Lemma

A $6 \times 6$ board is big to work with, so we want to reduce the size of this problem. To do so, we decided to split the board into two parts: the middle part and the border. The following lemma allows us to reduce the size of the problem.

**Lemma 3.1.** If $\mathrm{OvC}_{n\times n}(n, 2, p)$ is winning for Order, then $\mathrm{OvC}_{(n+2)\times(n+2)}(n + 1, s, p)$ is also winning for Order.

*Proof.* Suppose that $\mathrm{OvC}_{n\times n}(n, 2, p)$ is winning for Order. That means that on a board $B_1 \in S^{n\times n}$, Order can win by playing strategy $T_1$ with move $F := M((i, j), s)$ if $T_1(B_1) = ((i, j), s)$ for an empty board $B_1$. At the end of the game, there exists $(i, j) \in [n] \times [m]$, with $L((i, j)) = n$.

Now consider board $B \in S^{(n+2)\times(n+2)}$. This board can be partitioned into $A_1 \sqcup A_2$, with $A_1 := \{2, \dots, n + 1\}^2$ and $A_2 := \{1, \dots, n + 2\}^2\backslash A_1$. This means that $A_1 \in S^{n\times n}$ is equivalent to an $n \times n$ board, and $A_2$ is the border around it. We define *mirroring* with $m\colon A_2 \to A_2$ as follows

$$(i, j) \mapsto \begin{cases} (n + 1 - i, j) & \text{if } i \in \{1, n + 2\} \text{ and } j \notin \{1, n + 2\} \\ (i, n + 1 - j) & \text{if } i \notin \{1, n + 2\} \text{ and } j \in \{1, n + 2\} \\ (n + 1 - i, n + 1 - j) & \text{if } i, j \in \{1, n + 2\}. \end{cases}$$

This means that $m((i, j))$ gives the coordinates, on the exact opposite fields on the border of the board.

Now we have the following priority function

$$p(B) = \begin{cases} B|_{A_2} & \text{if there exists } (i, j) \in A_2 \text{ with } b_{ij} \neq \square \text{ and } b_{m(i,j)} = \square, \\ B|_{A_1} & \text{otherwise.} \end{cases}$$

We describe strategy $T_2$ as follows: we only play $T_2$, if there exists a $(i, j) \in A_2$ with $b_{ij} = s \in S^-$ and $b_{m(i,j)} = \square$. Our strategy in that case is, that $T_2(B) = (m(i, j), s')$ for $s' \in S^-\backslash\{s\}$.

We claim that $T := T_1 \cup T_2$ with transition function 'Bring the board in order' is a winning strategy for $\mathrm{OvC}_{(n+2)\times(n+2)}(n - 1, 2, p)$. In the next part we will show that this is winning.

In the beginning we have two possibilities: Order starts, or Order does not start. If Order is the starting player, he makes the first move $F$ on $B|_{A_1}$. From this moment on, we can use the same strategy for both cases. Order reacts on the move Chaos makes. Again we have two possibilities

1) Chaos plays move $M((i,j),s)$ with $(i,j) \in A_1$.

2) Chaos plays move $M((i,j),s)$ with $(i,j) \in A_2$.

If 1) happens, we just play our strategy $T_1$, and if 2) happens, we play $T_2$. We know that $A_2$ is the border of a square, so $|A_2|$ is even. Order only plays on $A_2$ if Chaos has played there the previous turn. This means that Order will always be the last player that plays there, so there will never be a transition function going from $A_2$ to $A_1$.

After a finite number of turns, for the resulting board $B$ it holds that $B|_{A_1}$ is stable. Suppose that Order has not yet won. Since Order has played along $T_1$, there is a point $(i,j) \in \{2,\dots,n+1\}^2$ with $L((i,j)) = n$ restricted on $B|_{A_1}$ and $s_L = b_{ij}$. Now it is Chaos' turn. Let $(a,b)$ an extension of row $L((i,j))$, then $(a,b) \in A_2$ and we have two possibilities (direct consequence of playing $T_2$), namely:

1) $b_{ab} = b_{m(a,b)} = \square$.

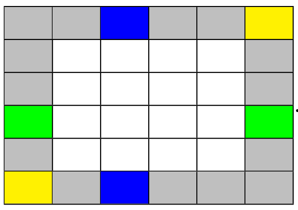
2) $b_{ab} = s$, $b_{m(a,b)} = s' \in S^- \backslash \{s\}$ or other way around.

In the case of 2), we already know that Order has won, because, restricted on $B$, $L((i,j)) = n+1$. For case 1) note that Chaos could have made a move on $(a,b)$ or $m(a,b)$ since the last observation. This means we have to define the following transition function

$$f(B) = \begin{cases} ((a,b), s_L), & \text{if } b_{ab} = \square; \\ (m(a,b), s_L), & \text{if } b_{(m(a,b))} = \square. \end{cases}$$

Note that restricted on $B$, $L((i,j)) = n+1$. Order wins, and $\mathrm{OvC}_{(n+2)\times(n+2)}(n+1,2,p)$ is winning for Order. $\square$

**Example 3.1.**

$B =$  .

Here we see that $B|_{A_1}$ is the white area and $B|_{A_2}$ is the grey area on the border. In addition to this, some symbols placed are on $B|_{A_2}$. If we mirror the corresponding coordinates, we have the other symbol on the other side of the border. For example, the green coordinate $(1,4)$ mirrored is the other green coordinate $(4,6)$, the blue coordinate $(1,3)$ mirrored is the ohter blue coordinate $(6,3)$ and the yellow coordinate $(6,1)$ mirrored is the other yellow coordinate $(1,6)$.

## 3.2 Monte-Carlo Treesearch

By virtue of Lemma 2.1 we know that $\mathrm{OvC}_{6\times 6}(5,2,\mathrm{O})$ is winning for Order, if there is a winning strategy for $\mathrm{OvC}_{4\times 4}(4,2,\mathrm{O})$. On the border of the board our strategy $T_2$ is mirroring, as

described in the proof of the lemma, so now we need a strategy $T_1$ for the middle part of the board.
To get to know a good strategy Order has to play to win, we use the Monte-Carlo treesearch algorithm. This section is primarily based on [3] and [9]. The classic approach to game AI requires either very high quality knowledge of the game, or a very long computational time. First, we have attempted a brute force method to find a strategy. However, knowing that there are $3^{16}$ different game states, it is never possible to determine an optimal solution. Monte-Carlo treesearch has turned out to be the perfect solution to this problem.

### 3.2.1 What is Monte-Carlo Treesearch?

Monte-Carlo treesearch (MCTS) is a best-first search technique combined with stochastic simulations. MCTS builds a tree from the root node, which is the empty board in our case. Every simulation, it adds nodes to the tree. In the beginning, the MCTS does not know anything, but it is a self-learning algorithm. That means, that the algorithm starts by playing random moves, gets an outcome, learns from it and with that extra knowledge, it is supposed to get a better result next time. The algorithm does a pre-specified amount of simulations and the goal is to gain enough information to obtain a satisfactory result.
The general idea of this algorithm is to only visit nodes in the tree that have a promising outcome. This means that we cut off a big portion of the tree, because the algorithm tends not to go to a node without a promising outcome.
The following mechanism makes sure that we only build promising game states:

#### Selection

Given board $B \in S^{n \times m}$, we can make different boards $B_1, \ldots, B_n$ by one single move. In the continuation of this section, a node in the search tree is a board of $S^{n \times m}$ in a specific state. Every node $i$ has some statistics, namely

- $v_i$: value of node $i$;
- $n_i$: visit count of node $i$;
- $r_{t,j}$: result of game $t$ (0 if lost, 1 if won) for player $j$, if he made the last move to reach this state;
- $R_{i,j} = \sum\limits_t r_{t,j}$: cumulative score of node $i$, where player $j$ did the last move to reach node $i$, of all the simulations.

We compute the value of node $i$ by $v_i = \frac{R_{i,j}}{n_i}$. Here we see that $v_i$ is the fraction of games won from node $i$.

We want to visit the board that leads to the best result (exploitation), but we also do not want to miss out on good moves we have not discovered yet (exploration). The most popular solution for this is the one we have used. It is called Upper Confidence Bounds applied to Trees (UCT). To explain this, let

$$I = \bigcup_{i=1}^{n} B_i$$

be the set of reachable states from board $B$. We select a child $B'$ that satisfies

$$B' \in \arg\max_{i \in I} \left( v_i + C\sqrt{\frac{\ln(n_B)}{n_i}} \right)$$

where $C$ is a constant. The only restricion we have is that $v_i \in [0,1]$. This is handled by the way we have computed $v_i$, namely it is the fraction of games won from node $i$. In this manner, we do not only look at the value $v_i$, but using $n_B$ and $n_i$, we also take into account whether a node has been visited a lot. It may occur that a node has value 0, because it has only been visited once. Because a path was chosen deterministicaly, this might still be a very promising node. Now, we see in our formula that the longer the node has not been chosen, the more likely this node will be chosen next time.
All in all, the term $v_i$ takes care of the exploitation, while the term $C\sqrt{\frac{\ln(n_B)}{n_i}}$ makes sure we also do some exploration. The bigger $C$, the more likely the algorithm is to explore new states.

**Expansion**

After some amount of moves, we will see that if we do a certain move, this resulting state has not been saved as a node in the tree, hence it does not have a value. When we see this happening, we will always select this state, no matter how promising the other nodes are, and save it in the tree. In this manner, we reduce the chances that we miss a good strategy. We also save memory by not playing all the unknown nodes and, because we only add one leaf, the quality of play reduces only slightly.

**Play-out**

After the expansion, we play random moves or semi-random moves until we reach a board that is either in order or chaotic. In order to make semi-random moves, we already need a simulation strategy. We do not have this, so we just use fully random moves. This might cause the simulations to take a bit longer, because we will need more simulations to get to a satisfactory result.

**Backpropagation**

At some point, we come to an end state and the values $r_{t,1}$ and $r_{t,2}$ are computed. This result is propagated from the end state, through all the previously visited nodes, all the way back to the root. When we visit a node $i$, we update $R_{i,j}$, $v_i$ and add 1 to $n_i$. Now every node we have visited has an updated and more correct value of how promising this node is.

**Example**

Figure 1 shows us an example of MCTS. The numbers of the nodes are the values, written as fractions. First, we see that the algorithm selects the nodes according to the selection mechanism. Then we see that it comes across a state with no value $(\frac{0}{0})$. From that point on, it simulates until the end, the white player loses and the grey player wins. These values are backpropagated all the way up to the root node. In this way, the values are updated and are a bit more accurate than they were before.

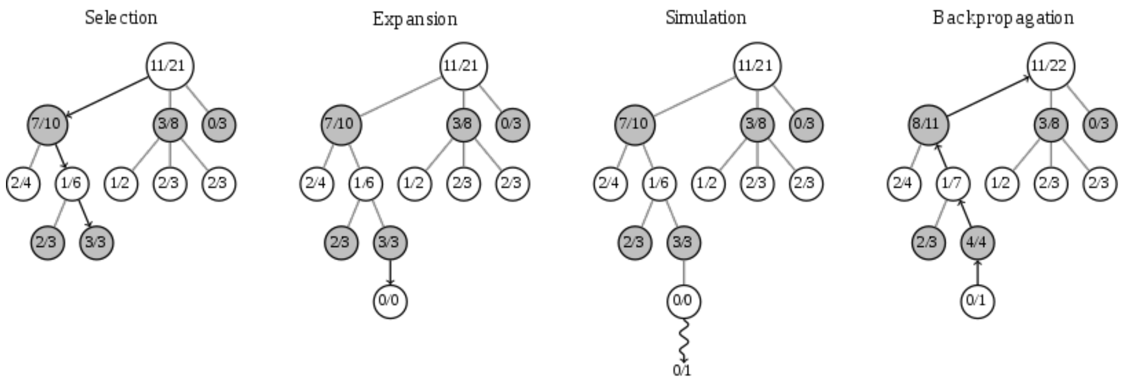


Figure 1: Monte-Carlo Treesearch in action.

#### 3.2.2 Implementation

We implement a script based on [6] in Python. We select $C = \sqrt{2}$ and we will make 5000 iterations per move. After these 5000, the program prints the board that has been visited the most. That means that this is a node with a very high change of winning, as it has the highest value. After this move has been made, it is Chaos' turn and it repeats. The program does this until an end state has been reached.

After performing MCTS, we have analysed the moves that almost maximize our chances of winning. Out of the 200 simulations we have done, Order wins 199 and Chaos only 1. This still is not sufficient to claim that Order always wins, because MCTS does deterministic moves. It gets pretty close to an always winning strategy. In the continuation of this section we will analyse the strategy based on MCTS and verify that this is a winning strategy.

## 3.3 Analysing strategy on $4 \times 4$ board

From analysis of MCTS, we observed a pattern, that became our first strategy $T_1$ for $\text{OvC}_{4\times 4}(4, 2, \text{O})$. The strategy is as follows for $4 \times 4$ board $B_1 = (b_{ij})_{ij}$ with $(i, j) \in \{1, 2, 3, 4\}^2$.

#### 3.3.1 First strategy

This first strategy has to be followed in the given order. If the first step is not possible, continue to the second and so on.

1. For a board $B_1$ that is empty:

$$T_1(B_1) = ((2, 2), \text{X}).$$

2. For a board $B_1$ with $(i, j) \in \mathcal{M}$ and $s \in S^-$ such that $P((i, j), s) = 3$:

$$T_1(B_1) = ((i, j), s).$$

3. For a board $B_1$ with $(i,j) \in \mathcal{M}$ such that for two elements $x, y \in \{\rightarrow, \uparrow, \nearrow, \searrow\}, x \neq y$ and for $s \in S^-$ we have $P_x((i,j),s) = P_y((i,j),s) = 2$:
$$T_1(B_1) = ((i,j),s).$$

4. For a board $B_1$ we define the set of moves such that we do not intersect other existing lines as
$$C_1 = \{((i,j),s) \in \mathcal{M} \times S^- : \text{for } s' \in S^- \backslash \{s\}, \text{ we have } P((i,j),s') = 0\}.$$
Note that that it may be possible that $P((i,j),s) = 0$. Then we place $s$ in an empty row!
   4.1 Define the set of moves that make the longest feasible line as
   $$C_2 \in \arg \max_{((x,y),z) \in C_1} \{P((x,y),z)\}.$$
   4.2 We then define the strategy:
   $$T_1(B_1) = \arg \min_{((i,j),s) \in C_2} \{i + 4j\}.$$

5. For a board $B_1$ such that $C_1 = \emptyset$, we don't want to intersect the longest line there is, so we define
$$C_3 = \{((i,j),s) \in \mathcal{M} \times S^- : P((i,j),s) < \max_{((x,y),z) \in \mathcal{M} \times S^-} P((x,y),z)\}.$$
   5.1 Define the set of moves that make the longest feasible line as
   $$C_4 \in \arg \max_{((x,y),z) \in C_3} \{P((x,y),z)\}.$$
   5.2 We then define the strategy:
   $$T_1(B_1) = \arg \min_{((i,j),s) \in C_4} \{i + 4j\}.$$

Here we have our first general strategy for every possible state a board can be in. The following step is to show that this strategy is always winning.

## 3.4 Checking strategy

The next step is to verify that the above defined strategy is indeed a winning strategy for Order. We check this by a brute force method.

### 3.4.1 Brute force search

Brute force searching builds a tree of states. We use Depth-First search, see [4] and [5]. We start with board $B_1$, with $b_{22} = \text{X}$. This was the first move $F$. Chaos plays every possible move and Order plays the move as defined by $T_1$. The result gets evaluated by Algorithm 1 below, since we need to check whether every reaction on every possible move is correct and does not lead to a winning result for Chaos. The algorithm we use to check whether our strategy is winning, is as follows:

**Algorithm 1** Brute force verifying

```
procedure DFS($B$)                                   ▷ $B$ is a board
    if $B$ is in order then                           ▷ Winning for Order
        return True
    end if
    if $B$ is chaotic then                            ▷ Winning for Chaos
        return False
    end if
    for $B'$ a child of $B$ do          ▷ Do this for every state reachable from $B$
        $B' \leftarrow$ orders-move($B'$)
        result$\leftarrow DFS(B')$
        if result=True then                ▷ Winning for Order, so continue
            continue
        end if
        if result=False then               ▷ Winning for Chaos, abort
            return False
        end if
    end for
    return True
end procedure
```

### 3.4.2 Result

There are 180690 different ways to reach a stable board and in only 7 of those, Chaos is able to win. Even though the strategy we have analysed is 99.996% accurate, it still is not a winning strategy. This means that we need to adapt our strategy $T_1$.

## 3.5 Adjustment strategy

First we tried to adjust our analysed strategy. However, every change we tried, only resulted in more losing states. Finally, we decided to make case distinctions for 6 boards. This solves our problem and then Order always wins. We verified this by using the same brute force search method as in Algorithm 1.
The red symbol is the move prescribed by $T_1$, and the green symbol is the move that Order has to make to win.

1. $B_1 =$

| | | | |
|---|---|---|---|
| O | | O | O |
| X | X | O | X |
| O | | X | |
| O | | | |

,

2. $B_1 =$

| | | | |
|---|---|---|---|
| | O | | |
| X | X | X | O |
| | | | O |
| | O | | O |

,

3. $B_1 =$

| | | | |
|---|---|---|---|
| | X | | X |
| O | X | X | |
| O | | | |
| | | | |

,

4. $B_1 =$

| | | | |
|---|---|---|---|
| O | O | O | O |
| X | X | | |
| | | | |
| | | | |

,

5. $B_1 =$

| | | | |
|---|---|---|---|
| X | | O | |
| O | X | O | |
| O | | | |
| O | | | X |

,

6. $B_1 =$

| | | | |
|---|---|---|---|
| | X | | |
| O | X | | |
| O | | | O |
| X | X | | O |

.

When we apply these adaptations to our strategy, there is no possible way that Chaos can win when playing our strategy $T_1$. We will explain what happens in these cases.

In cases 1 and 5, we see that instead of not breaking a line, one should make a line of 3. In case 1 this yields $P((2,5), \mathrm{O}) = 3$ and the following happens after doing this move.

| | | | |
|---|---|---|---|
| O | | | O |
| X | X | O | X |
| O | | X | |
| O | | | |

$\rightarrow$

| | | | |
|---|---|---|---|
| O | | | O |
| X | X | O | X |
| O | X | X | |
| O | | | |

$\rightarrow$

| | | | |
|---|---|---|---|
| O | | O | O |
| X | X | O | X |
| O | X | X | |
| O | | | |

$\rightarrow$

| | | | |
|---|---|---|---|
| O | X | O | O |
| X | X | O | X |
| O | X | X | |
| O | | | |

$\rightarrow$

| | | | |
|---|---|---|---|
| O | X | O | O |
| X | X | O | X |
| O | X | X | |
| O | X | | |

.

We see that Order will always win, if he does this move and then continues following strategy $T_1$. In case 5 this happens:

| | | | |
|---|---|---|---|
| X | | O | |
| O | X | | |
| O | | | |
| O | | | X |

$\rightarrow$

| | | | |
|---|---|---|---|
| X | | O | |
| O | X | | |
| O | | O | |
| O | | | X |

$\rightarrow$

| | | | |
|---|---|---|---|
| X | | O | |
| O | X | O | |
| O | | O | |
| O | | | X |

$\rightarrow$

| | | | |
|---|---|---|---|
| X | | O | |
| O | X | O | |
| O | | O | |
| O | | X | X |

$\rightarrow$

| | | | |
|---|---|---|---|
| X | | O | |
| O | X | O | |
| O | O | O | |
| O | | X | X |

$\rightarrow$

| | | | |
|---|---|---|---|
| X | | O | |
| O | X | O | |
| O | O | O | X |
| O | | X | X |

$\rightarrow$

| | | | |
|---|---|---|---|
| X | | O | O |
| O | X | O | |
| O | O | O | X |
| O | | X | X |

.

Again, we see that Order will always win. If Chaos makes another move than the one done here, he automatically loses, because then Order is able to create a row of 4.

In cases 2 and 6, we see that it is better to create an extra line of 2, instead of making a line of 3. This is reasonable, because now, we do not rely on only one line, and we can make better use of the moves Chaos makes.

In cases 3 and 4, $T_1$ prescribes a move too far left. However, by playing on $b_{14}$, Order is able to win, no matter what move Chaos makes afterwards.

After having identified the flaws in our strategy and having fixed them by adding these 6 individual cases, we have obtained our final strategy $T_1$ for Order for $\mathrm{OvC}_{4\times 4}(4, 2, \mathrm{O})$ on the board

$B_1$, namely:

Let $B_{1 \text{ exceptions}} = \{B'_1, B'_2, B'_3, B'_4, B'_5, B'_6\}$, with

$$B'_1 = \begin{array}{|c|c|c|c|}\hline \text{O} & & & \\ \hline \text{X} & \text{X} & \text{O} & \text{X} \\ \hline \text{O} & & \text{X} & \\ \hline \text{O} & & & \\ \hline \end{array}, \qquad B'_4 = \begin{array}{|c|c|c|c|}\hline \text{O} & \text{O} & & \\ \hline \text{X} & \text{X} & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array},$$

$$B'_2 = \begin{array}{|c|c|c|c|}\hline & \text{O} & & \\ \hline \text{X} & \text{X} & \text{X} & \text{O} \\ \hline & & & \\ \hline & & & \text{O} \\ \hline \end{array}, \qquad B'_5 = \begin{array}{|c|c|c|c|}\hline \text{X} & & \text{O} & \\ \hline \text{O} & \text{X} & & \\ \hline \text{O} & & & \\ \hline \text{O} & & & \\ \hline \end{array},$$

$$B'_3 = \begin{array}{|c|c|c|c|}\hline & & & \\ \hline \text{O} & \text{X} & \text{X} & \\ \hline \text{O} & & & \\ \hline & & & \\ \hline \end{array}, \qquad B'_6 = \begin{array}{|c|c|c|c|}\hline & & & \\ \hline \text{O} & \text{X} & & \\ \hline \text{O} & & & \\ \hline \text{X} & \text{X} & & \text{O} \\ \hline \end{array}.$$

We add one note to the strategy in section 3.3.1, namely

1. If $B_1 \in B_{1 \text{ exceptions}}$

$$T_1(B_1) = \begin{cases} ((1,4),\text{O}), & \text{if } B_1 = B'_1; \\ ((4,2),\text{O}), & \text{if } B_2 = B'_2; \\ ((1,4),\text{X}), & \text{if } B_3 = B'_3; \\ ((1,4),\text{O}), & \text{if } B_4 = B'_4; \\ ((4,4),\text{X}), & \text{if } B_5 = B'_5; \\ ((3,4),\text{O}), & \text{if } B_6 = B'_6. \end{cases}$$

2. Follow 3.3.1.

## 3.6 Final strategy on $6 \times 6$ board

Since the final strategy $T_1$ is winning for $\text{OvC}_{4\times4}(4, 2, \text{O})$, we can use Lemma 3.1. It provides us with a winning strategy for $\text{OvC}_{6\times6}(5, 2, \text{O})$, by using our earlier described "mirroring" strategy:. This strategy is as follows:

Let $A_1 = \{2, 3, 4, 5\}^2$ and $A_2 = B \backslash A_1$. Boards $B_1$ and $B|_{A_1}$ are equivalent, hence we can use strategy $T_1$ on $B|_{A_1}$. Then we define our strategy as:

$$T(B) = \begin{cases} T_2(B|_{A_2}) & \text{if } \exists (i,j) \in A_2 : b_{ij} \neq \square \text{ and } b_{m(i,j)} = \square, \\ T_1(B|_{A_1}) & \text{else.} \end{cases}$$

Now it has been proven that Order always wins $\text{OvC}_{6\times6}(5, 2, O)$ by playing strategy $T$. $\square$

# 4 Other rules

Now we have found a strategy for the classic version of the game, it is interesting to study what happens when we change the rules. First we examine what happens for $\mathrm{OvC}_{6\times6}(5,2,\mathrm{C})$, when Chaos makes the first move instead of Order. Secondly, the inovator Stephen Sniderman [2] suggested a change of rules where Order does not win if there is a row of 6. We show that this version is winning for Chaos. The last version we study, is the game where a row of 6 is not winning for Order, and Chaos starts. This version turns out to be winning for Order.

## 4.1 Chaos starts

Note that this version only makes it easier for Order. Since Chaos does the first move, Order can already make a row of 2 in the first move. The strategy is exactly the same as in section 3.6. And again, using Depth-First search, we verified that this is also winning for Order.

## 4.2 No row of 6

Since we have a different suspected winner in this scenario, things get different. We suspect that Chaos wins, if there is a winning player at all. Otherwise, this change of rules would make no sense. Building on this hypothesis, we need a very different approach than in the previous section. Lemma 3.1 does not apply and Monte-Carlo Treesearch does not give us any satisfactory result. Hence we create a winning strategy $T$ by hand.

**Theorem 4.1.** $\mathrm{OvC}_{6\times6}(5,2,\mathrm{O})$ where Order does not win with a row of 6 is winning for Chaos.

*Proof.* We have a board $B \in S^{6\times6}$ with 18 different possible rows, where we can make a row of length 5. This means that for every one of those 18 rows, we need a move to prevent that. In (2), we see that every coordinate in $B_{\text{strategy}} = (b'_{ij})_{ij}$ is paired with another coordinate.

$$B_{\text{strategy}} = \begin{array}{|c|c|c|c|c|c|} \hline 2 & 3 & 7 & 7 & 6 & 1 \\ \hline 4 & 11 & 18 & 13 & 18 & 5 \\ \hline 9 & 17 & 12 & 17 & 14 & 10 \\ \hline 9 & 11 & 16 & 13 & 16 & 10 \\ \hline 6 & 15 & 12 & 15 & 14 & 3 \\ \hline 1 & 5 & 8 & 8 & 4 & 2 \\ \hline \end{array}. \tag{2}$$

**Definition 4.1.** Let $(i,j) \in [6]\times[6]$ and $B_{\text{strategy}} = (b'_{ij})_{ij}$. Then we define the *pairing function* $f\colon [6]\times[6] \to [6]\times[6]$ as follows:

$$f(i,j) = (i',j')\ ,\ \text{if } b'_{i'j'} = b'_{ij}.$$

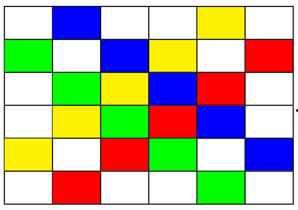
Then define strategy $T$ for every board $B = (b_{ij})_{ij} \in S^{6\times6}$ by

$$T(B) = \begin{cases} (f(i,j), s), & \text{if } (i,j) \in \{(1,1),(1,6),(6,1),(6,6)\} \text{ with } b_{ij} = s \in S^- \\ & \text{and } b_{i'j'} = \square; \\ (f(i,j), s'), & \text{if } (i,j) \in [6]\times[6]\backslash\{(1,1),(1,6),(6,1),(6,6)\} \text{ with} \\ & b_{ij} = s \in S^-,\ b_{i'j'} = \square \text{ and } s' \in S^-\backslash\{s\}. \end{cases}$$

Since Chaos is the second player to play, an odd number of coordinates is filled with a symbol on Chaos' turn. This means that, by the way we have defined our strategy, there is always a coordinate for which one element of the pair is filled and the other is not. That shows, that our strategy is well defined.

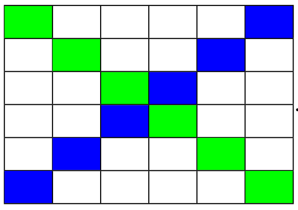
We will take a look at the possible lines in which Order can make a row of five. In the rest of the proof, we explain why Order is not able to make a row of 5 in these lines.

Suppose that Order is able to make a row of 5. In the highlighted rows in 3, there is no possibility that there could be a row of 6, so if there is a row of 5 in one of these rows, Order automatically wins and there is no possibility of extending this line.


. (3)

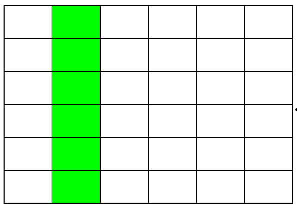
A direct consequence of playing our strategy, is that Order will be the first player to place a symbol on either $b_{5,1}$ or $b_{1,5}$ for the yellow line, $b_{2,6}$ or $b_{6,2}$ for the red line, $b_{2,1}$ or $b_{6,5}$ for the green line and $b_{1,2}$ or $b_{5,6}$ for the blue line. Now suppose (symmetric for the other lines) that $L_{\searrow}((1,2)) = 5$ of symbol $s_L$. That means that Order has done move $((1,2), s_L)$ or $((5,6), s_L)$. After Order has done this move, Chaos can respond by doing move $((5,6), s'_L)$ and $((1,2), s'_L)$ with $s'_L \in S^{-}\backslash\{s_L\}$ respectively. This means that $L_{\searrow}((1,2)) < 5$, which gives a contradiction. There can be no line of 5 here.

We also have the diagonals where we can also make a row of 6, as we see in (4). Suppose that Order makes a row of 5 in one of these lines.


. (4)

Again Order has to place a symbol on either $b_{11}$ or $b_{66}$ for the green line and $b_{61}$ and $b_{16}$ for the blue line first. Chaos responds by placing the same symbol on the other coordinate on the other side. Now suppose that if $L_{\searrow}((1,1)) = 5$, then $b_{11} = s_L$, and $b_{66} = s_L$. But that means that $L_{\searrow}((1,1)) = 6$. This is a contradiction and Order does not win with a row of 6. This means that there will never be a row of 5, such that Order wins on these lines.

Lastly, we have the horizontal and vertical lines. We will analyse one case, but it is symmetric for the other ones. In (5), we see one line


. (5)

If Order wants a row of 5 here, he needs to place symbol $s$ on both $b_{22}$ and $b_{42}$. If he does move $((2,2), s)$ though, Chaos immediately responds with move $((4,2), s')$ with $s' \neq s$, hence Order cannot make a row of 5 here.

All the other horizontal and vertical rows of 5 are prevented by pairs numbered 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17 and 18. Now we have seen that there is no way that Order can make a row of 5, if Chaos plays this strategy, hence $T$ is winning for Chaos. $\square$

## 4.3 Chaos starts, no row of 6

The last variant of Order and Chaos we will discuss is $\mathrm{OvC}_{6\times 6}(5, 2, \mathrm{C})$, where Order does not win with a row of length 6. We figured out a strategy $T$ that is winning for Order.

**Theorem 4.2.** $\mathrm{OvC}_{6\times 6}(5, 2, \mathrm{C})$ where Order does not win with a row of 6 is winning for Order.

We let $T_1$ the same as defined in section 3.5, and $T_2$ the same as defined in the proof of Lemma 3.1. Then our strategy is $T = T_1 \cup T_2$ and this is winning for Order. In the proof we will see why this is winning and well defined.

*Proof.* Since Order does not begin, he only reacts on the move Chaos makes. Note that both $|A_1| = 16$ and $|A_2| = 20$ are even numbers. Order only plays on $A_1$ if Chaos does and Order only plays on $A_2$ if Chaos does. Suppose that $A_1$ is full before $A_2$, so for all $(i, j) \in \{2, 3, 4, 5\}^2$ we have $b_{ij} \neq \square$. Since we have played strategy $T_1$, we know that there is a $(i, j) \in \{2, 3, 4, 5\}^2$ such that $L((i, j)) = 4$. Now look at a extenstion of $L((i, j))$, that is $(a, b) \in A_2$. We have two possibilities for this extension (direct consequence of playing $T_2$), namely

1. $b_{ab} = s_L, b_{m(a,b)} = s'_L \in S^- \backslash \{s_L\}$ or vice versa;
2. $b_{ab} = b_{m(a,b)} = \square$.

In case 1, we know that $L((i, j)) = 5$ and there is no possibility of making this a row of 6, hence Order wins. In case 2, no one has won, so we continue playing. Now it is Chaos' turn and he can only play on $A_2$ and we play our strategy $T_2$. At some point, Chaos would be the first to do move $((a, b), s)$ or $(m(a, b), s)$ for a $s \in S^-$, because $|A_2|$ is even. Still playing along $T_2$, we react by doing move $(m(a, b), s')$ or $((a, b), s')$ respectively with $s' \in S^- \backslash \{s\}$. Now we have that either $b_{ab} = s_L$ or $b_{m(a,b)} = s_L$, hence $L((i, j)) = 5$ and there is no possibility of making this a row of 6.

Now suppose that $A_2$ is full before $A_1$. Now we know for every $(i, j) \in A_2$ that if $b_{ij} = s \in S^-$, we have $b_{m(i,j)} = s' \in S^- \backslash \{s\}$. After $A_2$ is full, it is Chaos' turn and he has to do a move in $A_1$. Order will continue playing $T_1$ and at some point, also $A_1$ is full. Because we have played strategy $T_1$, there must be an $(i, j) \in A_2$ with $L((i, j)) = 4$. Since $A_2$ is full, we know that for an extension $(a, b) \in A_2$ of row $L((i, j))$ we have $b_{ab} = s_L$. Because we have played strategy $T_2$, we know that $b_{m(a,b)} = s'_L \in S^{-} \backslash \{s_L\}$ or vice versa, which gives us that $L((i, j)) = 5$ and there is no way of making this a row of 6.

We have covered both possibilities and we have seen that strategy $T = T_1 \cup T_2$ does not need a transition function $f$, and is well defined. All this combined, tells us that $T = T_1 \cup T_2$ is a winning strategy for Order. $\square$

## 5 Summary and further research

In this thesis, we have analysed an Order and Chaos game to find and define a winning strategy for Order. In section 2 we have defined and introduced all definitions and notations needed to analyse this game mathematically.

By doing research we knew that Order would win. In section 3 we used some tools to determine a winning strategy. One of the tools was Lemma 3.1. This gave us a strategy on the border of the board and ensured that we only had to find a strategy for $\mathrm{OvC}_{4\times 4}(4, 2, \mathrm{O})$. Another tool we used, was Monte-Carlo Treesearch for identifying a winning strategy. After studying the outcome, we identified certain patterns, yielding a first candidate for a winning strategy. However, we had to verify that it is winning. After some adjustments, Brute-Force search confirmed that we had found a winning strategy $T_1$ for Order for $\mathrm{OvC}_{4\times 4}(4, 2, \mathrm{O})$. Using Lemma 3.1, we had a well defined winning strategy $T$ for Order for $\mathrm{OvC}_{6\times 6}(5, 2, \mathrm{O})$.

Secondly, we examined the behaviour of the game when we changed the rules. The first change of rules was that Chaos begins instead of Order. We came to the conclusion that this made it easier for Order. Following the same strategy $T$ as the normal version of the game was sufficient to always get a winning result. The second change of rules was that Order would not win by creating a row of 6. This change of rules resulted in a winning game for Chaos. We defined a strategy and proved that this strategy is always winning for Chaos. The last change was a combination of both, namely Chaos would start and Order does not win by creating a row of 5. For this version it was also sufficient to follow strategy $T$ as normal, but without transition function $f$.

As regards further research on this game, it would be interesting to see what happens with different shapes and sizes of the board. We only considered a $6 \times 6$ board; Strategies would probably change a lot when the board has a different size. Daniël van Gent has done some research on $n \times n$ boards in general. He showed that $\mathrm{OvC}_{n\times n}(n-1, 2, \mathrm{O})$ is winning for Chaos for $n \geq 7$. Other interesting things to look further into are for example adding more symbols or adding more players. This would make the game a lot more complex and strategies would probably change drasticaly. Lastly, it would be nice to identify a more elegant winning strategy than the strategy we have found. It would be perfect if there is a strategy without any exceptions.

## References

[1] Austrian game museum. URL http://www.ludorium.at/prog/Anzeige.asp?sysname=english&offset=20&sn=&vl=GAMES%20MAGAZINE&at=&zg=&ge=&v1=&pa=&al=&sp=&kt=&cd=&yr=&sort=1&show=&cb=.

[2] Order and Chaos. URL https://en.wikipedia.org/wiki/Order_and_Chaos.

[3] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck. Monte-Carlo tree search: A new framework for game AI. In *AIIDE*, 2008.

[4] P. Meseguer. Interleaved depth-first search. In *IJCAI*, volume 97, pages 1382–1387. Citeseer, 1997.

[5] J. Pearl and R. E. Korf. Search techniques. *Annual Review of Computer Science*, 2(1): 451–467, 1987.

[6] D. W. Peter Cowling, Ed Powley. Monte Carlo tree search Python code, 2012. URL http://mcts.ai/code/python.html.

[7] A. N. Siegel. *Combinatorial game theory*, volume 146. American Mathematical Soc., 2013.

[8] B. Turner. Solved: Order wins, 2016. URL https://boardgamegeek.com/thread/1579397/solved-order-wins.

[9] M. H. Winands. Monte-Carlo tree search. *Encyclopedia of Computer Graphics and Games*, pages 1–6, 2015.