Department of Computer Science
Faculty of Mathematics, Physics and Informatics
Comenius University

Composite mathematical games

Bachelor Thesis

Rastislav Lenhardt

Advisor:
Doc. RNDr. Rastislav Kráľovič, PhD.

Bratislava, 2007

Abstract

Decomposition of mathematical games to smaller parts is essential because of the large number of possible positions. We have studied the degenerated variation of the famous Dots and Boxes game where the winner is a player who first creates the first box. This variation is strongly related to the original game and can often help us to find a strategy for the original game. We have found and proved which player has a winning strategy if we start from the initial empty board. For arbitrary positions we found out that the game is being decomposed to the smaller subgames where the winner is a player who wins the first of these subgames.

We have introduced w-function that assigns w-numbers to the positions of subgames and by combining these values we can find the w-number and thus who has a winning strategy in the composite games of this type. These results allow us to find solutions to these games much faster. If n is the size of the subgames and k number of subgames then the complexity is only $O(kn)$ in comparison to $O(n^k)$ obtained by the standard algorithm. We used this general concept to solve the degenerated Dots and Boxes game. However, it can be used in any other games decomposing this way. We have also succeeded to find the way how to play this combination of games with the misére play rule.

The first part contains known ways of adding impartial games with proofs and many examples to help orient in this area. We have also created a combinatorial game API to support and improve conditions in researching the combinatorial games. Almost all these games including the degenerated Dots and Boxes game and algorithms for composite games are implemented in this API.

Keywords: Dots and Boxes, composite games, w-numbers

Contents

1 Introduction

The combinatorial game theory is a quite new branch of mathematics. However, people have been playing games for ages. It is a natural human behaviour and our basic instinct to compete, have fun and dominate the peers. There are applications and connections to complexity, logic, graph theory, networks, error correcting codes, distributed systems, surreal numbers and analysis and design of mathematical and commercial games.

Unofficially we can divide games into two categories. In the first there are games people play. In the second there are games mathematicians play. The first category contains challenging games that people will purchase and play. It is really hard to invent such games. They cannot have a simple winning strategy. But at the same time the thing that makes them very amusing and popular is that even an inexperienced player has a certain feeling from the position. For example he can say if the position in a chess game is strong or weak for the current player.

As Judea Pearl wrote mathematical games offer a perfect laboratory for studying complex problem solving methodologies. With a few rules, one can create complex situations that need a huge amount of insight. Difficulty of combinatorial games can be showed by comparing it to cryptography. In a cryptography we want to know if $\exists$ a cryptosystem that $\forall$ attacks on it will not break him. In games we have a decision problem. We want to know if $\exists$ a move for the first player that for $\forall$ opponent's moves $\exists$ a move for the first palyer that $\forall$ ... such that the first player will win? What makes games even harder is that we are not playing against a passive opponent like when we are solving NP-hard problems like finding Hamiltonian cycle or Travelling salesman problem. Our opponent in a game is taking all possible actions against us. So we often need to examine the whole decision subtree, not simply find a path.

But not everything is lost. Fortunately, we can do better and profit from the fact that we know better structures of some games. The essential fact is that games are often composed from more disjunctive (and simpler) parts. We can precompute some information for parts and then add it together to find out the result for the whole game. It is one of the goals of this thesis to review existing ways of adding impartial mathematical games to help other students at undergraduate level to orient in this topic. So in the third section algorithms can be found for how to play different types of composite games with proofs and many examples.

The next part of the thesis is more research oriented. Mr Elwyn Berlekamp, a professor at UC Berkeley, co-author of bibles of combinatorial games [Ww01] and author of [Be00], has proposed to me a problem in the Dots and Boxes game. The Dots and Boxes game is a game people play, but it is examined a lot by mathematicians, as well. We have been researching how to play degenerate variation of this game in which a player who creates the first box wins. Knowledge of how to play this degenerate variation can help a lot to play the original game. In the fourth section there are answers for how to play this degenerate game from the initial empty board. While looking for a strategy for any arbitrary positions we have realised that the game is being decomposed to smaller subgames, but with a rule that a player who first wins one of these subgames wins the whole game. This was different

from all known ways presented in the third section. So from the concrete game we get to the new general problem, but we managed to solve it. We introduced a new w-function that assigns w-number to every position. Then we can combine these w-numbers to find out if the first or the second player has the winning strategy. But these numbers represent also the whole sum of parts. If we want to add another part to this compound we just need to combine its w-number with the w-number of the sum obtained before. This result allows us to find results much faster. We can decrease complexity from $O(n^k)$ to $O(kn)$ where n is a number of positions of subgame and k number of subgames. It is very important, because n is often very high. The solution to this compound for any games is presented in Section 5. Later it is implemented and used for solving the degenerated variation of the Dots and Boxes game. However, this general concept can be used to find solutions to any other games that decompose this way. Introduced w-numbers helped to a quite unexpected discovery. We found the strategy how to play this combination of games with the miseére play rule. It was unexpected, because there is no known strategy for any other types of composite games played with the misére play rule.

Almost all algorithms are implemented and all concrete results gained by using newly developed Combinatorial game API. It is an open and carefully designed tool created to support and improve conditions in researching the combinatorial games. It would be almost impossible to get and check any results without having such a tool[1]. The whole API with source codes is an attachment to this thesis.

I would like to thank my thesis advisor Rastislav Kráľovič for his time, valuable feedback and a lot of advice. I would also like to thank to my family and friends for all their love and support.

Rastislav Lenhardt

[1]You are as strong as your tools allow you to be.

2 Mathematical games

Mathematical games are two-person games with perfect information and no chance moves, and with a win-or-lose outcome. Players are usually alternatively moving until they reach a terminal position. The terminal position is position from which no possible moves are possible. After that one player is declared the winner and the other the loser.

2.1 Definitions

Definition 2.1. *The game G is a mathematical (combinatorial) game if it satisfies the following properties:*

1. *There are two players.*
2. *There is a finite set X of possible positions of the game.*
3. *The rules of the game specify for both players and each position which moves to other positions are legal moves.*
4. *The players alternate moving.*
5. *The game G ends when there is no possible move from the position for the player whose turn it is.*
6. *The game ends in a finite number of moves no matter how it is played.*

Definition 2.2. *Game G is **impartial** if for each position $x \in X$ there are the same options (possible moves) for both players. Otherwise game G is called **partizan**.*

Definition 2.3. *Under the **normal play rule**, the last player to move wins (the player who is unable to move loses).*

Definition 2.4. *Under the **misére play rule**, the last player to move loses.*

Corollary 2.1. *Impartial game G is determined by a set of positions X, including an initial position $x_0 \in X$, moving function $f : X \to 2^X$ that returns positions to which a player can move from every position $x \in X$ and winning condition.*

In this thesis (if not stated explicitly otherwise) we will mean by game G an impartial game played under normal play rule. The player who starts the game will be called First and the other one will be called Second.

Example 2.1. Cards are not a mathematical game because players do not have perfect information about the position.

Example 2.2. Tic-Tac-Toe is a mathematical game, but it is partizan (not impartial), because one player can make only X‘s and other only O‘s.

2.2 Basic concept

Definition 2.5. *Every position in game G is either a Winning position (often called W-position) or Losing position (L-position). Winning positions are all positions from which player whose turn it is can force a win against his opponent.*

Positions of every game have following properties:

- All terminal positions are L-positions.
- If a player is able to move from position x to L-position then x is W-position.
- If a player is able to move from position x only to W-positions then x is L-position.

These properties lead to straightforward recursive implementation of algorithm. While we have only finitely many positions we can step by step label all of them either W-position or L-position.

Note 2.1. While a player can lose only if he cannot move, he can secure a win by finding a strategy that will give him in all stages of the game at least one possible move.

Rules 2.1 (Limited Nim). *Limited Nim is a game played with n coins and defined finite substraction set $S \subset \mathbb{Z}^+$. The player whose turn it is takes away k coins, where $k \in S$. If he is unable to move he loses.*

Example 2.3. Limited Nim Game with 11 coins and substraction set $S = \{1, 3, 4\}$:

n	0	1	2	3	4	5	6	7	8	9	10	11
position	L	W	L	W	W	W	W	L	W	L	W	W

2.3 The Game of Nim

The most famous mathematical game is probably the Game of Nim.

Rules 2.2 (The Game of Nim). *There are n piles of coins. When it is a player's turn he chooses one pile and takes away at least one coin from it. If someone is unable to move he loses (so the one who removes the last coin is the winner).*

If there is only one pile, the solution is simply to remove all coins. If there are two piles we know that the terminal position is $(0, 0)$. We can use the power of symmetry to obtain that only (x, x) are losing positions. If one player moves from this position, the second player can return to this position by taking the same number of coins from another pile. Therefore the second player has always turn so he will win.

Finding patterns for three and more piles is not so simple. The idea behind the solution is the following equivalent game.

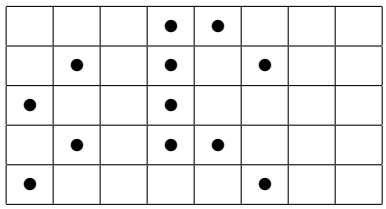

Figure 1: Example of L-position in Chessboard Nim (all columns with even number of coins)

Rules 2.3 (Chessboard Nim). *Consider a chessboard with n rows. At the beginning there are coins on some squares of the chessboard (at most one coin on one square). When it is a player's turn he must choose one of these coins to remove and in his move he can also remove or add any coins in that row that are left of the removed one. If someone is unable to move he loses.*

Finding a winning strategy of the game is often about finding an invariant which divides positions to winning and losing ones.

Theorem 2.2. *In the game Chessboard Nim losing positions are all with an even number of coins in every column of the chessboard.*

Proof. We can prove that by seeing that from every such position it is possible to move only to the positions with at least one odd column (it is the column in which player chooses one coin that must be removed). On the other hand if there is at least one odd column, player can correct it by removing the coin from the most right odd column and then he can change parity in all other columns to the left as he wants. The terminal position (empty chessboard) has all even columns, so a player who is moving to all-even-columns positions will always be able to move. □

Remark 2.1. Every row of the chessboard is equivalent to the one pile of the Game of Nim. The size of the pile is encoded in a binary system by the position of coins on chessboards (the most left column is 2^0, then 2^1, 2^2, ...).

Corollary 2.3. *Position of the Game of Nim with piles p_1, p_2, ... p_n is losing if and only if $p_1 \oplus p_2 \oplus \ldots \oplus p_n = 0$, where $\oplus$ is operation xor.*

Example 2.4. Position $(1, 2, 3)$ is losing because $1 \oplus 2 \oplus 3 = (1)_2 \oplus (10)_2 \oplus (11)_2 = 0$.

Example 2.5. Position $(7, 4, 1)$ is winning because $7 \oplus 4 \oplus 1 = (111)_2 \oplus (100)_2 \oplus (01)_2 = (10)_2 = 2 \neq 0$

There are many games that are directly based on the knowledge of the Game of Nim. To find out more about them we recommend to look for Turning Turtles, Staircase Nim, Nimble or Northcott's Game.

3 Known ways of adding impartial games

In this chapter we will get through many ways of adding mathematical games together. It is an essential part of studying mathematical games, because it allows us to solve a much bigger number of games.

By adding games G_1 and G_2 (with position sets X_1, X_2) together we get a new game G_3. We can look at this new game G_3 as a standard game and we can examine it in the same way as G_1 and G_2. But it is not a very good idea. The number of positions $|X_3|$ in game G_3 is mostly about $|X_1||X_2|$. And in a more general case if we have k subgames with approximately n positions each, then the number of positions in this composite game is about n^k. It is increasing exponentially and while the standard algorithm for determining W-positions and L-positions needs to search almost all possible positions, it will be almost impossible to find winning strategies by using it.

Fortunately, we can do better and profit from the fact that we know better a structure of composite game. Usually we will precompute some values for every subgame and then combine them in order to find out if the position is winning or losing.

3.1 Ordinary sum of games (xor)

In this subsection we will combine games G_1, G_2, ..., G_n in a way that the player whose turn it is must choose one of the games and make a move in it. A player who is not able to move in all the games loses. This combination is called ordinary sum and we can write $G = G_1 \oplus G_2 \oplus \ldots \oplus G_n$.

First we will introduce Sprague-Grundy function g that will assign values to all positions of all subgames.

Definition 3.1. *The Sprague-Grundy function of a game G (X, f) is a function $g : X \to \mathbb{Z}_0^+$ such that*

$$g(x) = min(n \geq 0 : n \neq g(y) \text{ for } y \in f(x))$$

In words, the Sprague-Grundy function is recursively defined and it gives to the position the smallest non-negative integer value that is not found among all following possible positions to which player can move. After assigning grundy values to all positions we can see that x is L-position if and only if $g(x) = 0$. It follows from these position's properties:

1. All terminal positions are L-positions and all have $g(x) = 0$.
2. If at position x we have $g(x) > 0$ then there is at least one following position y with $g(y) = 0$.
3. If at position x we have $g(x) = 0$ then it is not possible to move to position y with $g(y) = 0$.

Example 3.1. Grundy numbers for Limited Nim Game with 11 coins and substraction set $S = \{1, 3, 4\}$:

n	0	1	2	3	4	5	6	7	8	9	10	11
g(n)	0	1	0	1	2	3	2	0	1	0	1	2
position	L	W	L	W	W	W	W	L	W	L	W	W

Grundy numbers give us much more information about the game than whether a position is winning or losing. They allow us to find quickly the ordinary sum of more games. In this sum every position x in a game is equivalent to one pile in the Game of Nim of size k, where $g(x) = k$. We know how to play the Game of Nim so we will be able to use it to play any ordinary sum of any games.

Theorem 3.1. *Given game $G(X, f)$ that is the ordinary sum of n games $G_1(X_1, f_1)$, $G_2(X_2, f_2)$, $\ldots G_n(X_n, f_n)$. Position $x = [x_1, x_2, \ldots x_n]$ is losing if and only if $g(x_1) \oplus g(x_2) \oplus \ldots \oplus g(x_n) = 0$ where $x \in X$, $x_i \in X_i$ and $\oplus$ is operation xor.*

Proof. The combination of n piles in the Game of Nim is the same as rules for adding n games in the ordinary way. Therefore it is enough to prove that position $x_i \in X_i$ with $g(x_i) = k$ is equivalent to the pile in the Game of Nim of size k.

In the Game of Nim if there is a pile of size $k > 0$ (if $k = 0$ it is not possible to move) then a player is able to move to any smaller size $u < k$. In game G_i we are now in position x_i with $g(x) = k$ and while $u < k$ there must be following position y with $g(y) = u$ (otherwise $g(x_i)$ would be at most u). So the equivalent move to the smaller pile in the Game of Nim of size u is to move in G_i to position y.

In Game G_i at position x_i with $g(x_i) = k$ we will consider only moves to following positions y where $g(y) < k$. We can do that because there is no following position of x with $g(y) = k$ (otherwise $g(x) \neq k$) and if player moves to position y with $g(y) > k$ his opponent can move back to another position of size k. If player moves to position y with $g(y) < k$ then we can also decrease the size of the corresponding pile in the Game of Nim. □

Note 3.1. Operation $\oplus$ on games is associative, commutative and $g(G_1 \oplus G_2) = g(G_1) \oplus g(G_2)$.

3.2 Union of games (or)

In this subsection we will combine games G_1, G_2, ..., G_n in a way that the player whose turn it is must choose at least one of these games and make one move in every chosen one. A player who is not able to move loses. This combination is called union of games and we can write it as $G = G_1 \vee G_2 \vee \ldots \vee G_n$.

Rules 3.1 (Game of Queens). *Game of Queens is played on an $n \times n$ chessboard*[2] *. At the beginning there are some queens at some squares on the chessboard (there can be more*

[2]Everything in this chapter works of course for all impartial games. We are using board ones only because it is much more simple to show numbers at positions on a chessboard than having a complicated game tree of an arbitrary game.

queens on one square). When it is player's turn he must choose at least one queen and move with her in west, north or north-west direction.

First we can notice that it will be the same situation if every queen has her own chessboard. After that it is obvious that Game of Queens is a union of games with only one queen on each chessboard. Following theorem shows us solution.

Theorem 3.2. *Given game $G(X, f)$ that is union of n games $G_1(X_1, f_1)$, $G_2(X_2, f_2)$, $\ldots G_n(X_n, f_n)$. Position $x = [x_1, x_2, \ldots x_n]$ is losing if and only if $\forall i$ $g(x_i) = 0$ where $x \in X$, $x_i \in X_i$.*

Proof. Terminal position is L-position. At position y which has $g(y) = 0$ we can move only to positions with non-zero grundy numbers. Therefore if $\forall i$ $g(x_i) = 0$ and while a player must move at least in one subgame he will move to a new position where at least one of her subgames will have $g(x_k) \neq 0$.

On the other hand if there is at least one subgame with a position in which the grundy number is non-zero, the player can choose all subgames for which $g(x_i) \neq 0$ and move in them to $g(x_i) = 0$. □

0	1	2	3	4	5	6	7
1	2	**0**	4	5	3	7	8
2	**0**	1	5	3	4	8	6
3	4	5	6	2	**0**	1	9
4	5	3	2	7	6	9	**0**
5	3	4	**0**	6	8	10	1
6	7	8	1	9	10	3	4
7	8	6	9	**0**	1	4	5

Figure 2: Grundy numbers for 8×8 Game of Queens

To solve instances of Game of Queens, first we need to precompute grundy numbers as in Figure 2. If all queens are at squares with grundy number 0 then the player whose turn it is will lose otherwise he can win by moving all of them to 0 positions.

3.3 Selective compound (almost or)

In this subsection we will combine games G_1, G_2, ..., G_n in a slightly different way from one used in union. The player whose turn it is must choose at least one of these subgames, but he cannot choose all of them and then make one move in every chosen one. A player who is not able to move loses. This combination is called selective compound of games and we can write it as $G = G_1 \uplus G_2 \uplus \ldots \uplus G_n$.

Example 3.2. An example of such a game is a slightly changed variation of Game of Queens, where a player cannot choose all the queens to move.

Theorem 3.3. *Given game $G(X, f)$ that is selective compound of n games $G_1(X_1, f_1)$, $G_2(X_2, f_2)$, $\ldots G_n(X_n, f_n)$. Position $x = [x_1, x_2, \ldots x_n]$ is losing if and only if $g(x_1) = g(x_2) = \ldots = g(x_n)$ where $x \in X$, $x_i \in X_i$.*

Proof. Terminal position is L-position, because $\forall i\ g(x_i) = 0$. In L-position $\forall i\ g(x_i) = u$ and while a player cannot move in all subgames, after move $\exists x_k\ g(x_k) = u$. But while a player has to move at least in one subgame there will be at least one position with $g(x_l) \neq u$.

Let $m = min(g(x_i))$ through all positions x_i. If player is in W-position then all subgames cannot have the same grundy number. Therefore a player can always make a move by decreasing $g(x_i)$ to m in all positions x_i with $g(x_i) > m$. □

Theorem 3.3 shows us how to play the game from Example 3.2. If all queens are at squares with the same grundy numbers (we can use grundy numbers from Figure 2) it is L-position. Otherwise a player can move all of them to the same grundy number (which is the minimum among grundy numbers of current positions in subgames).

3.4 Conjunctive compound (and)

In this subsection we will combine games G_1, G_2, ..., G_n in a way that the player whose turn it is must make a move in every subgame. A player who is not able to move loses. This combination is called conjunctive compound of games and we can write it as $G = G_1 \wedge G_2 \wedge \ldots \wedge G_n$.

We will start with an example game used also in the 9th chapter of [Ww01]. The name of the game is All the King‘s horses and it is very similar to Game of Queens.

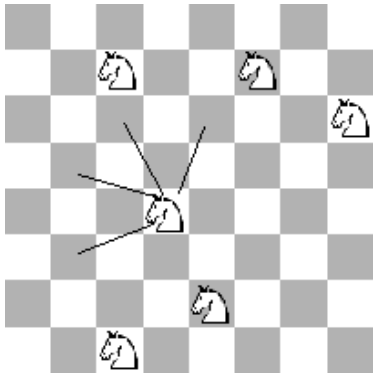

Figure 3: Possible moves (jumps) for horse in All the King‘s horses game.

Rules 3.2 (All the King‘s horses). *Game is played on an $n \times n$ or quarter-infinite chessboard. At the beginning there are some horses at some squares on the chessboard. There can be more than one horse at one square. When it is a player‘s turn he must move with*

all the horses. A player who cannot move loses. Horses move (jump) like knights in chess but at most only to four positions [3] *with a lower sum of row and column coordinates.*

We can look at this game as it is a conjunctive compound of n subgames played on n chessboards with exactly one horse on every chessboard. In turn a player must move in every subgame. Assume for a while that we have only one chessboard with only one horse. We can use a standard algorithm for dividing positions to W-positions and L-positions. The result of this algorithm for $n \times n$ board is shown in Figure 4.

L	L	W	W	L	L	W	W
L	L	W	W	L	L	W	W
W	W	W	W	W	W	W	W
W	W	W	W	W	W	W	W
L	L	W	W	L	L	W	W
L	L	W	W	L	L	W	W
W	W	W	W	W	W	W	W
W	W	W	W	W	W	W	L

Figure 4: All the King‘s horses on 8×8 chessboard with only one horse

Note 3.2. When it is a player‘s turn he must play in all subgames, therefore if he can force a win in all of them he will win this game, because he will always be able to move.

Except the previous simple case it can happen that in some subgames there is in winning position First and in the others Second player. Winning subgames are good for both players, because they cannot lose because of them. Therefore both players will win all their subgames they can. Only losing subgames are dangerous for them and while rules say "Player not able to move loses", the winning strategy is for both of them to win quickly and lose as slowly as possible.

Definition 3.2. *Remoteness function r tells us how many moves the game will last if a player who can force a win will try to win as soon as possible and the losing player will try to lose as slowly as possible.*

Remark 3.1. L-positions have even remoteness and W-positions have odd remoteness.

Theorem 3.4. *Remoteness value of position x:*

1. $r(x) = 0$ *if x is terminal position.*
2. $r(x) = 1+$ *least even number* $r(k)$, $k \in f(x)$ *if such exists.*
3. $r(x) = 1+$ *greatest odd number* $r(k)$, $k \in f(x)$ *if there is no even number* $r(k)$.

[3] as it is shown in Figure 3

Proof. 1. If position x is terminal position it is obvious.

2. If we have the option to move to a position with even remoteness (losing position) we are in a winning position. We want to win as soon as possible so we will choose the losing position $k \in f(x)$ with smallest remoteness and move to it.

3. We cannot move to an even remote (losing) position, therefore we are in a losing position. We want to play as long as possible so we will choose the position $k \in f(x)$ with the highest remoteness and move to it.

□

0	0	1	1	2	2	3	3
0	0	1	1	2	2	3	3
1	1	1	1	3	3	3	3
1	1	1	3	3	3	3	5
2	2	3	3	4	4	5	5
2	2	3	3	4	4	5	5
3	3	3	3	5	5	5	5
3	3	3	5	5	5	5	6

Figure 5: Remoteness values for All the King‘s horses on 8×8 chessboard

Theorem 3.5. *Given game $G(X, f)$ that is conjunctive compound of n games $G_1(X_1, f_1)$, $G_2(X_2, f_2)$, $\ldots G_n(X_n, f_n)$. Position $x = [x_1, x_2, \ldots x_n]$ is losing if and only if $min(r(x_1)$, $r(x_2)$, $\ldots$ $r(x_n))$ is even, where $x \in X$, $x_i \in X_i$.*

Proof. From the definition of conjunctive compound we have that the first finished game determines the winner. First finished game will be the game with the least remoteness, so if this remoteness is even First will lose this game and therefore he will lose conjunctive compound, as well. □

Corollary 3.6. *Remoteness value of position $x = [x_1, x_2, \ldots x_n]$, $r(x) = min(r(x_1)$, $r(x_2)$, $\ldots$ $r(x_n))$ where position x is position of conjunctive compound of positions x_1, x_2, $\ldots$ x_n.*

Note 3.3. Operation $\wedge$ on games is associative, commutative and $r(G_1 \wedge G_2) = min(r(G_1)$, $r(G_2))$.

3.5 Continued conjunctive compound (also)

In this subsection we will combine games G_1, G_2, $\ldots$, G_n in a way that the player whose turn it is must make a move in every subgame he can and the game ends and a player loses only if he cannot move anywhere. This combination is called continued conjunctive compound of games and we can write it as $G = G_1 \triangle G_2 \triangle \ldots \triangle G_n$.

Example 3.3. Consider All the King‘s horses game with slightly changed rules where a player must move only with the horses he can. And he loses only if he cannot move with all.

The winning strategy is a parody to one used in conjunctive compound of games. It is the same for players that only subgames they will lose are dangerous. But in this case it does not matter if player loses subgame if there are other not finished subgames. Therefore a player will try to finish losing subgames as soon as possible and play winning subgames as long as possible. This time the winner is determined by the last finished subgame.

Definition 3.3. *Suspense function s tells us how many moves a game will last if player who cannot force win will try to lose as soon as possible and winning player will try to play as long as possible.*

Remark 3.2. L-positions have even suspense number and W-positions have odd suspense number.

Theorem 3.7. *Suspense number of position x:*

1. $s(x) = 0$ *if x is terminal position.*
2. $s(x) = 1+$ *greatest even number $s(k)$, $k \in f(x)$ if such exists.*
3. $s(x) = 1+$ *least odd number $s(k)$, $k \in f(x)$ if there is no even number $s(k)$.*

Proof. Almost the same as proof of Theorem 3.4. □

0	0	1	1	2	2	3	3
0	0	1	1	2	2	3	3
1	1	1	3	3	3	3	3
1	1	3	3	3	3	5	5
2	2	3	3	2	4	5	5
2	2	3	3	4	4	5	5
3	3	3	5	5	5	5	5
3	3	3	5	5	5	5	6

Figure 6: Suspense numbers for changed All the King‘s horses on 8×8 chessboard from Example 3.3

Theorem 3.8. *Given game $G(X, f)$ that is continued conjunctive compound of n games $G_1(X_1, f_1)$, $G_2(X_2, f_2)$, $\dots G_n(X_n, f_n)$. Position $x = [x_1, x_2, \dots x_n]$ is losing if and only if $max(s(x_1), s(x_2), \dots s(x_n))$ is even, where $x \in X$, $x_i \in X_i$.*

Proof. Almost the same as proof of Theorem 3.5. □

Corollary 3.9. *Suspense number of position* $x = [x_1, x_2, \ldots x_n]$, $s(x) = max(s(x_1), s(x_2), \ldots s(x_n))$ *where position* x *is position of continued conjunctive compound of positions* x_1, x_2, $\ldots$ x_n.

Note 3.4. Operation $\triangle$ on games is associative, commutative and $s(G_1 \triangle G_2) = max(s(G_1), s(G_2))$.

4 The Dots and Boxes game

The Dots and Boxes is a pen and pencil game for two players. It starts with an empty grid of dots. Players alternatively move by adding a vertical or horizontal line between two unjoined adjacent dots. If a player completes the fourth side of the box he earns one point and takes one more turn. The game ends when there is no additional move left and the player with more points is the winner.

A lot of information about the game and strategies can be found in [Be00]. Figure 7 shows example of simple game on 3×3 board where First player won 3:1.

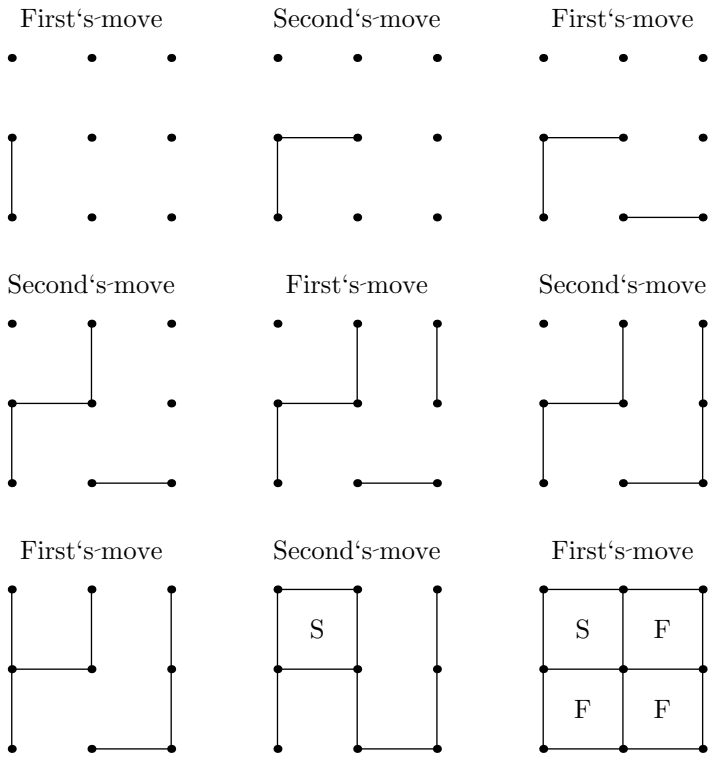

Figure 7: Example of 3×3 Dots and Boxes game

4.1 Degenerate variation

In this chapter we will study a degenerate variation of the Dots and Boxes game. It is similar to the original one with the rule that it ends after someone completes the first box and the player who made that box wins.

The reason why we study a degenerate variation is that it is closely related to the original Dots and Boxes. Dots and Boxes is a game that can be played at more levels and players who know how to play it at a higher level than their opponents can usually win it easily. At these higher levels it is mostly about taking control of the game. Usually if there are sufficiently many chains of length at least 3 and loops of length at least 4, it often happens that a player cannot hope to win unless he captures the next box. Although that is not always sufficient, its necessity forces a player to make moves consistent with winning moves in the degenerate game. So it can be a very strong constraint and this degenerate game can be very helpful in finding a winning move in the original version (or proving that no such move exists).

4.2 Solution for starting positions of degenerate variation

Definition 4.1. *Degenerate $m \times n$ Dots and Boxes game is played on the board that consists of m rows and n columns of dots where $m, n \geq 2$.*

First we can consider $2 \times n$ degenerate games. All vertical moves except the most left and the most right divides the playing board into two parts. An idea that could come to an experienced readers mind could be to use symmetry, because it is one of the strongest weapons (although sometimes hard or impossible to use) that is used in combinatorial game theory.

Now imagine for a while that we are in a little bit different situation. First player made a vertical line that has divided the game into two equal parts (it is possible for odd n). Now we have two independent games. If we use ordinary sum (that player who can not move loses) for these two games, it is a winning situation for First. For him it is enough to wait for the opponent‘s move and then just copy it to another game. So First will always have the possibility to move so he will win such a game.

Now go back to $2 \times n$ degenerate game. It is only a little bit more difficult for First to win if n is odd. Similarly to the previous case he will start the game by joining two dots in the center. So the board is divided into two equal parts (subgames). If someone completes the box in either left or right part he will win this game. After splitting the game into two parts it is Second player‘s turn. The best response to every Second‘s move is to

- complete the box (make a winning move) if possible
- copy Second‘s move into another subgame (make a symmetric move) otherwise.

Second player cannot win, because when it is his turn left and right subgames are the same. So if he had the possibility to complete a box in one of the subgames then First would have the same possibility in his last move in at least one subgame.

We will generalize this symmetric concept for even n and also for $m > 2$. It is quite clear that this works for games which are divided by a set of lines to two equal subgames. As we will see from the following theorem it will work also for games without this delimiting set of lines.

Theorem 4.1. *If $m + n$ is even Second player can win all degenerate $m \times n$ Dots and Boxes games.*

Proof. If $m + n$ is even there are two possibilities. Both m and n are odd or both m and n are even. In the first case the center of symmetry of $m \times n$ board S is in the middle of the center box and in the second case it is in the central dot. In both cases every line a has its own twin line a' which is its image in the symmetry by S. We claim that if Second wants to win it is enough for him to

- complete the box (make a winning move) if possible
- make a symmetric move to First's move (So if First added a line a Second will add its twin line a').

To prove the theorem we will prove that after Second's move either game has finished (so Second won) or every box has at most two lines and the board is symmetric. At the beginning this statement is true. First can add line a in the way that either one box will have three lines or not. In the first case Second player will complete this box and win the game. In the second case Second player will add twin line a'. There are two possibilities again. If a and a' are not part of the same box then while the board is symmetric and after adding a there was no box with three lines there is no such box also now. If a and a' are part of the same box it means that S is in the center of this box. After Second's move the board is symmetric so this center box has either two or four lines. If four then Second won otherwise this box has only two lines. □

Theorem 4.2. *If $m + n$ is odd First player can win all degenerate $m \times n$ Dots and Boxes games.*

Proof. If $m + n$ is odd then either m is even and n is odd or m is odd and n is even. In both cases the center of symmetry of $m \times n$ board S is in the center of one line l. Every other line a has its own twin line a' which is its image in symmetry by S. First player can force a win by starting the game by adding line l. After that we have the situation in which the board is symmetric, every box has at most two lines and for every free line a there is its twin line a' that is free, too. So we are in the same situation as in the second part of proof of Theorem 4.1. □

5 w-numbers

In the previous section we explained solution to empty boards of degenerated Dots and Boxes game. During looking for strategies for any other positions in this game we noticed that it often happens that the whole game is divided into subgames like in Figure 8.

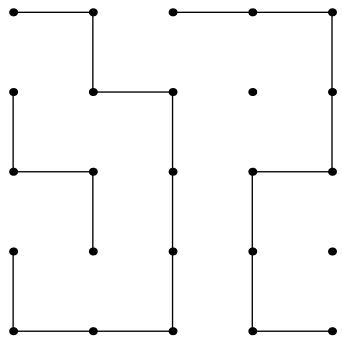

Figure 8: Position in the degenerated Dots and Boxes game

In Figure 8 we can see that the board is divided into three independent parts bordered by lines or edges of the board. When it is a player's turn he can choose one of these subgames and add a line into it. If he creates a box he will win not only that particular subgame but also the whole game which is a combination of these three parts.

After that we decided to look more deeply into how to play a combination of games where the rule is that if someone wins first game he will win the whole combination of games. Because of that we started to call this type of compound game a WTIA game, where WTIA stands for The winner takes it all[4]. In this section we will provide our results.

Definition 5.1. *WTIA game G is a game that is a combination of n subgames $G_1, G_2, \ldots G_n$. If it is a player's move he chooses one of the subgames and makes move in it. A player who loses one of the subgames G_i loses the whole game G.*

First observation how it does work was done in the previous section. If we have a WTIA game G that consists of the two same subgames it is a losing position. We can simply extend our strategy used in Subsection 4.2[5] to show it.

If a WTIA game G consists of the standard Game of Nim subgames it is not a good research model for us, because First player can easily take all the coins from any of the piles and immediately win. After some experiments we have found a good model game. We named it Level Nim and it has helped me to find a solution how to combine games in a WTIA way.

Rules 5.1. *The game of Level Nim is played with n superpiles of coins. A superpile of coins consists of several piles of coins which are located on the each other. When it is a*

[4] maybe it is also because I like that famous song

[5] make winning move if possible or copy opponent's move otherwise

player's turn he chooses one superpile and takes at least one coin from the most top non-empty pile of this superpile. After someone takes the last coin of the superpile a game is finished and that player is declared the winner.

Example 5.1. Lets have Level Nim with two superpiles. The first superpile has piles with the sizes 3, 7, 2 and the second superpile consists of piles 4, 6. Now when it is a player's turn he is either able to take away up to 2 coins from the first superpile or up to 6 coins from the second superpile. A player can take coins from lower piles only if all higher ones are empty.

Notice that there are some special positions in the Level Nim. If one of the superpiles is empty then game is finished and we will call this position **superlosing** (If we add any other superpiles we will keep being in losing position). As opposite to them there are positions from which players can win in one move. We will call them **superwinning**. All the positions in Level Nim which are not superlosing and have at least one superpile with only one non-empty pile are superwinning, because a player whose turn it is can win by removing all the coins from this superpile.

1. After dividing all the positions to superlosing, superwinning and normal we can see that if at least one superpile has only one non empty pile then a game is in a super position else it is in a normal position.

2. If we are in a super position we know how to play. If we are in a normal position we can move to normal positions or to superwinning ones (Not to superlosing, because if we were able to then we would be in a superwinning position, not normal).

3. But nobody wants to move to a superwinning position, because he will lose in next move after that. So for players it is the same situation as if there were no moves to superwinning positions.

4. Therefore if a game G is in a normal position then the first player who will not be able to move in a normal positions will have to move to a superwinning position and he will lose. So we can reduce our problem to the ordinary sum of games where subgames are superpiles without the most bottom piles.

In the next subsection we will generalize this result for all WTIA games.

5.1 Algorithm

We will introduce w-function that gives w-number to every position of the game G and then use it to divide all the positions to winning and losing ones.

Definition 5.2. *All possible w-numbers are all non-negative integers extended by special values SL[6] and SW[7].*

[6] stands for superlosing
[7] stands for superwinning

Definition 5.3. *Let $G(X, f)$ is a game. We will assign w-number to every position $x \in X$.*

1. *$w(x) = SL$ if x is a terminal position*
2. *$w(x) = SW$ if exists a terminal position $u \in X$ and $u \in f(x)$*
3. *$w(x) = min(n \geq 0 : n \notin M)$ where $M = \{w(u) : u \in f(x), SW \neq w(u) \neq SL\}$*

Note 5.1. In words, we assign SL to the superlosing positions, SW to the superwinning positions and then assign other w-numbers in the same way as by the Sprague-Grundy function considering only a normal positions subset of X.

Corollary 5.1. *Given game $G(X, f)$. The position $x \in X$ is losing if and only if $w(x) = SL$ or $w(x) = 0$.*

Proof. If $w(x) = SL$ or SW it is clear. If $w(x) > 0$ it is possible for a player to move to $w(y) = 0$ (L-positions). If $w(x) = 0$ player can move only to the positions y where $w(y) = SW$ or $w(y) > 0$ (W-positions). □

Theorem 5.2. *Given WTIA game $G(X, f)$ that is a compound of subgames $G_1(X_1, f_1)$, $G_2(X_2, f_2)$, $\ldots G_n(X_n, f_n)$. The position $x = [x_1, x_2, \ldots x_n]$ is losing if and only if at least one of these conditions is held:*

1. *$(\exists i)$ $w(x_i) = SL$*
2. *$(\forall i)$ $w(x_i) \notin \{SL, SW\}$ and $w(x_1) \oplus w(x_2) \oplus \ldots \oplus w(x_n) = 0$*

where $x \in X$ and $x_i \in X_i$.

Proof. We will check if our division of positions is correct. All the terminal positions x have one subgame x_i that is superlosing ($w(x_i) = SL$), because if not then a player would be able to move.

If a player is in a losing position then $w(x) = SL$ or $w(x_1) \oplus w(x_2) \oplus \ldots \oplus w(x_n) = 0$. The first case we did as a part of the terminal positions. If a player moves in any subgame k then either he moves to the superwinning position in that subgame or to the position y with $w(y) \neq w(x_k)$. In both cases he can move only to winning positions.

If a player is in a winning position then $w(x) = SW$ or $w(x_1) \oplus w(x_2) \oplus \ldots \oplus w(x_n) \neq 0$. In the first case while $w(x) = SW$ there exists subgame x_k such that $w(x_k) = SW$. A player can move in this subgame to superlosing position. If xor among all $w(x_i) \neq 0$ then a player can easily move to change it to 0. He can do it in the same way as in the Game of Nim. □

Definition 5.4. *We will define operation $\oplus_w$ on games G_1, G_2 as a combination of games G_1 and G_2 in a WTIA way.*

Remark 5.1. Operation $\oplus_w$ on games is associative, commutative and $w(G_1 \oplus_w G_2)$ is equal to

1. SL if $w(G_1) = SL$ or $w(G_2) = SL$
2. SW if $w(G_1), w(G_2) \neq SL$ and if $w(G_1) = SW$ or $w(G_2) = SW$
3. $w(G_1) \oplus w(G_2)$ otherwise

5.2 Relationship between w-numbers and Grundy numbers

When using functions that assign values to the positions, like the Sprague-Grundy, remoteness, suspense function or w-function we are still working with the same game G. These functions just allow us to look at the game from different views and use the one we need to combine it with the other games.

Therefore if position in game G is losing then **it is losing in all the views from which we are looking at it**.

Corollary 5.3. *In game $G(X, f)$ $w(x) = 0$ or $w(x) = SL$ if and only if $g(x) = 0$.*[8]

Note 5.2. We know that $w(x) = 0 \Rightarrow g(x) = 0$. Notice that it does not work the other way and $g(x) = 0 \Rightarrow w(x) = 0$ is not true.

This property from Corollary 5.3 is sometimes very useful. Consider game $G = (A \oplus_w B \oplus_w C) \oplus D$ and that number of positions in all the games is about k. We want to find out if G is a losing position. Trivial algorithm can solve our problem in time $O(k^4)$ by searching through all the possible positions. We can do better, $O(k^3)$, by noticing that $g(G) = g(A \oplus_w B \oplus_w C) \oplus g(D)$. But if we are happy we can try to compute $w(A \oplus_w B \oplus_w C) = w(A) \oplus_w w(b) \oplus_w w(C)$ and if it is equal to 0 then we can deduce from that $g(A \oplus_w B \oplus_w C) = 0$ and then whole our algorithm will take only $O(k)$ steps.

5.3 Application in games

In this subsection we will demonstrate using of w-numbers on simple examples. Their application on degenerated Dots and Boxes game is in the next section.

We will start with calculation of w-numbers for Limited Nim with substraction set $S = \{1, 3, 4\}$:

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
w(n)	SL	SW	0	SW	SW	1	2	0	2	0	1	3	1	2	0	2	0	1	3
g(n)	0	1	0	1	2	3	2	0	1	0	1	2	3	2	0	1	0	1	2

Example 5.2. Lets have three Limited Nim games with this substraction set and with sizes 5, 8, 6 with the rule that the overall winner will be one who wins first of these games. A player whose turn it is in a winning position, because $w(5) \oplus_w w(8) \oplus_w w(6) = 1 \oplus_w 2 \oplus_w 2 = 1$.[9] If the sizes of piles were 4, 6, 11 then a player will be in a winning position, too, because $w(4) \oplus_w w(6) \oplus_w w(11) = SW \oplus_w 2 \oplus_w 3 = SW$.

Note 5.3. In above table for Limited Nim we can see that Corollary 5.3 really works. We can also see that both w-numbers and grundy numbers become periodic. It is because value for position x depends only on last k values where k is the maximum number in substraction set.

[8] We can create similar equivalences for the other functions as well

[9] by n in $w(n)$ we mean the Limited Nim position of size n

Now consider All the King's horses game with slightly changed rules. A player whose turn it is must choose one horse and move with him. The player who first moves any horse home (to left-up 2×2 corner from which horses are unable to move) is declared the winner.

This game is a WTIA combination of the games with only one horse on each chessboard. Figure 9 shows us w-numbers that will help us to determine the winner. To find out who is the winner of a game it is enough to use the operation $\oplus_w$ among the positions of all the horses that are placed on chessboard.

SL	SL	SW	SW	0	0	1	1
SL	SL	SW	SW	0	0	2	1
SW	SW	SW	SW	1	2	3	2
SW	SW	SW	1	2	1	2	1
0	0	1	2	0	0	1	2
0	0	2	1	0	0	2	1
1	2	3	2	1	2	1	2
1	1	2	1	2	1	2	0

Figure 9: w-numbers for 8×8 All the King's horses modification

Note 5.4. Like we are combining different games using Grundy numbers we can combine different games also using w-numbers. For example All the King's horses modification with Limited Nim games from Example 5.2. We just put together their w-numbers in a usual way.

5.4 Misére play rule

In the previous subsections we explained the strategy for the WTIA games played under the normal play rule. Now we will share our strategy for the WTIA games played under the misére play rule[10]. In the degenerated Dots and Boxes, it means, that now we have a strategy how to force our opponent to create the first box. All other sums of the games presented in this thesis does not have known strategies for misére play rule and therefore this discovery of strategy for WTIA games was quite unexpected. Probably newly introduced w-numbers helped a lot.

The difference between the WTIA sum of games played under the normal play rule and the WTIA sum of games played under the misére play rule is that now all terminal positions are superwinning, nobody wants to move to them, because they mean immediate lose. Therefore it is the same as if a player will not have the possibility to move to them. Only if a player cannot move in normal positions he must move to a superwinning one. Therefore our problem is reduced to the ordinary sum of the games played on the subgraphs without the terminal (SW) positions. And while we know how to play the ordinary sum

[10]The player who first wins (finishes) the first subgame is declared the loser

of the games we know how to play the WTIA sum of the games played under the misére play rule.

Definition 5.5. *Let $G(X, f)$ is a game. The function w_m will assign each position $x \in X$ non-negative integer or the value SW:*

- *if x is a terminal postion then $w_m(x) = SW$*
- *else $w_m(x) = min(n \geq 0 : n \notin M)$ where $M = \{w_m(u) : u \in f(x), SW \neq w_m(u)\}$*

Note 5.5. In words, we assign SW to the superwinning positions and then assign other w-numbers in the same way as by the Sprague-Grundy function considering only a normal positions subset of X.

Corollary 5.4. *Given game $G(X, f)$ played under the misére play rule. The position $x \in X$ is losing if and only if $w_m(x) = 0$.*

Now we can use that we have knowledge of w-numbers and operation $\oplus_w$ to provide simpler way to identify losing positions.

Theorem 5.5. *Given WTIA game $G(X, f)$ that is a compound of subgames $G_1(X_1, f_1)$, $G_2(X_2, f_2)$, $\ldots G_n(X_n, f_n)$ and that is played under the misére play rule. The position $x = [x_1, x_2, \ldots x_n]$ is losing if and only if $w_m(x_1) \oplus_w w_m(x_2) \oplus_w \ldots \oplus_w w_m(x_n) = 0$, where $x \in X$ and $x_i \in X_i$.*

Proof. If we are in a terminal position, one of the games has already finished, $w_m(x) = SW$, so we are in a winning position. If we are not in a terminal position and $w_m(x) > 0$ it means we can move to the losing position u for which $w_m(u) = 0$, therefore we are in a winning position. If $w_m(x) = 0$ it means we cannot move to another position u for which $w_m(u) = 0$, so we can move only to the positions which w_m value is SW or higher than 0. All these positions are winning so we are in a losing position. □

6 The Dots and Boxes game reloaded

In this section we will go back to the degenerated Dots and Boxes game. We will use the fact that now we have a new tool introduced in the previous section: w-numbers.

We could examine smaller games by using the standard winning-losing algorithm by searching through the all positions of the game. The problem is that this would take huge amount of time for bigger games. If we have $m \times n$ Dots and boxes game, it has $m(n-1)+(m-1)n$ lines, so it has almost $2^{m(n-1)+(m-1)n}$ positions. If we want to examine all of them the complexity of the standard algorithm will be $O(2^{m(n-1)+(m-1)n})$ that is almost the same as $O(2^{2mn})$. Because of that nowadays it takes about a minute to examine 4×4 position of the game, about a half of the day to examine 4×5 position and about a year to examine 5×5 position.

Fortunately we can profit from the fact that we know better the structure of some positions and that they are just a WTIA combination of subgames. We will describe an algorithm in the second part of this section.

6.1 Values for smaller games

In Figure 10 we can see some randomly chosen positions of the degenerated Dots and Boxes game. We have analyzed these positions and have found w-numbers by using an algorithm described in the previous section.

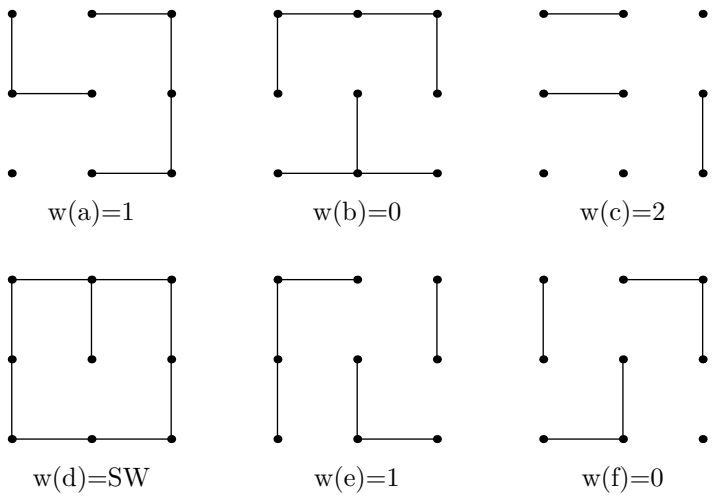

Figure 10: Positions in degenerated Dots and Boxes game

Now we can compare these results with "look and see" approach:

1. It is obvious that the position d (in Figure 10) is superwinning, because a player will win by adding any new line.

2. While we were proving Theorem 4.1 we proved that all the symmetric positions with at most two lines in any box are losing ones. Therefore the position b is losing. On the other hand the positions a and c are winning, because a player can add a line to make them symmetric with at most two lines in any box.

3. A player can add an upper right horizontal line in the position e to make all of the boxes to have exactly two lines. So his opponent will have to move to a superwinning position and therefore the position e is winning.

6.2 Computation of values for bigger games

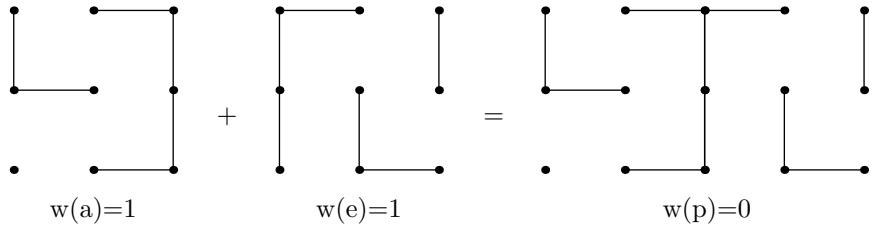

Figure 11: Sum of the positions in degenerated Dots and Boxes game

Example 6.1. While the position p (in Figure 11) is a WTIA sum of the positions a and e (which we have already met in Figure 10) $w(p) = w(a) \oplus_w w(e) = 1 \oplus_w 1 = 0$.

Similarly we can divide every degenerated Dots and Boxes game to independent parts, compute w-number for every subgame and then combine it together. This can extremely improve our ability to find if a certain position is winning (losing).

Our new algorithm that is determining if the position x is winning (losing) will work in the following way:

1. The position x is losing iff $w(x) = 0$ or $w(x) = SL$.

2. To compute $w(x)$ we will start by dividing x into independent parts. We can do this by performing BFS[11] that will take about $O(mn)$ operations where the position x has the size $m \times n$.

3. We will recursively calculate w-numbers for every independent part and then combine them together.

Very important fact is that m and n are much smaller (exponentially) than a number of positions in a game. In practice we are interested in computing values for up to 5×6 games.

[11] Breadth-first search

Although BFS for every position will slow us down about mn times, we are interested in games where mn is only about 30, so we can look at it as at small constant.

On the other hand we will gain much more. In Figure 11 we needed only about 2^{12} operations to calculate and combine values for the positions a and e. If we used the standard algorithm to find a value for the position p we would need about 2^{27} operations. We are calculating w-numbers recursively for the subgames so we will get the results even faster as an additional bonus.

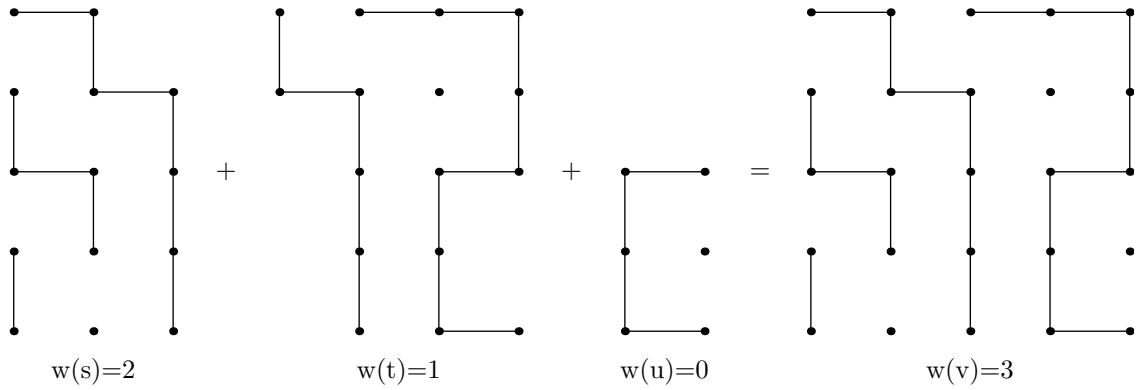

Figure 12: Sum of the positions in degenerated Dots and Boxes game

Example 6.2. Figure 12 shows us how to divide the position v into the subgames s, t and u. While v is a WTIA combination of them we obtain that $w(v) = w(s) \oplus_w w(t) \oplus_w w(u) = 2 \oplus_w 1 \oplus_w 0 = 3$. So now we know much faster that the position v is a winning one. And much more we know that we can find a winning move in the subgame s and decrease its w-number to 1 to obtain overall sum equal to 0. All boxes in the position s except two at the bottom has two lines. If we add to one of these two a bottom line our opponent will have to add the second one (or move to a superwinning position that means an immediate lose for him). So if we take one of these two bottom lines we will change its w-number to 1.

7 API for combinatorial games

The goal was to create a supporting tool for the research of mathematical (combinatorial) games. It is especially for the impartial games with the normal play rule, but it is planned to be extended to support also partizan games and games with the misére play rule in the future.

There is a similar and more complex existing solution of combinatorial game suite[12], developed mainly by Aaron Siegel, that is quite flexible, written in Java, with nice environment that is similar to the Maple and Mathematics. It has support for plug-ins for new games and its own interpreted programming language. On the other hand while it is quite complex it is needed to implement a lot of interfaces that are not necessary need or develop programs in a non-standard language, so it is not so simple to connect it with an external environment.

7.1 Overview

During designing and developing this API we keep emphasis on open, simple and easily extensible solution. Because of that we used all advantages of object oriented programming and many design patterns like Strategy, Decorator and Visitor that helped me to achieve this goal.

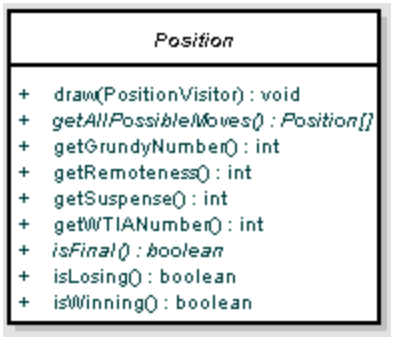

Figure 13: UML class diagram for the class Position

It is whole based on an abstract class **Position** that represents a position in a game. To create a concrete position we must implement two basic methods. Method **getAllPossibleMoves()** should return the positions to which we can move from the current position and method **isFinal()** should return true iff our position is terminal.

These two methods specify the game tree in a way it is defined and used in this thesis. Therefore we can use the corresponding algorithms introduced in the previous sections, independent from the types of games, that operate on these game trees:

- **isWinning()** - returns if the current position is winning

[12] it can be downloaded from `http://cgsuite.sourceforge.net/`

- **isLosing()** - returns if the current position is losing
- **getGrundyNumber()** - returns Grundy number for the current position
- **getWTIANumber()** - returns w-number for the current position
- **getRemoteness()** - returns remote function value for the current position
- **getSuspense()** - returns suspense function value for the current position.

Using Strategy pattern, all these algorithms are implemented in their own classes (see Figure 14) and abstract class Position is just calling them by default. Of course they can be overridden if we are creating our game by extending the class Position and know from the structure of the game how to compute any of these values faster. We will return back to this option later on.

All algorithms support **built-in caching** to improve performance and not to compute anything twice. But if we want to use it we have to tell the computer when two positions are equal. To implement this we have to override methods **equals(Object o)** and **hashCode()**. While implementing them keep in mind that according to Java contract if two objects that are equal they must have the same hashcode[13].

The last method of the Position class is **draw(PositionVisitor v)**. This method introduces a design pattern Visitor to our class Position. By invoking it it calls the method draw on PositionVisitor and provides him whole position as a parameter. Therefore if user wants to get information about the current position he just needs to implement interface PositionVisitor and call method draw with it as a parameter. This implementation decouples any output about the current state from the logic of the position behaviour. We have also freedom to have multiple visitors for one game: one to generate text output to the console or graphical swing, html, ... For example we have created metapost PositionVisitor (see Figure 16) for the Dots and Boxes game to generate pictures of the positions for this thesis.

7.2 More details of technical design

An idea was to create an environment that will allow us to create games as a compound of the other games and perform all operations on them like if they were simple games. On the other hand we want to profit from the fact that we know the structure so we can compute some values like for example Grundy numbers much faster. In Figure 15 it can be seen how it is implemented. We used design pattern Decorator that allows us to "decorate[14]" simple games to create composite games and access them just like they were simple ones.

The Decorator pattern allows us to create any combination of different simple or composite games. For example we can create a game G =**WTIAPosition(LimitedNimPosition(), OrdinarySumPosition(BoardPosition(), LimitedNimPosition()))**. This notation

[13] equals should be also reflexive, symmetric, transitive and consistent

[14] pack

means that we are playing two games with the winner takes it all rule where the first game is a Limited Nim game and the second one is a composite game that contains two other games played together as an ordinary sum. If somebody asks what is grundy number of the game G we will call the standard algorithm which looks at our composite game as at a one big game tree. However, if we want to know w-number of the game G we can use our knowledge about WTIA games and compute it as a w-number of Limited Nim $\oplus_w$ w-number of an ordinary sum of those two games. To achieve this goal we have just overridden getWTIANumber() method in a WTIAPosition class. Therefore in this case it will compute w-number in a more clever way and if a game is not a WTIAGame it will use the standard algorithm (WTIAAlgorithm.getWTIANumber()) that always works. Similarly we can override method getGrundyNumber() in OrdinarySumPosition class and methods isWinning() and isLosing() in all the classes where we can find the answers faster by using for example Grundy numbers.

This combinatorial game API contains:

- basic framework (Position, PositionVisitor and algorithms classes)
- composite games support (OrdinarySumPosition, WTIAPosition, ANDPosition)
- Limited Nim game implementation
- degenerate Dots and Boxes game implementation
- board games implementation

If a user wants to create a new game he can do it very fast and Limited Nim is a very simple example. He just need to extend Position class and implement methods getAllPossibleMoves() and isFinal(). If he wants to use caching then do not forget to implement equals() and hashCode() methods. After that he is finished and he can call on his game any algorithm he wants and combine it with any other games.

Board games covers (except many other games) all variants of Game of Queens and All the King's horses. These games are very good for any experiments.

Finally, there is attached an implementation of the degenerate Dots and Boxes game. As it can be seen in Figure 16 there are two classes DotsAndBoxesPosition and DotsAndBoxesSmallPosition. The first one covers only positions of rectangular shape. The second one extends the first one by optional disabling the boxes we do not want to be a part of the position. It is essential when we are decomposing big games to smaller ones. Note that they share their cache and by careful implementation of equals() and hashCode() methods they can share their results[15]. To have a debugging output there is implemented a Text visitor and to have a nice MetaPost output there is a MetaPost visitor.

[15] DotsAndBoxesSmallPositions with no disabled boxes are the same as DotsAndBoxesPositions if the board of both is the same

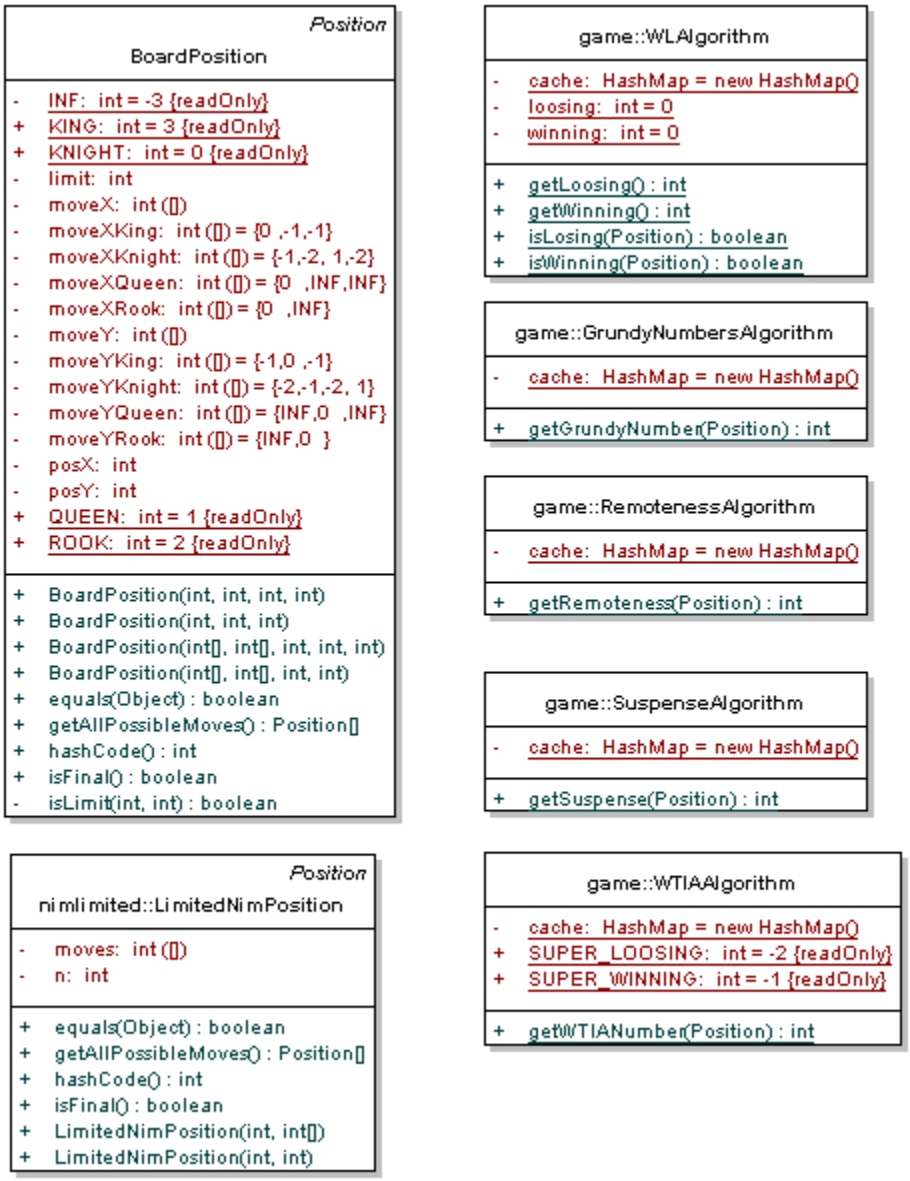

Figure 14: UML class diagram for the games NimLimited and Board Games on the left and class diagrams of algorithms in package sk.lenhardt.game on the right

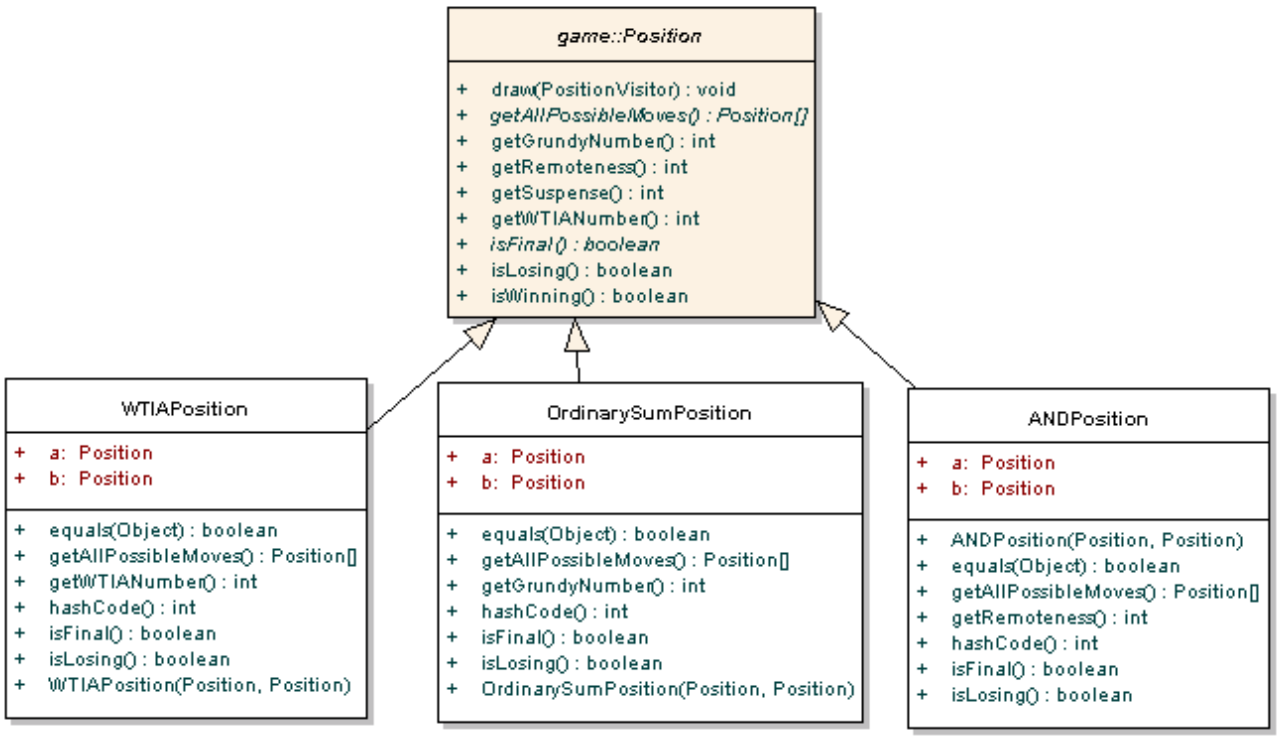

Figure 15: UML class diagram for the classes of composite games in sk.lenhardt.sum

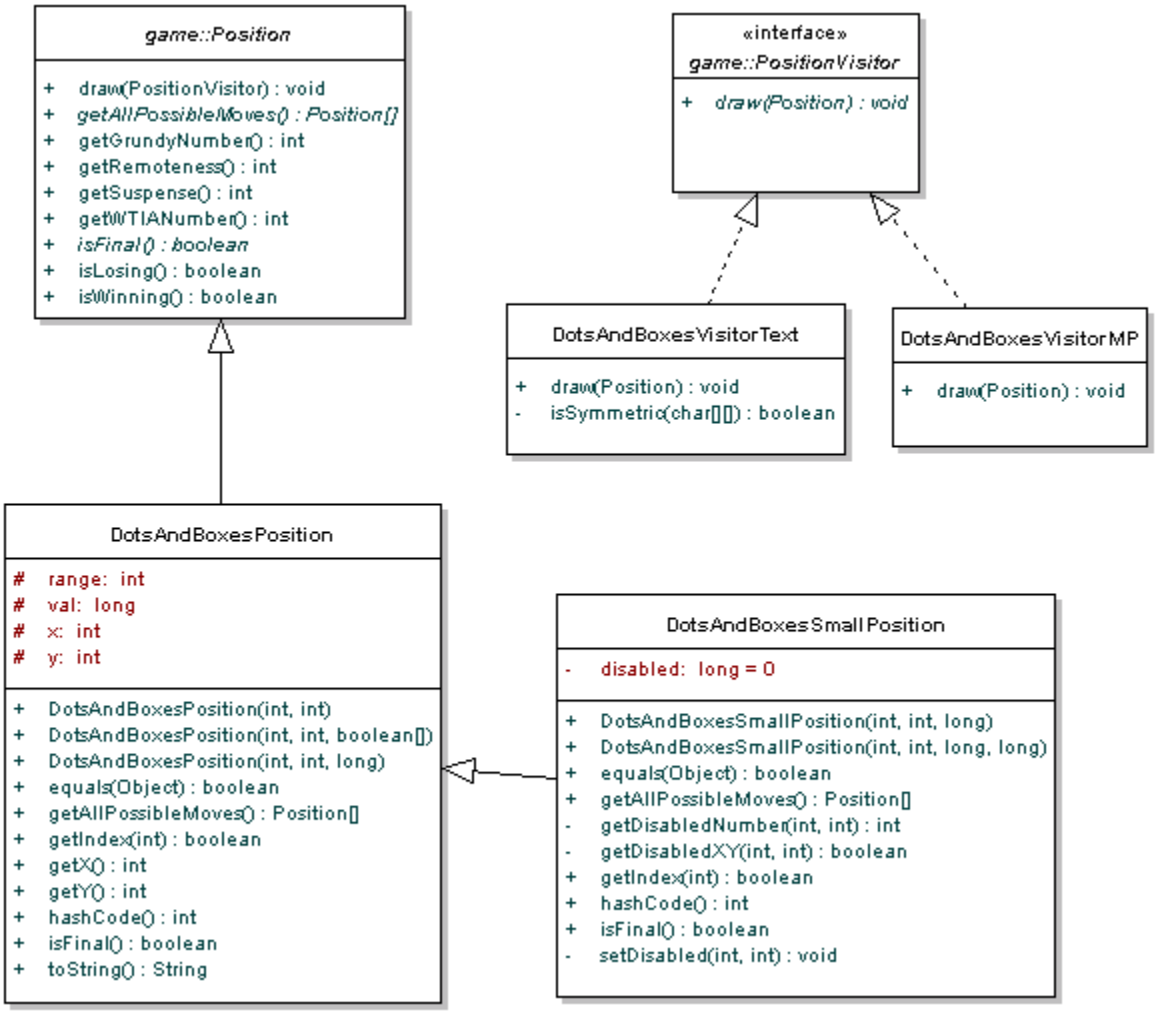

Figure 16: UML class diagrams for the Dots and Boxes game

8 Conclusion

The most important result of this thesis is finding the way how to play the composite games where the winner is the player who wins the first subgame. We have introduced a new w-function and an algorithm that can help us solve such composite games. Because of that now we can analyse these composite games in $O(kn)$ time in comparison to $O(n^k)$ obtained by the standard algorithm, where n is the size of particular subgames and k number of subgames. We have discovered also the way how to play it with the misére play rule.

We have found out all this while looking for a winning strategy for the degenerated Dots and Boxes game. Now we can say that we have found this strategy. It can be used and be very helpful for the players of the original Dots and Boxes game. However, it cannot be used every time. So it opens a new question how to profit from it as much as possible to become closer to the solution to the original game. As it is said in [Be00] a player could not know everything about Dots and Boxes until he knew everything about NimString[16]. Similarly we can say that if we had not know how to play the degenerated variation, we would not know everything about Dots and Boxes. So this is another part of the puzzle called Dots and Boxes solution.

One of the goals of this thesis was also to create an introduction to (composite) mathematical games for students at undergraduate level. Therefore we have presented five known ways of adding impartial games with proofs and many examples. Almost all these games and algorithms are implemented in a new created Combinatorial game API. This API provides an environment to experiment with the impartial games. While almost all algorithms presented in this thesis are implemented, researching a new game can be easily started. It is needed only to implement a simple interface to provide what are possible moves from the position. After that the user can start combining the games and finding out values for the positions like Grundy numbers, w-numbers, remoteness, suspense, ... We used it a lot to gain and check the results and produce all the figures and tables. In the future the API could be extended to support games with misére play rule and also partizan games.

In this thesis we are limited to games with a win-lose outcome. From a general point of view, another challenging task would be to find out if it is possible to apply some of these results to games where we have more possible outcomes. There are games where also draws are possible. And there are many games that consist of parts and the final outcome is the sum of outcomes from the subgames. Finding out that would have great applications in Economics, as well. We know how to find out outcomes for simple games by using the standard algorithms like min-max and alpha-beta pruning. One of the possible ways could be to reduce them to win-lose outcome games. For example we would declare the player A the winner iff he scores at least n. By trying all possible n we could find out how many points he could secure by playing optimally.

[16] The NimString is a variation of Dots and Boxes where the winner is a player who creates the last box

A CD attachment

An attached CD contains whole Combinatorial Game API as a project. It includes all source codes, javadoc and compiled classes.

References

[1] [Ww01] Elwyn R. Berlekamp, John H. Conway, Richard K. Guy *Winning ways for your mathematical plays* (Volumes I-IV, Second edition) 2001: A K Peters, Ltd. Wellesley, MA

[2] [Be00] Elwyn R. Berlekamp *The dots-and-boxes game: sophisticated child's play* 2000: A K Peters, Ltd. Wellesley, MA

[3] [Co01] John H. Conway *On numbers and games* (Second edition) 2001: A K Peters, Ltd. Wellesley, MA

[4] [Ferg] Thomas S. Ferguson *Game Theory Text* [`http://www.math.ucla.edu/~tom/Game_Theory/Contents.html`] Mathematics Department, UCLA

[5] [Lim] Lim Chu Wee *Introductory Combinatorial Game Theory* [`http://sps.nus.edu.sg/~limchuwe/cgt`]

[6] [Fra] Aviezri S. Fraenkel *Combinatorial Games: Selected Bibliography with a Succint Gourmet Introduction* The electronic journal of combinatorics, 2007 [`http://www.combinatorics.org/Surveys/ds2.ps`]

List of Figures

Abstrakt

Dekompozícia matematických hier na menšie časti je nevyhnutná kvôli veľkému počtu pozícií. V práci sa venujeme degenerovanej verzii známej hry Dots and Boxes, v ktorej sa víťazom stane hráč, ktorý ako prvý vytvorí štvorček. Táto verzia úzko súvisí s orginálnou hrou a často nám pomôže nájsť aj v nej víťaznú stratégiu. Objavili a dokázali sme, ktorý hráč má víťaznú stratégiu z počiatočnej prázdnej hracej plochy. Pre rozohraté hry sme si všimli, že sa samotná hra rozdeľuje na menšie hry, kde víťaz celej hry sa stane hráč, ktorý ako prvý vyhrá jednu z podhier.

Tento typ zložených hier sme následne skúmali vo všeobecnosti. Zaviedli sme w-funkciu, ktorá jednotlivým pozíciam v podhrách priraďuje w-čísla, ktoré vieme spájať a zistiť tak aj w-číslo celej zloženej hry. Na jeho základe vieme povedať, či je daná pozícia vyhrávajúca alebo prehrávajúca. Tieto výsledky nám umožnili nájsť riešenia hier oveľa rýchlejšie. Ak n je veľkosť jednolivých hier a k počet hier, tak zložitosť nášho algoritmu na zistenie výhernosti pozície je len $O(kn)$ v porovnaní s $O(n^k)$ získaných použitím štandardného algoritmu. Tento všeobecný poznatok sme následne použili na hľadanie víťazných stratégií v degenerovanej verzii hry Dots and Boxes. Samozrejme, môže byť použitý aj v kombinácii ľubovoľných iných hier. Našli sme tiež spôsob ako hrať túto kombináciu zložených hier s pravidlom, že hráč, ktorý vyhrá prvú z podhier, celkovo prehrá.

Prvá časť práce slúži ako úvod do problematiky a obsahuje známe spôsoby kombinácie hier s dôkazmi a mnohými príkladmi. Vytvorili sme tiež API pre matematické hry na podporu a zlepšenie možností výskumu v oblasti matematických hier. Umožňuje jednoducho a rýchlo zadefinovať herný strom, nad ktorého abstrakciou fungujú naimplementované algoritmy. API sme používali a pomocou neho získavali výsledky a aj obrázky. Takmer všetky hry, zahŕňajúc aj degenerované Dots and Boxes, sú v ňom aj naimplementované.

Kľúčové slová: Dots and Boxes, zložené hry, w-čísla