

BOSTON UNIVERSITY  
GRADUATE SCHOOL OF ARTS AND SCIENCES

Dissertation

**SCIENCE FOR FUN: NEW IMPARTIAL BOARD GAMES**

by

**KYLE WEBSTER BURKE**

B.A., Colby College, 2003

Submitted in partial fulfillment of the  
requirements for the degree of  
Doctor of Philosophy

2009

UMI Number: 3357626

### INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

**UMI**<sup>®</sup>

---

UMI Microform 3357626

Copyright 2009 by ProQuest LLC.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest LLC  
789 E. Eisenhower Parkway  
PO Box 1346  
Ann Arbor, MI 48106-1346

Approved by

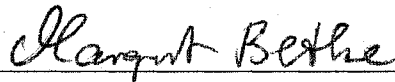
First Reader



---

Shang-Hua Teng, Ph.D.  
Professor of Computer Science

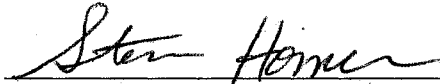
Second Reader



---

Margrit Betke, Ph.D.  
Associate Professor of Computer Science

Third Reader



---

Steve Homer, Ph.D.  
Professor of Computer Science

## DEDICATION

I dedicate this work to my parents, Robert D. and Janelle B. Burke, who have given me all the opportunities I needed to get this far.

# Acknowledgments

Many people have contributed to the success of this dissertation and my academic growth. I have been very lucky to fall on this path of game development and study, but it was not simply luck that has given me the chance to work in such a pleasing area of research. Officemates, professors, teachers, collaborators, family and friends have all helped me to excel and enjoy what I do. I would like to thank every one of these people, but they are far too numerous to include here. Some, however, have had a more direct influence on this dissertation than others, and I would like to individually acknowledge each of them.

Shang-Hua was my advisor before he was my advisor, giving me food for thought before I had yet become a student at Boston University. He has given me a great deal of freedom in my research, without which these games would not exist. Although combinatoric game theory was not his main focus of research, he gave me all the encouragement and direction necessary to bring this work together. Shang-Hua's devotion to this project is evident; he now sees new impartial games wherever he goes! I am certain we will work together in the future on more new board games. I have not met a single person who could have led me with the same confidence and courage, instilling in me the desire to succeed, while keeping me grounded in the complexities of our field.

I would like to also thank Leo Livshits, from whom I took my sole game theory course. During this course, Leo assigned me a project on the relationship between Hex and the Brouwer Fixed-Point theorem. Without this project, it is likely that I would have never had the off-hand thought that Sperner's Lemma could be made into a game!

I would like to thank Jan Janigian for introducing me to Nim in grade school. Mr.

Janigian saw my interest in games and pushed me to hone my strategic skills. His quarter-long class on mathematical games made a lasting impression on me.

I have taught with and for many people, but I would like to single out Otto Bretscher, for whom I have taught the most and with whom I discovered my deep love for teaching. I am extremely pleased to find that the games we describe here have such potential in the classroom and not just in theory books and gaming halls.

My undergraduate research adviser, Marc Smith, taught me the joys of research and of presenting my own work. The quality of the presentation of this thesis is reflective of the standards he taught me six years ago.

To my two readers, Steve Homer and Margrit Betke, I offer my thanks for more than just agreeing to read an extra-long document about some abstract games. Steve was instrumental in bringing me to Boston University, a place where I have unexpectedly flourished, and I am very lucky my application caught his eye six years ago. On the other hand, Margrit provided me with the wonderful opportunity to use Atropos as a project in her AI class here. This experience provoked me to try to explain my strategies while playing the game as her students came by to play against me. Somehow her students were able to take some of my gobbledygook explanations and encode them into their successful game-playing programs!

During my time at Boston University, I have needed a lot of technical help on my own system. One of my officemates, Debajyoti Bera, has been of immense help over the past six years, not only as a personal IT assistant, but also as a close friend.

Also in graduate school I met David Charlton, who made a great effort to help me through my quals, which were a very difficult road block for me. With his assistance, I was able to recover from my first attempt and pass on my second just before the deadline. I attempted to repay the favor for his quals, but he hardly needed my help.

The end of this journey has been the most difficult, and much of this burden has been eased by the love and attention of Olivia George. Many times she has guided me away from a stressful collapse and kept me going forward with this dissertation. She also has helped

guide this work by playing each of the games with me, often testing many different variants as I made improvements.

Most importantly, I would never have had the chance to even consider pursuing graduate study without the love, support, and every facet of guidance from my parents. They have provided me with opportunities, but mostly have lent their understanding, even when my attitude did not in any way deserve it.

Finally, I have played Atropos, Matchmaker and Dictator with too many people to mention over the past many years. To all those who agreed to sit down with me and help me out with “research”, I thank you. None of these games were designed in one stroke, and it is in this playtesting that the strengths and weaknesses of a game are discovered.

# SCIENCE FOR FUN: NEW IMPARTIAL BOARD GAMES

(Order No.                    )

**KYLE WEBSTER BURKE**

Boston University, Graduate School of Arts and Sciences, 2009

Major Advisor: Shang-Hua Teng, Professor of Computer Science

## ABSTRACT

We enhance the realm of combinatorial game theory by introducing three new impartial games. These games are born from economic and mathematical topics—both current and classic—in an effort to gather interest and provide potential teaching tools. The three games are Atropos, based on Sperner’s Lemma; Matchmaker, inspired by the Stable Matching problem; and Dictator, based on Arrow’s Impossibility Theorem.

The first of these, Atropos, has already gained some popularity. Here, players alternate choosing colors for vertices of a two-dimensional Sperner simplex. A player loses when their play creates a three-colored triangle. We show many variants of Atropos, one which is PSPACE-complete.

Our second game, Matchmaker, uses the two players as matchmakers for two lists of candidates. In order to increase the playability, we have limited the scope to settings with universal preference lists. Players take turns matching one couple and the player who makes the last move to reach the stable configuration wins the game. Again, we have many versions of this game and show a few which are solvable in polynomial time.

The last of these games, Dictator is played on an partially-filled list of ballots which are the input to a voting function. In this game, players take turns adding a candidate to one of the ballots, but must avoid creating a situation either where one of the ballots matches the output or the ballots cause the function to violate one of Arrow’s requirements for fairness. Two natural problems arise: how to detect violations by the opponent and how to play so that the opponent is the first to cause a violation. We address both in different versions of Dictator.



In addition, we discuss the general benefits of such games, as well as how computational complexity is an important aspect of their enjoyability. Finally, we describe open problems from our games as well as suggest some new possibilities for future combinatorial games.

# Contents

<b>Acknowledgments</b>	<b>iv</b>
<b>Contents</b>	<b>ix</b>
<b>List of Tables</b>	<b>xii</b>
<b>List of Figures</b>	<b>xiii</b>
<b>List of Abbreviations</b>	<b>xvii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Notation and Definitions</b>	<b>5</b>
2.1 Basic Definitions . . . . .	5
2.2 Definitions of New Games . . . . .	7
2.2.1 Atropos . . . . .	7
2.2.2 Matchmaker . . . . .	7
2.2.3 Dictator . . . . .	7
<b>3 Game Creation &amp; Design</b>	<b>9</b>
3.1 Hex . . . . .	9
3.2 Sperner's Lemma . . . . .	10
3.2.1 Three-Player Atropos . . . . .	12
3.2.2 Unrestricted Atropos . . . . .	12
3.2.3 (Restricted) Atropos . . . . .	13
3.3 Stable Marriage . . . . .	13
3.3.1 Full Matchmaker . . . . .	15
3.3.2 Matchmaker . . . . .	16

3.3.3	Stubborn Matchmaker . . . . .	16
3.4	Arrow's Theorem . . . . .	17
3.5	Summary . . . . .	29
<b>4</b>	<b>Computational Complexity of Games</b>	<b>31</b>
4.1	Atropos . . . . .	32
4.1.1	Taking Hints from GEOGRAPHY . . . . .	33
4.1.2	Reduction Widgets . . . . .	35
4.1.3	Paths . . . . .	36
4.1.4	Variables . . . . .	38
4.1.5	Splitting and Rejoining Paths . . . . .	40
4.1.6	Path Crossing . . . . .	41
4.1.7	Parity Switching . . . . .	42
4.1.8	Sample Reduction . . . . .	43
4.1.9	Some Additional Concerns . . . . .	45
4.1.10	Algorithmic Reduction: Putting the Pieces Together . . . . .	46
4.2	Unrestricted Atropos . . . . .	51
4.3	Full Matchmaker . . . . .	57
4.4	Basic Dictator . . . . .	59
4.5	Dictator . . . . .	61
4.6	Other Game Complexities . . . . .	62
<b>5</b>	<b>Programming to Solve Games</b>	<b>63</b>
5.1	Nimbers for Impartial Games . . . . .	63
5.2	Atropos . . . . .	64
5.3	Stubborn Matchmaker . . . . .	66
<b>6</b>	<b>"Making it real"</b>	<b>73</b>
6.1	Atropos . . . . .	73
6.1.1	Correctly Coloring the Border . . . . .	73
6.1.2	Borders in the Physical Prototype . . . . .	75

6.1.3	Extensions to the Physical Game . . . . .	75
6.1.4	Atropos with Dice . . . . .	76
6.1.5	Atropos with Cards . . . . .	79
6.1.6	An Automated Atropos Opponent . . . . .	81
6.1.7	Atropos Variants . . . . .	82
6.1.8	Atropos for Facebook . . . . .	83
6.2	Matchmaker . . . . .	83
6.2.1	Poster Matchmaker . . . . .	83
6.3	Dictator . . . . .	85
6.4	Summary . . . . .	86
<b>7</b>	<b>Pedagogical Contributions</b>	<b>88</b>
7.1	Atropos . . . . .	88
7.2	Matchmaker . . . . .	89
7.3	Dictator . . . . .	90
<b>8</b>	<b>Conclusions</b>	<b>91</b>
8.1	Atropos . . . . .	91
8.2	Matchmaker . . . . .	92
8.3	Dictator . . . . .	92
8.4	Overall Conclusions . . . . .	93
<b>9</b>	<b>Future Work</b>	<b>95</b>
9.1	Atropos . . . . .	95
9.2	Matchmaker . . . . .	96
9.3	Dictator . . . . .	97
	<b>Bibliography</b>	<b>99</b>
	<b>Curriculum Vitae</b>	<b>102</b>

# List of Tables

5.1	Some known results for Stubborn Matchmaker. The left-hand side of the table lists the number of unmatched pairs. Above, the first number in the pair is the minimum number of loners, while the right is the larger number of loners. Listed numbers assume the board has no crosses. . . . .	70
5.2	More known results for Stubborn Matchmaker. The notation is the same as in the previous table, but we start with five free pairs, as boards with less do not exist at this size. . . . .	71
5.3	More known results for Stubborn Matchmaker. The notation is the same as in the previous table, but we start with six free pairs, as boards with less do not exist at this size. . . . .	71
5.4	More known results for Stubborn Matchmaker. The notation is the same as in the previous table, but we start with eight free pairs, as boards with less do not exist at this size. . . . .	72

# List of Figures

2.1	From left to right: A barred circle, a shaded circle and a filled circle. . . . .	6
3.1	An Empty Hex Board. . . . .	10
3.2	An empty Atropos Board of size seven (the number of open circles on a side). . . . .	11
3.3	Left: empty Matchmaker board. Middle: fully-matched board. Right: the only stable Matchmaker configuration possible. . . . .	15
3.4	An empty Dictator board, where $n = 4$ and $m = 5$ . . . . .	19
3.5	This board violates Pareto efficiency; in all ballots, $D > B$ . . . . .	19
3.6	This board also violates Pareto. . . . .	20
3.7	We can use IIA to show that this board violates Pareto. . . . .	20
3.8	This board violates Pareto: $D > A$ in all ballots. . . . .	21
3.9	The three lower boards are the candidate cases of the top board at the entry marked with a star. . . . .	24
3.10	This board cannot have a dictator, but we cannot show a Pareto violation in Dictator. . . . .	26
3.11	Left: counterexample for using $\{A, B, D\}$ transformations. Right: counterexample for $\{A, C, D\}$ . . . . .	27
3.12	We need more than one IIA-manipulation on this board to find a violation. . . . .	28
3.13	A sequence of IIA-manipulations leading to a Pareto violation. . . . .	29
4.1	Construction Sketch: End of the Game. Clause $c_i$ contains the literals $x_a, x_b, x_c$ . . . . .	34
4.2	Construction Sketch: Setting the Variables. The Variable Investigation Scheme is laid out in Figure 4.1 . . . . .	35

4.3	A Single-Symbol Path . . . . .	36
4.4	A Two-Symbol Path . . . . .	36
4.5	A Switch from Two-Symbols to Single-Symbol . . . . .	37
4.6	A 60 degree turn in a One-Symbol Path. . . . .	37
4.7	Another 60 degree turn in a One-Symbol Path. . . . .	38
4.8	Variable Widget . . . . .	38
4.9	A Premature Play at $x_i$ . . . . .	39
4.10	This widget splits a path. . . . .	40
4.11	This widget converges two paths. . . . .	41
4.12	This widget crosses two paths. . . . .	42
4.13	This widget effectively switches the order of play in a path. . . . .	43
4.14	Atropos board equivalent to $\exists x : \forall y : \exists z : [(\bar{x} \vee \bar{y}) \wedge (y \vee z) \wedge (y \vee \bar{z})]$ . . . .	44
4.15	Filling in Holes Between the Widgets. We also remove the unnecessary notation from the nodes ( $S$ is still the start node, and must be denoted). . . .	45
4.16	An empty hex tile. The border of the tile will overlap with neighboring tiles.	46
4.17	Two paths come in from the top, cross paths, and exit through the bottom.	48
4.18	These paths don't need to cross and thus just go straight down. . . . .	49
4.19	This tile joins two paths from above and exits to the right. . . . .	50
4.20	This tile joins two paths from above and exits to the left. . . . .	51
4.21	This tile contains a variable widget. . . . .	52
4.22	In this splitter tile, one path splits into two. . . . .	53
4.23	Basic parity "switches". A play in either of these will create an even number of bad triangles. We refer to these as A-Switches. . . . .	53
4.24	New Symbols for parity switch diagrams. On the left, the circle does not have any safe colors available. On the right, the circle can only be colored with bars. . . . .	54
4.25	B-Switch. On the left is the general form. The right figure is a diagram of an example B-Switch. . . . .	54

4.26	C-Switch. On the left is the general form. The right figure is a diagram of an example C-Switch. . . . .	55
4.27	D-Switch. On the left is the general form. The right figure is a diagram of an example D-Switch. . . . .	55
4.28	E-Switch. On the left is the general form. The right figure is a diagram of an example E-Switch. . . . .	56
4.29	In a Basic Dictator game. Only the third ballot could be the dictator. Thus, players must avoid playing in that ballot as long as possible. Any play there will either create the dictator or make the dictator impossible, violating Pareto.	60
4.30	Near the end of a Basic Dictator game. Both of the unfilled ballots have the potential to be a dictator. The next player must choose one of them and play so that the ballot is not a dictator. The other player will then lose by either creating the Dictator or violating IIA and Pareto. Even though there are potentially more than one Dictator, this does not change the parity of the outcome. . . . .	61
5.1	We conjecture: the number of the left game is equal to the number of the right game $\oplus 1$ . . . . .	67
5.2	Left: a matching, $M$ . Right: $M(3, 5)$ . We conjecture that the numbers for these boards are the same. . . . .	67
5.3	Left: a matching, $M$ . Right: $M_{\text{un-Z}}(5, 3)$ . We conjecture that the numbers for these boards are the same. . . . .	68
6.1	The online Atropos applet. . . . .	74
6.2	Layout of coloration for prototype board. We can cover inside rows with white poker chips to increase the size of the board. . . . .	76
6.3	Photo of the actual prototype. . . . .	77
6.4	An example for the dice version of Atropos. If the next play must take place in the top-most open circle, we can analyze the probability a player losing in the next few turns. . . . .	78



6.5 Applet for Matchmaker. . . . .	84
------------------------------------	----

# List of Abbreviations

EXPTIME	Exponential Time
IAP	Industrial Affiliates Program
IIA	Independence of Irrelevant Alternatives
NP	Nondeterministic Polynomial Time
PPAD	Polynomial Parity Arguments on Directed graphs
PSPACE	Polynomial Space
QBF	Quantified Boolean Formula
TQBF	True Quantified Boolean Formula

# Chapter 1

## Introduction

The relevance of game theory in computing continues to grow. Somewhat surprisingly, board games provide intuitive vehicles for characterizing and analysing aspects at the core of theoretical computer science. Due to the inherent competition in games, study of these topics can be deceptively entertaining; even the tedium of scientific progress can be disguised as an enjoyable undertaking.

Moreover, the central topic of computational complexity is perhaps nowhere more obvious than in the realm of combinatorial games. Even those unschooled in computer science—much less in complexity theory—understand that the player who knows the game better will have a greater chance to win. It is not prudent for someone to attempt to predict all the possible moves in advance. Instead, one must try to figure out, from the rules of the game, whether there is a trick to learning easily how best to play.

For example, a novice nim player will have trouble against someone who knows how to manipulate numbers and quickly evaluate game boards. “This game is hard!” they proclaim, after losing again. “No, Nim is actually very easy,” their opponent assures them, before giving away the trick<sup>1</sup>. This is the same language we use in complexity: games which are easy have some algorithmic trick providing fast evaluation, while games that are hard appear not to have any such shortcuts.

---

<sup>1</sup>Just xor the sizes of the piles together. If the result is zero, then no winning strategy exists from this board. Otherwise, a winning strategy does exist!

On the other hand, this study has a reversed perspective towards these hardness results. For our two players, once the trick is revealed, they may never again play the game of Nim together, as there is no longer any suspense in the game play. Usually, the existence of an efficient algorithm is a positive result for a problem. Instead, the more interesting games for competitive play are those without an efficient trick. Sometimes this means that we don't yet know enough about the game and sometimes this means we have proven a hardness result. Such a proof of, say, *PSPACE*-hardness means not only that no trick is known, but that such a shortcut for one hard game translates into a shortcut for many! Thus, these tricks are unlikely to exist; hard games may require exponential-sized calculations to determine a winner.

This is the affirmation which gives humans a chance against computer players. Although supercomputers can evaluate far more Chess moves per second than the best human player, they cannot always find the *best* move. Thus, humans sometimes beat the machines. It is the hardness of games which reveals this strategic difference. The interesting aspect of the competition is the difference in evaluation patterns between humans and machines, instead of the base calculation speed.

Aside from this intuitive hardness versus easiness in games, there lies a true relationship to computational problems. Unlike Chess, which is *EXPTIME*-complete, our games are all in *PSPACE*, and many have potential for *PSPACE*-completeness. Aside from games, this class is indicative of many other computational concerns. Problems in the realms of alternation, deadlocks, periodic optimization and interactive protocols are often complete for *PSPACE*. These concerns beg a more thorough study of the class. Thus, as we explore these games, we will often hope to push the boundary of completeness. Which properties make a game complete? What changes to the game can we make to either push it into or out of completeness?

Our first such attempt in this dissertation hinges on our variants of the game Atropos. By changing the rules on where players are allowed to make their move, the game became more difficult, and we proved that it is *PSPACE*-complete. From here, we considered

looking at other hard games. What if we apply the same movement restrictions to a well-known *PSPACE*-complete game such as Hex? It turns out that this does not change the complexity of the game in general<sup>2</sup>. Despite the similarities between Atropos and Hex, these movement changes may not have the same or opposite effects.

In this work, we focus on the realm of two-player impartial games. An impartial game is one in which for any game board (state), all players have the same possible moves on their turn [4]. The focus of our work is the analysis of three impartial games: Matchmaker, Atropos and Dictator, all of which were invented in the course of this research. The first of these, Matchmaker, is based on the Stable Marriage problem. The players act as matchmakers themselves, taking turns pairing candidates from two separate groups until the system reaches equilibrium. We define the winner to be the player who made the last move.

We also discuss the “fixed-point” game Atropos. This game is inspired by Sperner’s Lemma, to the point that the game board begins as an incomplete instance of the two-dimensional version of the lemma. In this game, players take turns painting uncolored nodes of a triangular array, choosing from either red, blue or green. As soon as a player creates a triangle with one vertex painted each of the three colors, that player loses. As with Matchmaker, we define multiple variants of Atropos.

The third game, Dictator, is derived from Arrow’s Theorem concerning fairness in voting. In this game, players alternate filling empty slots in a list of ballots to be read by a social choice function. A player loses when they either demonstrate that a dictator exists in the system or find that one of Arrow’s conditions for fairness have been violated. This game provides the extra difficulty of proving that the opposing player has made an error along the way, giving it an added dimension of difficulty not found in our other games.

The sections of this dissertation are laid out in terms of different aspects of games in general. First, Chapter 2 offers a reference list of notation and definitions. Then, in Chapter 3, we discuss the creation of different games. We follow this by presenting results based on

---

<sup>2</sup>Reisch’s reduction to Hex creates boards with no neighboring uncolored hexagons, thus movement is already unrestricted by our Atropos rules and the same reduction applies[16].

the analysis of games and finding game strategies in Chapter 4. This chapter is based on provable findings for games. The next two chapters handle actual implementations for the games. Chapter 5 covers implementing code to test and analyze games, while Chapter 6 describes hurdles in designing actual playable versions of games. This chapter covers both physical and software prototypes and versions of our games.

After these technical discussions, we make summarize our contributions in two separate chapters. First, we make the point of discussing the pedagogical uses of our games in Chapter 7. Following this, Chapter 8 relates our other contributions and conclusions. Next, Chapter 9 and lists a number of open problems and questions stemming from the work accomplished here. Some of these topics are partially-researched from the work in this thesis while others are questions that naturally arise from these new games.

Finally, I wish the reader enjoyment in this exploration of games. Although this material is often difficult to experiment with in the absence of opponents, this has been an extremely fun thesis topic and it is the greatest wish that the inherent fun of studying board games carries over to those who partake in learning about or playing any variant of Atropos, Matchmaker or Dictator. Please, enjoy!

## Chapter 2

# Notation and Definitions

In this chapter, we list different notation used throughout this dissertation. Readers may use this as both a reference tool for looking up background terms used within as well as a cheat sheet for locating descriptions of games defined in later chapters.

### 2.1 Basic Definitions

In this section we list definitions used by this dissertation. Many of these definitions are specific for discussing combinatorial game theory, while others are simply mathematical tools we will use.

**Definition 2.1.1 (Label Set)**  $[n] = \{1, \dots, n\}$  also known as the *Label Set of size  $n$* .

Although we will not often refer to label sets as such, we find that the notation is extremely useful.

**Definition 2.1.2 (Combinatorial Game [1])** A combinatorial game,  $G = \{L|R\}$ , is a pair of lists of combinatorial games,  $L$  and  $R$ , such that the “left player” may move to any game in  $L$  and the “right player” may move to any game in  $R$ .

For any game state, we want to define the children of that state:

**Definition 2.1.3 (Child)** Game  $G'$  is a child of game  $G = \{L|R\}$  if  $G' \in L$  or  $G' \in R$ .

Impartial games are games in which, intuitively, there is no identification between the players in the game state. All of the games studied in this dissertation are impartial games.

**Definition 2.1.4 (Impartial Game [4])** A game,  $G = \{L|R\}$  is impartial if  $L = R$ .

In order to evaluate these games, we will need to use the Minimum Excluded number:

**Definition 2.1.5 (*mex*)**  $mex(S) = \min((\mathbb{N} \cup \{0\}) \setminus S)$

From here we can do Sprague-Grundy evaluation to determine the value of an impartial game.

**Definition 2.1.6 ( $\mathcal{G}$ [20] [12])**

$$\mathcal{G}(G) = \begin{cases} 0, & G \text{ has no children and is a loss.} \\ 1, & G \text{ has no children and is a win.} \\ mex(\{\mathcal{G}(G') \mid G' \text{ is a child of } G\}), & \text{otherwise} \end{cases}$$

**Definition 2.1.7 (Nimber)** The number of a game,  $G$  is the natural number  $\mathcal{G}(G)$ .

In one of our games, we will need to discuss painting circles one of three different colors (or leaving them unpainted). Soemtimes we will talk about the actual colors (red, green and blue), but other times we use three different symbols: bars, shading and filled-in, as demonstrated in figure 2.1. We will often use the verbs *barring*, *shading* and *filling* to describe the action of assigning the relative symbol to a node. These symbols both allow us to be flexible with the notation (they can represent any permutation of the three colors) and also provide a clear black-and-white method for describing colors of board situations.



Figure 2.1: From left to right: A barred circle, a shaded circle and a filled circle.



## 2.2 Definitions of New Games

In this section, we list games invented in this work. We do not provide full definitions here, but reference where they may be found in coming chapters.

### 2.2.1 Atropos

We define two Atropos variants in Chapter 3. Although we consider a Three-Player version in Section 3.2.1, this is not a formal definition (mostly as this is a study narrowed to two-player games). Instead, in Definition 3.2.2, we present our first game: Unrestricted Atropos. Following this, Definition 3.2.3 reveals the rules for the standard version of Atropos.

Due to the popularity and interest in Atropos, we have also developed a few randomized versions of the game. We define a die-based version in Definition 6.1.1 and an engaging variant using cards in Definition 6.1.2. For more than two players, the card version in 6.1.3 works very well. When people ask to play with me, they often would rather play one of these versions!

### 2.2.2 Matchmaker

Three versions of Matchmaker games are available in the same chapter. First, we describe Full Matchmaker in Definition 3.3.1. After this, we discover our basic game of Matchmaker in Definition 3.3.2. Finally, Definition 3.3.3 gives us Stubborn Matchmaker, the most-studied of our variants.

### 2.2.3 Dictator

From Arrow's Theorem, we create two versions of Dictator. The game Basic Dictator, described in Definition 3.4.7, gives us most of the rules and the game board. Changing the rules for proof construction in Definition 3.4.15 yields Tree-Proof Dictator, which prevents players from invoking Arrow's Theorem for their proofs. Since proofs for this version become too unwieldy, however, we introduce a Chain-Proof variant of the game in Definition 3.4.16

which winds up being our final version.

## Chapter 3

# Game Creation & Design

In this chapter, we describe the invention of games which occurred during the research for this thesis. Fundamentally, the inspiration for each of these games arises from a variety of interdisciplinary areas related to computer science. From this inception, we test and tweak the game design until we reach an acceptable format. Often, we are looking to conform to the adage, “easy to play, hard to master”. In order to do this, we will seek to revise the game until we find a version where strategies are computationally hard to determine. As an example, we look at the classic game of Hex, a two-player game played on a hexagonal grid (see Figure 3.1). Hex has been shown to be *PSPACE*-complete [16].

### 3.1 Hex

Indeed, the design and rules of Hex are simple enough that it was independently invented both by mathematician and poet Piet Hein (1942) and mathematician and economist John Nash (1947). The empty Hex board is exactly the hexagonal grid shown in Figure 3.1 with the four sides colored: two blue sides opposite each other, and the other two red. Initially all hexagons are uncolored. The players are named “Red” and “Blue” and on their turn, that player paints one of the uncolored hexagons their own color. The goal for both players is to form a connected path of their-color hexagons bridging their two sides.

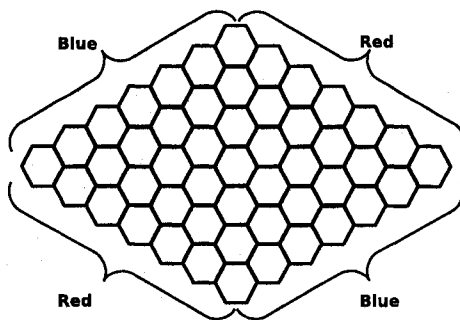


Figure 3.1: An Empty Hex Board.

While these above rules are extremely simple—anyone can learn to make legal moves instantly—strategically playing the game requires a great amount of practice.

One of the elegant aspects of Hex is that the game can never end in a draw. Nash initially proved this when he developed the game. This “Hex Theorem”—that exactly one of the two players will have won at the end—has even been shown to be equivalent to the Brouwer Fixed-Point theorem [10]. Due to this correlation, Hex is often referred to as a fixed-point game.

### 3.2 Sperner’s Lemma

Just as the Hex Theorem is the basis of the game Hex, we use other such mathematical foundations to develop further games. Another such fact equivalent to the Brouwer Fixed-Point theorem is Sperner’s Lemma. In 1928, mathematician Emanuel Sperner noticed a strange phenomenon in the  $(n+1)$ -coloring of an  $n$ -dimensional simplex. By enforcing a few simple rules on the coloring only of the boundary vertices, he discovered at least one cell in the simplex must have a vertex of each color, no matter how the colors on the interior are chosen [19]. The two-dimensional (three-color) space for this is known as Sperner’s Triangle and it is the board for our first game.

Inspired by the relationships of Hex and Sperner’s Lemma to Brouwer, Atropos uses a “hollow” Sperner Triangle as the game board. The game uses three colors: red, green and

blue, and only the boundary of the triangle begins already colored. This initial coloration is simply an enforcement of Sperner's boundary conditions, ensuring that by the time the board is entirely colored, a three-colored triangle will exist. The left-hand side of the triangle is colored in alternating green and red circles (starting at the bottom with green), the right-hand side is colored with alternating green and blue (starting at the bottom with blue), while the bottom is colored red and blue (starting on the left with red) (see Figure 3.2).

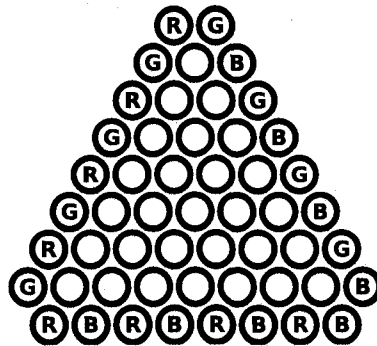


Figure 3.2: An empty Atropos Board of size seven (the number of open circles on a side).

From the starting configuration, the plan for Atropos is simple: players take turns coloring the uncolored nodes until a three-colored triangle is created, signalling the end of the game (by Sperner's Lemma, this end must be reached). Since it is relatively easy to create this triangle, we determined that the player who creates this triangle loses. Because of this, we refer to such a triangle as a "bad triangle".

**Definition 3.2.1 (Bad Triangle)** *In a Sperner Triangle, a bad triangle is an atomic triangle which has vertices colored each of the three colors.*

From this basic plan, many versions of Atropos came about before we settled on the canonical variant.

### 3.2.1 Three-Player Atropos

Blindly following the scheme of Hex, we happened upon a three-player variant of Atropos. Each player is assigned to one of the three colors: Red, Green or Blue, and on their turn, paints one of the uncolored circles with their color. By the basic rules, the player forced to create a three-colored triangle loses. The winner is then defined as the player who took their turn directly prior to the loser. The third player neither “wins” nor “loses”.

Unfortunately, this awkwardness for that third player’s outcome carries over into the actual game play. With this in mind, we move to defining variants for only two players.

### 3.2.2 Unrestricted Atropos

In order to change the scheme to fit two players, it no longer makes sense to assign each player to a color. Instead we allow both players to choose from all colors on each turn.

**Definition 3.2.2 (Unrestricted Atropos)** *Unrestricted Atropos is a game played between two players on an incomplete Sperner Triangle. Each turn, a player chooses one of the uncolored circles and paints it either red, green or blue. The loser of the game is the first player to create a three-colored triangle.*

Notice that this game is impartial; players have exactly the same options for playing and have the same winning criteria (see Definition 2.1.4). Additionally, the loss of the coloring-restrictions allow more freedom between the players. Unfortunately, there appear to be simple strategies for winning this game. In Section 4.2, we show some intuition arguing that from the starting configuration this game can always be won by the player who would not paint the last circle (if the whole board were to be colored). Thus, to determine whether or not you can win, all you should do is determine the parity of the number of circles in the initial triangle. If there are an even number, the first player will win. Otherwise, the second player has a winning strategy.

### 3.2.3 (Restricted) Atropos

We find that with a slight change to the rules, we can remove this potential simplicity. Instead of allowing players to paint whichever circle they desire, we enforce that they play adjacent to the last move.

**Definition 3.2.3 ((Restricted) Atropos)** *(Restricted) Atropos is exactly the same game as Unrestricted Atropos, aside from the following change of rules. The first player begins by painting whichever circle they choose. On each subsequent turn, the current player must choose an circle adjacent to the last move to paint. If no such circle exists, the current player may choose any uncolored circle on the board to paint.*

With the rules change, we finally reach our “easy to play, hard to master” version of the game. Even with this added rule, the game is still very easy to learn to play; the effect of the adjacency rule is just to reduce the number of options each player has on a turn. Instead of making the winning strategem simpler, however, this restriction actually makes it more difficult to plan ahead or interfere with your opponents’ intentions. From here on, we will often refer to this game as simply *Atropos*. In Theorem 4.1.1 we show that solving the problem of determining whether the current player has a winning strategy in *Atropos* is PSPACE-complete.

## 3.3 Stable Marriage

For the inspiration for further games, we again dip into mathematical concepts. The Stable Marriage problem [11] is the mathematical problem of finding a stable matching across two equal-sized sets of candidates. We refer to our sets as *West* and *East* so that  $|West| = |East| = n$ . Each candidate,  $w_i \in West$  has a preference list (strict ordering) on the candidates in *East* and vice versa for each candidate,  $e_i \in East$ . We treat these lists as functions from one group to  $\{1, 2, \dots, n\}$ . Thus, the preferences of  $w_i \in West$  is described by  $p_{w_i} : East \rightarrow \{1, 2, \dots, n\}$  so that if  $w_i$  prefers  $e_j$  most, then  $p_{w_i}(e_j) = 1$  and so forth, so that if  $p_{w_i}(e_k) = n$ ,  $e_k$  is the least desirable candidate for  $w_i$ .

A matching of the candidates is simply a bijection from *West* to *East*. A matching,  $M$ , is stable if  $\forall (w_i, e_k), (w_j, e_l) \in M : p_{w_i}(e_k) < p_{w_i}(e_l) \rightarrow p_{e_l}(w_i) < p_{e_l}(w_j)$  (and vice versa). Informally, a matching is stable if no two candidates both prefer each other over the candidates they are matched with.

The famous Stable Marriage Problem [11] is the problem of finding a stable matching given the two sets of candidates and their preferences. In 1962, Gale and Shapely determined that a stable marriage could always be found, and described an algorithm for finding the correct matching. In this setting, we have one authority who is performing the pairings and matching candidates to each other.

For the purposes of a game, we want to allow the players to be these authorities, or matchmakers. We allow players to change the current matching by proposing a new pair  $(w, e)$  which is legal only if  $w$  and  $e$  prefer each other more than their current partners. Whenever a legal match is proposed, the board changes in the following way. If  $(w, e)$  is the proposed pair,  $\forall (w, e')$  and  $(w', e) \in M$ , the new marriage,  $M' = M \cup \{(w, e), (w', e')\} \setminus \{(w, e'), (w', e)\}$ . We will define the game to end once the state reaches a stable marriage; no more matches can take place. The player who performs the last legal match will be the winner.

With these rules, we already have a game, though it is a very complex game. Each candidate has its own preference list, meaning that the game requires  $O(n^2)$  information before players even begin to make moves. In keeping with our desire for simple game states, we wish to remove this vast amount of data. Thus, instead of having individual preference lists, we give a universal ranking for both groups.

Now,  $\forall w_i, w_j \in West : p_{w_i} = p_{w_j}$  and  $\forall e_i, e_j \in East : p_{e_i} = p_{e_j}$ . With this universal ordering, we now refer to  $w_1$  (or, simply "1") as the most-preferred candidate in *West*,  $w_2$  (or 2) as the second-most-preferred candidate, and so forth. We do the same for the elements of *East*. With this ranking, we can now draw out a board for our matching game (see Figure 3.3).



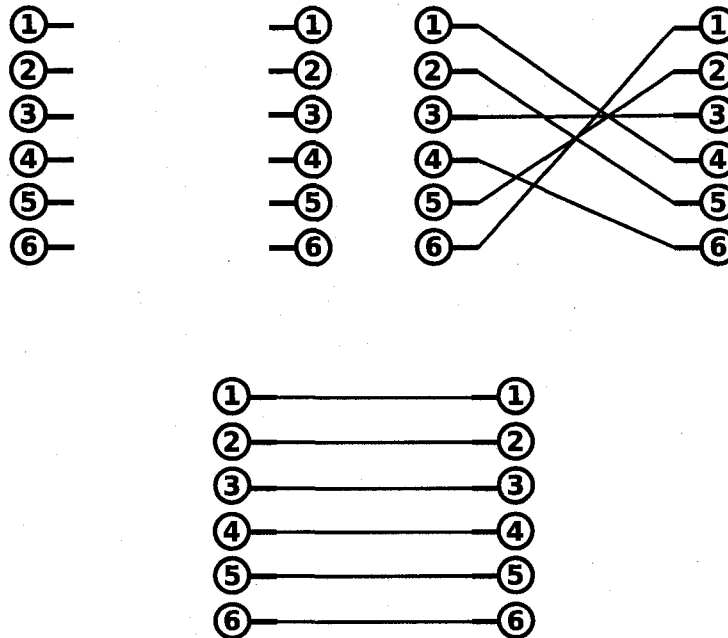


Figure 3.3: Left: empty Matchmaker board. Middle: fully-matched board. Right: the only stable Matchmaker configuration possible.

### 3.3.1 Full Matchmaker

All that remains for our game is to choose a starting configuration. We define *Full Matchmaker* to begin with the top candidate in *West* paired with the worst candidate in *East*, the second-best candidate in *West* paired with the second-worst candidate in *East* and so forth:

**Definition 3.3.1 (Full Matchmaker)** *Full Matchmaker* is the two-player game where players alternate pairing candidates from two sets,  $West = \{w_1, w_2, \dots, w_n\}$  and  $East = \{e_1, e_2, \dots, e_n\}$ . The starting position has the following matched pairs  $\forall k : (w_k, e_{n-k+1})$ . Players change the board by performing a new legal match. The first player who cannot make a match loses the game.

Unfortunately, it is *too* simple to determine who will win this game. Even worse, each turn, all moves a player can make are equivalent! To determine who will win the game,

we simply need to count the number of crosses between edges on the board at any state. If there are an odd number, the next player will win (no matter which move they make). Otherwise, the second player will win the game. We discuss a proof of this property in Lemma 4.3.1.

In order to alleviate this simplicity, we allow boards to not have all  $n$  pairs in the matchings. From this, we will define a new game.

### 3.3.2 Matchmaker

A partial matching is a bijection between subsets  $West' \subset West$  and  $East' \subset East$ . First we need to revise our definition of when a pair is legal to propose. Our clarification is simply that all candidates prefer to be matched with anyone rather than not be in the matching.

Additionally, we need to alter the rule for incorporating proposed matches: if a player proposes a pair,  $(w, e)$ , legal for the current partial match,  $M$ , the new (partial) match,  $M' = M \cup \{(w, e), (w', e')\} \setminus \{(w, e'), (w', e)\}, \forall (w, e'), (w', e) \in M$ . Now, if a pair is proposed and both candidates are not already matched, only the proposed pair is assimilated into the new matching; no pair of spurned candidates is added.

**Definition 3.3.2 (Matchmaker)** *Matchmaker is exactly the same game as Full Matchmaker, except that we use the following rule for matching candidates. After proposing a legal match,  $(w, e)$ , on a game state with (partial) matching,  $M$ , the new game state will have the match*

$$M' = M \cup \{(w, e), (w', e')\} \setminus \{(w, e'), (w', e)\}, \forall (w, e'), (w', e) \in M.$$

*The starting position for this game is the empty matching:  $M = \{\}$ .*

### 3.3.3 Stubborn Matchmaker

Alternatively, we can simplify the game a bit by restricting exactly which pairs are legal plays. We change our rule by enforcing that, for some matching  $M$ ,  $(w, e)$  is not a legal pair if, for some  $e' \in East : (w, e') \in M$  but  $\forall w' \in West : (w', e) \notin M$  (or vice versa).

Now a legal pair must consist of either two unmatched candidates, or two already-matched candidates.

**Definition 3.3.3 (Stubborn Matchmaker)** *Stubborn Matchmaker is the same game as Matchmaker, except that it uses the following rule to determine whether a proposed pair is legal: for some matching  $M$ ,  $(w, e)$  is not a legal pair if, for some  $e' \in \text{East} : (w, e') \in M$  but  $\forall w' \in \text{West} : (w', e) \notin M$  (or vice versa).*

We have analyzed many game situations for both Matchmaker and Stubborn Matchmaker. We describe this analysis in Section 5.3.

### 3.4 Arrow's Theorem

Another bit of inspiration from economics leads us to a final game. This game is based on voting systems and has rules in place as a reference to Arrow's (Impossibility) Theorem for social choice functions[3]. In this schema, we have a function which takes as input a list of ballots, each of which are complete preference lists on a set of candidates. This social choice function then returns a separate preference list on the candidates, revealing the outcome of the election.

For our purposes, we use the following definitions:

**Definition 3.4.1 (Preference List)** *A preference list on a set (of candidates)  $S$  is an ordered tuple where each entry is a unique element of  $S$ . If element  $A$  is in list  $L$  before element  $B$ , we say that  $A$  is preferred to  $B$  (or  $A > B$ ) in  $L$ .*

**Definition 3.4.2 (Social Choice Function)** *A social choice function on a set candidates  $S$  is a function which takes preference lists on  $S$  as inputs and returns a preference list also on  $S$ .*

**Definition 3.4.3 (Pareto Efficiency)** *A social choice function,  $F$  on  $S$ , is pareto efficient if for any two candidates  $A \in S$  and  $B \in S$  and any preference lists  $L_1, \dots, L_n$  on  $S$ , where  $\forall i : A > B$  in  $L_i$ , then  $A > B$  in  $F(L_1, \dots, L_n)$ .*

**Definition 3.4.4 (Dictatorship)** *A social choice function,  $F$ , is a dictatorship if  $\exists i \in [n]$  such that  $\forall L_1, \dots, L_n : F(L_1, \dots, L_n) = L_i$ .*

**Definition 3.4.5 (Independence of Irrelevant Alternatives (IIA))** *A social choice function,  $F$ , satisfies independence of irrelevant alternatives (IIA) if  $\forall A, B, Y_1, \dots, Y_n, Z_1, \dots, Z_n : \text{if } \forall i : A \text{ and } B \text{ have the same relationship in } Y_i \text{ and } Z_i, \text{ then } A \text{ and } B \text{ have the same relationship in } F(Y_1, \dots, Y_n) \text{ and } F(Z_1, \dots, Z_n)$ .*

From these definitions, we have the following surprising result of Arrow:

**Theorem 3.4.6 (Arrow's Theorem[3])** *There does not exist a social choice function  $F$  which is pareto efficient, satisfies IIA, and also not a dictatorship.*

The surprising facet of this theorem is that although both Pareto efficiency and IIA seem to be reasonable desires for a choice function, any voting system that utilizes them must also be dictatorship! A constructive version of the proof shows exactly how to find the dictator by testing only  $n$  different lists of ballots.

The proof is somewhat difficult to understand; we design an impartial game to help explain the effects of IIA and Pareto efficiency that force the dictatorship.

**Definition 3.4.7 (Basic Dictator)** *Basic Dictator is a game played on a partially-filled ballot list,  $X_1, \dots, X_m$ , where each ballot has  $n$  entries. The initial state is a list of empty ballots (see Figure 3.4) while any state is legal if none of the ballots have exactly one un-filled slot. The candidate set is  $[n]$ , though often if  $n \leq 26$ , we use  $A, B, \dots, n^{\text{th}}$  letter of the English alphabet. We assume that there is some social choice function  $F$ , such that  $F(X_1, \dots, X_m) = (1, 2, 3, \dots, n)$  and that  $F$  satisfies IIA.*

*There are two phases to each turn:*

- *Proving  $F$  Is Not Pareto—The current player may attempt to prove that  $F$  cannot satisfy Pareto efficiency. If the current player successfully proves this, they win. The player must show that Pareto efficiency is violated no matter how the rest of the ballots are filled out.*

- *Filling a Slot*—The current player chooses one of the unfilled slots in a ballot and a candidate not yet listed in that ballot. That slot is then filled with that candidate. If only one more slot remains unfilled in that ballot, it is also filled with the remaining candidate not yet in the ballot. If this creates a ballot equal to  $(1, 2, 3, \dots, n)$  (a possible dictator) then the current player loses.

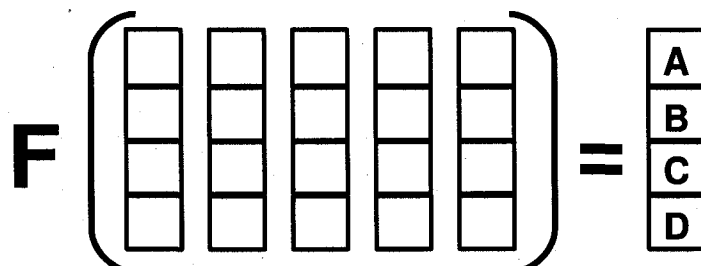


Figure 3.4: An empty Dictator board, where  $n = 4$  and  $m = 5$ .

Thus the goal is to avoid either creating a “dictator” or violating Pareto efficiency. Also, a good player will be able to see when boards violate this Pareto property. For example, in Figure 3.5 we have a board which violates Pareto. Even though not all the slots are filled in, we can see that  $D > B$  in all ballots. Thus, since  $F(X_1, \dots, X_5)$  has  $B > D$ ,  $F$  is not Pareto efficient. The next player may use this as their proof to win the game.

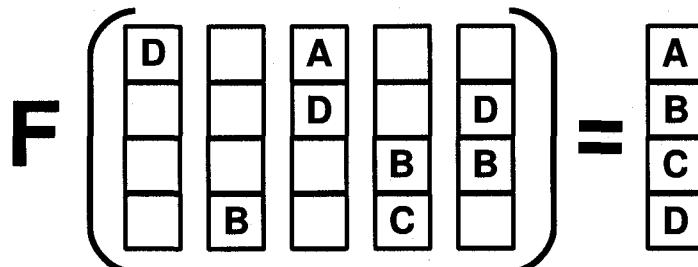


Figure 3.5: This board violates Pareto efficiency; in all ballots,  $D > B$ .

In fact, using IIA, we can prove even more difficult violations. If we change the board from Figure 3.5 by just swapping the two entries in the fifth ballot (see Figure 3.6), we find still another violating board. Here our proof is a bit more difficult.

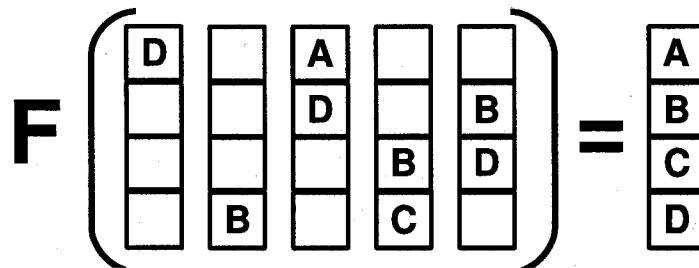


Figure 3.6: This board also violates Pareto.

**Lemma 3.4.8** *The social-choice function in Figure 3.6 violates Pareto efficiency.*

**Proof.** We must show that  $F$  which is consistent with this board violates Pareto. This uses a similar strategy as used in proofs of Arrow's Theorem.

Let  $X = (X_1, \dots, X_5)$  be a list of our ballots. Let  $Y$  be equal to  $X$  except that the  $D$  and  $B$  are switched (this is exactly the ballot list shown in Figure 3.5). Since IIA holds for our  $F$ , and  $D$  and  $B$  are adjacent to each other in that last ballot,  $F(X)$  and  $F(Y)$  may differ in their relationship between  $D$  and  $B$ , but not with any other candidates. Thus, neither may change their relationship with  $C$ , which is between them. Thus,  $F(Y)$  must equal  $F(X)$ . However, by Pareto (as we showed for the previous example)  $D > B$  in all the ballots in  $Y$  and thus  $F(Y)$  should rank  $D > B$ .

Instead,  $F(Y)$  ranks  $B > D$ . Thus,  $F$  does not satisfy Pareto.  $\square$

We can have even more difficult boards from which we can prove a violation. A further example is given in Figure 3.7.

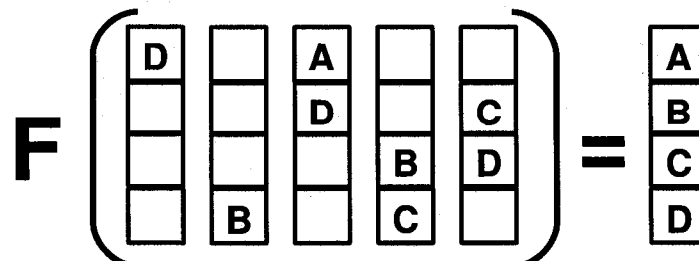


Figure 3.7: We can use IIA to show that this board violates Pareto.

**Lemma 3.4.9** *The social-choice function in Figure 3.7 violates Pareto efficiency.*

**Proof.** We must show that  $F$  violates Pareto. This time we will use IIA on three candidates ( $A$ ,  $C$ , and  $D$ ) and make more changes to create a board which is a clear violation.

Using IIA on  $A$ ,  $C$  and  $D$ , we can rearrange the board in Figure 3.7 to the board in Figure 3.8. For most of the ballots, this transformation is straightforward: the first ballot does not change, and  $A$  and  $D$  are definitely adjacent in the third and fourth ballots (and can thus be reordered by IIA). In the second ballot, all three are above  $B$ , so we can just put  $D$  up top and  $A$  beneath. In the last, since  $C$  and  $D$  are in the middle,  $A$  must be adjacent to one of them. Thus, all of  $A$ ,  $C$  and  $D$  are adjacent, and we can move them to put  $D$  above  $A$  in the middle, moving  $C$  either up top or to the bottom (though we leave it out instead of diagramming both cases). Now, with our reorganized board,  $D > A$ . Since  $D$  cannot have changed relationship to  $B$  in the result, it cannot have risen above  $A$  (it could have swapped with  $C$ , and thus the bottom of the result list is not shown in our example). Thus we have shown a violation of Pareto.  $\square$

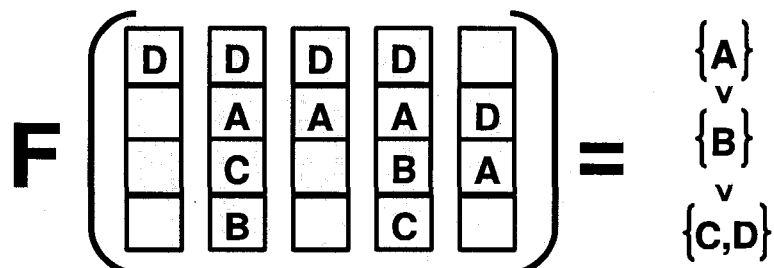


Figure 3.8: This board violates Pareto:  $D > A$  in all ballots.

As mentioned, these proofs use basic manipulations of IIA to find Pareto violations. Any logic followed like this teaches the player the mechanics of Arrow's Theorem. Unfortunately, there is another, simpler, method of proving the above violation. Instead of actually toying with our voting rules, we can just invoke Arrow's Theorem. Since none of the ballots in our list can be the dictator (they all have at least one candidate out of place) by Arrow, it must be that any IIA-consistent  $F$  does not satisfy Pareto. (We will prove more about

using Arrow's Theorem in Section 4.4.) Since this realization saps most of the fun from the game, we alter the rules slightly to produce a new version of Dictator.

The big change for this new game is that we will clearly define what steps players may use to prove Pareto Violations. This requires a flurry of new definitions.

The first of these deals with the result of using IIA on a game board. Since the voting function's result will become blurry if IIA is used on neighboring candidates (as with the manipulation in Figure 3.8, IIA was used on  $C$  and  $D$ , so their relationship could have changed in the result) we need a way to express this as something that still resembles a preference list.

**Definition 3.4.10 (Partial Preference List)** *A partial preference list on candidate set  $S = \{X_1, \dots, X_n\}$  is an ordered tuple of disjoint subsets  $S_1, \dots, S_m$  of  $S$  which cover  $S$ . If subset  $S_i$  is in partial preference list  $L$  before subset  $S_j$  then for any candidate  $A \in S_i$  and  $B \in S_j$ , we say that  $A$  is preferred to  $B$  (or  $A > B$ ) in  $L$ .*

We will use the following as an intermediate Dictator "board" in a step of proving a Pareto Violation.

**Definition 3.4.11 (Proof Board)** *A proof board,  $P$ , on candidate set  $S$  is a list of partially-filled preference lists on  $S$  (the "ballots" of  $P$ ) and a partial preference list on  $S$  (the "result" of  $P$ ).*

The ballots of a proof board,  $P$ , are just the input to the social choice function and the result describes some of the limits on the output of  $P$  on those ballots.

We would like an explicit way to talk about the effect IIA has on proof terms. First, we will look at the effect it has on partial preference lists. The idea here is that if we use IIA on candidates which are neighbors in a partial preference list, then we are no longer certain about the relationship between those candidates and any other candidates in their sets. For example, if we use IIA on  $B, D, F$  and the result of the current proof board is  $(\{A\}, \{B, C\}, \{D\}, \{E\}, \{F\})$  then this should yield a new board where the result is  $(\{A\}, \{B, C, D\}, \{E\}, \{F\})$ . We describe this as "shuffling".



**Definition 3.4.12 (Shuffling)** For any partial preference list,  $L = (S_1, \dots, S_m)$ , the shuffling of  $L$  on  $C \subset S$  is the following partial preference list:  $(T_{a(1)}, \dots, T_{a(k)})$  where  $T_i$  is the smallest set which contains all the elements of  $S_i$  and, if  $\exists X, Y \in C$  where  $X \in S_i$  and  $Y \in S_{i+1}$ , then  $T_i = T_{i+1}$  and  $a(i) = j$  means that  $T_i$  is the  $j^{\text{th}}$  unique candidate set in the list  $T_1, \dots, T_m$ .

Thus, in our language, we can express the above example by saying: the shuffling of  $(\{A\}, \{B, C\}, \{D\}, \{E\}, \{F\})$  on  $\{C, D, F\}$  is  $(\{A\}, \{B, C, D\}, \{E\}, \{F\})$ . This allows us to present a clear definition of the effects of an IIA-manipulation in our proofs.

**Definition 3.4.13 (IIA-Manipulation)** Proof board  $P'$  is an IIA-manipulation on a subset,  $C$ , of the candidate set,  $S$ , of proof board  $P$  if the result of  $P'$  is the shuffling of the result of  $P$  on  $C$  and the ballots of  $P$  and  $P'$  differ only in their relationships of candidates in  $C$ .

IIA-manipulations will be one of the options players have for each step when constructing Pareto Violations. We will give them only one other.

**Definition 3.4.14 (Candidate Case)** Proof board  $P'$  is a candidate case at  $(i, j)$  of  $P$  if  $P$  and  $P'$  have the same result list and differ in their ballots only in the  $i^{\text{th}}$  entry of the  $j^{\text{th}}$  ballot in the following way:  $P$  has no candidate listed there while  $P'$  does.

We show an example of candidate cases in Figure 3.9. Notice that a proof board  $P$  has a Pareto violation exactly if  $\exists(i, j)$  such that all candidate cases of  $P$  at  $(i, j)$  have Pareto violations. While attempting to show a violation, we allow players to build a proof tree at each step, extending their case by using an IIA manipulation or splitting into multiple threads by examining all possible candidate cases at one location. Now we provide a new definition:

**Definition 3.4.15 (Tree-Proof Dictator)** Tree-Proof Dictator has the same rules as Basic Dictator (Definition 3.4.7), except that we replace the first phase of the turn with the following:

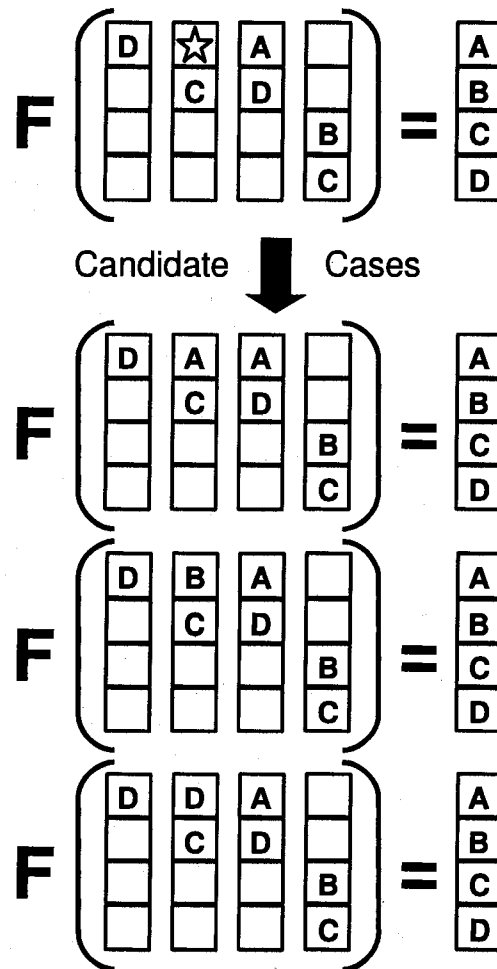


Figure 3.9: The three lower boards are the candidate cases of the top board at the entry marked with a star.

- *Proving  $F$  Is Not Pareto*—The current player may prove that  $F$  cannot satisfy Pareto efficiency. The current player may do this by either showing a Pareto violation on the current board,  $P$  or recursively finding: either a Pareto violation on an IIA manipulation of  $P$  on any subset or some  $i, j$  where all candidate cases of  $P$  at  $(i, j)$  violate Pareto.

This game has the interesting property that a player can successfully challenge exactly when they could successfully challenge in Basic Dictator; any violation of Pareto Efficiency can be shown using a combination of the two proof steps defined above. On the other hand,

since a player cannot just invoke Arrow's Theorem, it's not clear whether a proof can be found or shown efficiently. This results in players spending a lot of time trying to find a correct proof, since they can be certain one exists. Furthermore, the proof structure of a tree is not very compatible with casual games; it is easy for one player to describe a long proof and lose the other player. We would like proofs to be easier to verify.

For this reason, we remove the candidate case option for proofs, giving us our final version of the game:

**Definition 3.4.16 ((Chain-Proof) Dictator)** *(Chain-Proof) Dictator is the same game as Basic Dictator (Definition 3.4.7), except that we replace the first phase of the turn with the following:*

- *Proving  $F$  Is Not Pareto—The current player may attempt to show that  $F$  cannot satisfy Pareto efficiency. The current player may do this by either showing a Pareto violation on the current board,  $P$  or recursively finding a Pareto violation on an IIA manipulation of  $P$  on any subset.*

Now proofs must have a linear structure; there is no splitting up into various cases. This has two effects on the game play. First, it makes the verification of proofs simpler. The verifying player needs only to follow each consecutive step and not to recall parts of previous steps or keep track of cases yet to be proven. Furthermore, since each step is just one IIA manipulation, this process quickly becomes easy to follow after just a little practice.

Second, it is no longer immediately clear when a Pareto violation can be shown. As we will show next, there are situations where a Pareto violation exists but cannot be shown using these proof rules. Thus, the game play does not have all the difficulty in the proving step. Instead the difficulty lies in both parts, which we will talk about further in the next chapter.

All of the examples in Figures 3.5, 3.6 and 3.7 show boards upon which this proof method can be used. We now seek a board which cannot have a dictator, but which does

not have chain of IIA-manipulations to create a violation. One example can be studied in Figure 3.10. Here we have a board which, by Arrow's Theorem, must violate Pareto, but would *need* to use candidate cases.

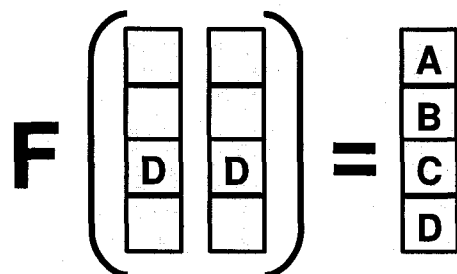


Figure 3.10: This board cannot have a dictator, but we cannot show a Pareto violation in Dictator.

**Lemma 3.4.17** *We cannot use only IIA-manipulations to prove a Pareto Violation in Figure 3.10.*

**Proof.** In order to show a violation, we would have to use IIA on up to three candidates who are not all adjacent in the result list. Thus, in order to force the inconsistency, at least either  $B$  or  $C$  could not be moved by applications of IIA. So, we are allowed to change the board by shifting a subset of either  $\{A, B, D\}$  or  $\{A, C, D\}$ . We will show that for each of these choices, there is an actual ballot configuration where this subset alone cannot be used to violate Pareto.

- Case 1: using a subset of  $\{A, B, D\}$ . Since we are not going to move the candidate  $C$ , the result list must have the form  $(-, -, C, D)$  (thus it will not suffice to switch only  $A$  and  $B$ ). As a counterexample, it could be then that the ballots are filled in in the following way:  $(A, C, D, B); (B, C, D, A)$  (see Figure 3.11). First notice that without performing any IIA-manipulations, Pareto Efficiency still holds. Next note that by swapping only  $A, B$  or  $D$ , we cannot forcibly change the relationship between  $D$  and any of the other candidates in any useful way.  $D$  is stuck below  $C$ , and since  $A$  and  $B$  are each stuck on top on one ballot, there is no way to move either of them below

either  $C$  or  $D$ . Thus, there is an example where first using any IIA-manipulation on a subset of  $\{A, B, D\}$  gets us nowhere.

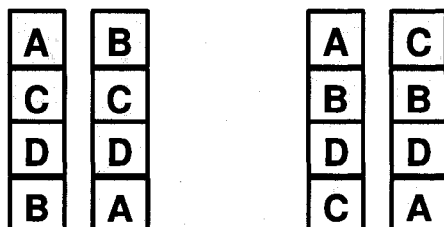


Figure 3.11: Left: counterexample for using  $\{A, B, D\}$  transformations. Right: counterexample for  $\{A, C, D\}$ .

- Case 2: using  $\{A, C, D\}$ . Since we are not going to move the candidate  $C$ , the result list must have the form  $(A, B, -, -)$ . The counterexample for this case is similar to the previous, we simply produce the ballots:  $(A, B, D, C); (C, B, D, A)$  (see Figure 3.11). We again have the same problem: Pareto is initially satisfied and we cannot move either  $C$  or  $D$  above  $A$  or  $B$  in all ballots;  $A$  and  $B$  are stuck up top in the first ballot. Thus, using an IIA-manipulation on  $\{A, C, D\}$  first won't get us anywhere.

□

Thus, this version of our game does differ from previous versions of Dictator: in some situations not having the possibility of a dictator does not mean a successful Pareto challenge can be made.

At the same time, the notion of allowing a sequence of IIA-manipulations becomes extremely vital, both for the reasons that it is only a sequence (and not a complete case-analysis tree) and that the prover can use more than just a single IIA-manipulation. Indeed, even some completely-filled boards require multiple manipulations to find a Pareto violation. We demonstrate one of these boards in Figure 3.12, and show that the board requires more than one invocation of the IIA.

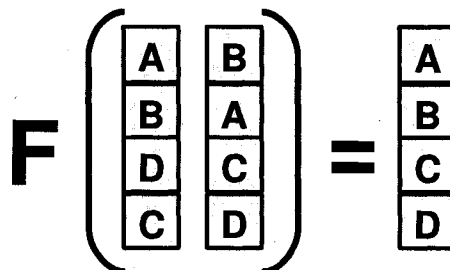


Figure 3.12: We need more than one IIA-manipulation on this board to find a violation.

**Lemma 3.4.18** *We cannot prove a Pareto violation in the board in Figure 3.10 using only one IIA-manipulation.*

**Proof.**

We will prove that we need more than one IIA-manipulation to demonstrate a Pareto violation. To do this, we will show that neither a one-shot manipulation on a subset of  $\{A, B, D\}$  nor  $\{A, C, D\}$  suffices to find a violation. Note that we have not already violated Pareto without the help of any manipulations.

First we will attempt to use an IIA-manipulation on a subset of  $\{A, B, D\}$ . Here, we want to somehow describe a situation in which either  $D > B$  in both ballots or  $D > A$  in both ballots. We cannot rely on  $B > A$ , since the result will have  $A$  and  $B$  in the same set. Using IIA on  $A, B$  and  $D$ , however, we cannot force  $D$  above either of the other two, as in the second ballot it is trapped beneath  $C$ .

Instead, we can try a manipulation on a subset of  $\{A, C, D\}$ . Here, since  $C$  and  $D$  are neighboring in the result, we need to create a violation by causing either  $C > A$  in both ballots, or  $D > A$  in both.

Due to the first ballot, however, with  $A$  trapped directly above  $B$ , we cannot now change its relationship to either of the other two candidates we have chosen to use IIA with. Hence, we cannot force either of the two relationships with  $A$  that we would like.

Thus, we cannot find the Pareto violation in only one step.  $\square$

We find it valuable here to present a working sequence of IIA-manipulations to find a

violation for the example in Figure 3.12. We will use Figure 3.13 as a guide.

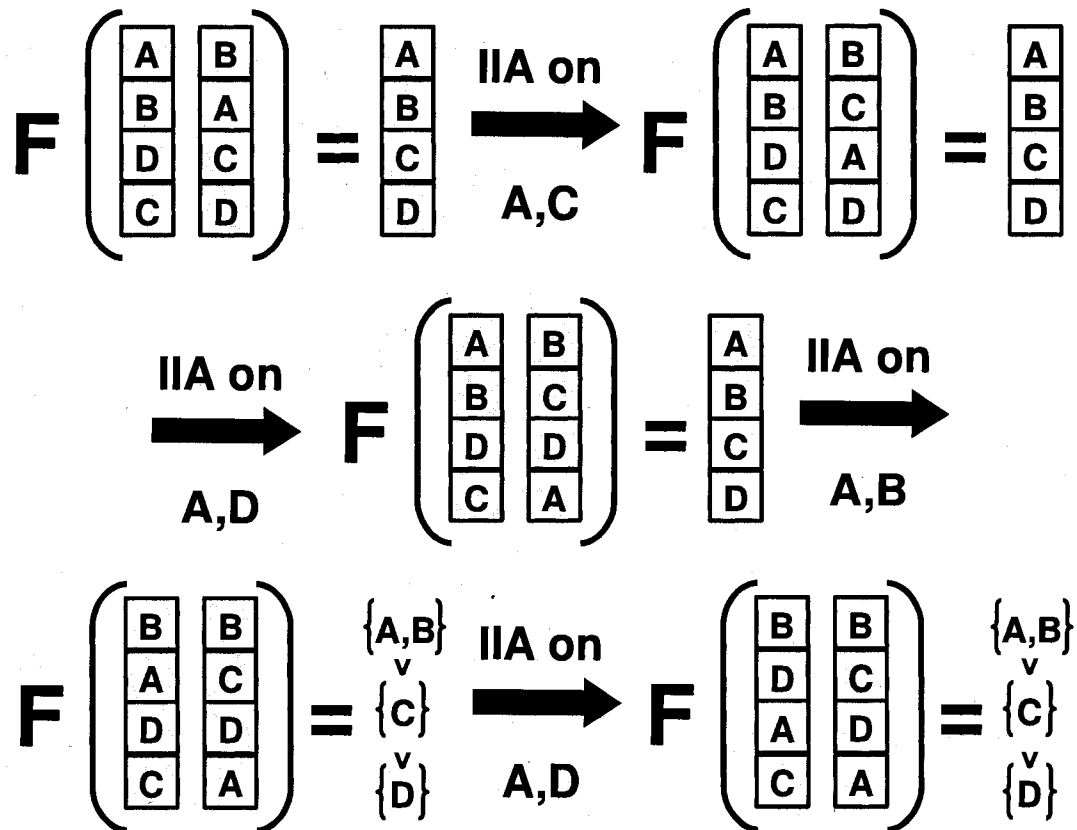


Figure 3.13: A sequence of IIA-manipulations leading to a Pareto violation.

The diagrammed process uses four steps, each time operating on two candidates. The end result of these steps is to force  $D > A$  in all the ballots, while keeping the relationship  $A > D$  in the result. Our sequence of IIA-manipulations (on  $\{A, C\}$ ,  $\{A, D\}$ ,  $\{A, B\}$  and  $\{A, D\}$  respectively maintains this relationship.

### 3.5 Summary

Our method for game design here is well carried out. All three of our games went through major revisions with the goal of reaching a completeness result while maintaining both the initial character of the game as well as the ease and elegance of the rules.

With Atropos we have had great success. With our second real version of the game, we had the *PSPACE*-hardness we desire as well as the basic characteristics of Sperner's lemma still built in. Matchmaker, on the other hand, has endured a number of revisions and a great deal of analysis without finding a hard version. Although the complexity of even Stubborn Matchmaker remains unknown, enough patterns have emerged that polynomial-time evaluation seems very probable. Still, our design cycle has not given up on Matchmaker or its variants, and there is always the possibility that it will limbo with unresolved complexity for a while (such as with Sprouts and Chomp where analysis is on-going[2][21]).

Dictator, while being a very young game, has already been through a few loops of the design cycle. We have made improvements to the game, both in the legitimacy of the proof structure and in the complexity of the game. It is because of these redesigns that we are able to phrase both of the complexity questions.

The design process has also actually simplified the Dictator rules. Avoiding the ability to use candidate cases in violation proofs removes a great deal of confusion during game play. This little tweak to the rules has actually been a great help for the game, and it came about due to complexity desires.

Thus, although this design cycle was used here without any formal game-design training, the results have been excellent. One of our games has produced a final variant, while the other two have undergone rigorous testing and are definitely in a playable state, awaiting either to be shown to be hard, or the next round of improvement.



## Chapter 4

# Computational Complexity of Games

Although we wish for our games to be easy to play, in the sense that players should know which moves are available to them, “better” games are often those which are a challenge to play *well*. Even though players should know how they can move, it should be difficult to determine how they *should* move on a given game board.

We can phrase this as a computational question: Given some game,  $G$ , what is the “best” move we can make? Meaning, which of the children of  $G$  (say,  $G_1, G_2, \dots, G_k$ ) should we move to? Here, the “best” move refers to any move which will win the game for the current player, no matter what action the opposing player takes.

If the number of children,  $k$  of  $G$  is polynomial in the size of  $G$ , then the hope to answering this question in polynomial time lies in determining whether any of the children,  $G_1, \dots, G_k$  is a losing position. In this case—all our games have a polynomial amount of children—we can rephrase our computational problem as, simply, “Given some game,  $G$ , is  $G$  in a winnable state?”

The brute force approach to solving this problem is to simply compute the winnability of the children. This, in turn, leads to the computation of the grandchildren, etc. In many cases, this leads to an evaluation of an exponential number of descendants. Thus, we want

to know when a trick exists for swift evaluation of a game. In this section, we will resolve the computational complexity of some of our games, either by finding a trick for evaluating them in polynomial time, or by showing a completeness result for that game.

## 4.1 Atropos

We find the following result for Atropos:

**Theorem 4.1.1** *Determining whether the current player has a winning strategy in Atropos is PSPACE-complete.*[5].

In order to show this, we need to prove both that Atropos is in *PSPACE* and that it is *PSPACE*-hard.

**Lemma 4.1.2** *Determining whether winning strategies exist for Atropos is in PSPACE.* [5].

**Proof.**

Since the number of plays is at most the number of nodes in the gameboard, the depth of every branch of the game tree is linear in the size of the input. Thus, in polynomial space we can determine the result of following one path of the game tree. In order to search for a winning result, we can systematically try each possible game branch. Thus, we require only space enough to evaluate one branch at a time, as well as some bookkeeping to recall which branches we have already visited. This bookkeeping will require only  $O(m^2)$  space, where  $m$  is the number of nodes on the board. Thus, in polynomial space, we can evaluate all the possible outcomes of the game tree until we either find a winning strategy or determine that none exists.

□

It remains to be shown that strategies for the Sperner Game are *PSPACE*-hard.

**Theorem 4.1.3** *Determining whether the current player has a winning strategy in Atropos is PSPACE-hard.*[5].

Classically, we show that problems are *PSPACE*-hard by reducing *TQBF* to them [14]. In general, *TQBF* is the problem of determining whether a quantified boolean formula—a formula of the form  $\exists x_1 : \forall x_2 : \exists x_3 : \forall x_4 : \dots Q_n x_n : \phi(x_1, x_2, \dots, x_n)$ —is true. In our notation here,  $\phi(x_1, \dots, x_n)$  is a conjunctive normal form formula using the literals  $x_1$  through  $x_n$ , while  $Q_n$  is a quantifier (either  $\forall$  or  $\exists$ ).

Because of the inherent alternation in quantified boolean formulae, many games with non-obvious strategies for two players have been shown to be *PSPACE*-hard [14]. Indeed, we see that fulfilling a QBF (Quantified Boolean Formula) is much like playing a game. The hero will choose a variable assignment for  $x_1$ , then the adversary will choose for  $x_2$ . The hero chooses  $x_3$  in response, and so on.

#### 4.1.1 Taking Hints from GEOGRAPHY

Our reduction will model this behavior. We will create a Sperner Game state from a *TQBF* such that a winning strategy in the game exists if and only if the formula is true. Our reduction is inspired by the reduction of *TQBF* to Geography (see [14] for a clear description of this reduction). The game will proceed by letting the appropriate players make moves corresponding to the assignment of values to the variables  $x_i$ . Each player will then make one further choice and one of the literals in one of the clauses will be selected. We will “investigate” the literal through its interpretation in the game state and force an end of the game with it.

In our resulting game boards, most of the plays players make will be very restrictive. In our construction, there is a resulting “prescribed flow of play” directing players to make choices in the order we described above. Our reduction provides punishment strategies such that any player violating the prescribed flow of play will lose in a constant number of turns. We describe these punishment strategies, ensuring that players must follow the prescribed flow in order to have a chance of winning the game.

Using our construction, each player’s last choice is easily described: the adversary will choose a clause to investigate, and the hero will choose one of the literals in that clause.

That literal will be evaluated, according to the assignment it received. If the literal is true, the hero should win, otherwise the adversary should win.

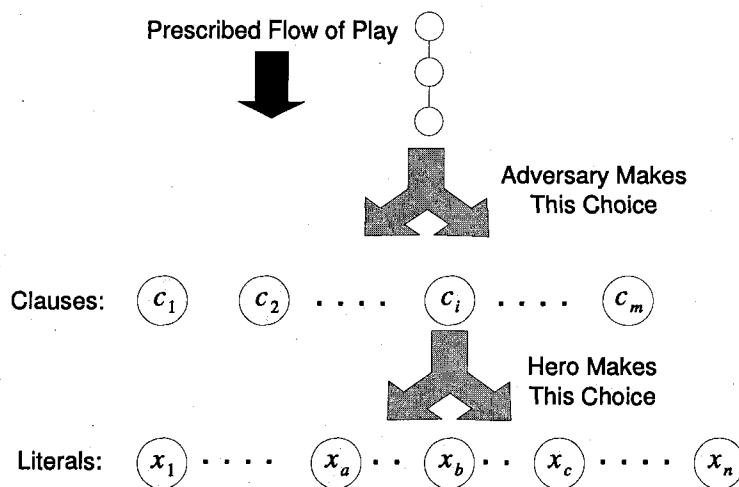


Figure 4.1: Construction Sketch: End of the Game. Clause  $c_i$  contains the literals  $x_a, x_b, x_c$ .

We give the adversary the power to pick a clause because, in the case where the formula is true, all the clauses must be true; the adversary cannot cheat at this point. However, if at least one of the clauses is false, the formula will be false, and the adversary should be able to pick whichever clause they want to discredit the correctness. Figure 4.1 illustrates the layout style we desire.

The prescribed flow of play directs players to finally investigate a literal at the end of the game. The adversary picks a clause to investigate, and the hero then chooses one of the variables.

Before the flow of play reaches this point, we have to have already set all of the variables. In order to accomplish this, the path of play must pass by each of the variable settings, forcing the appropriate player to make a decision at each variable. Once the settings have been accomplished, we can move to the investigation procedure, as portrayed in Figure 4.2.

Overall, this plan is very reminiscent of the reduction from GEOGRAPHY. In addition, our topology meets some similar hurdles that must be overcome in that construction. For instance, our plan has a non-planar design (paths must often intersect between the selection

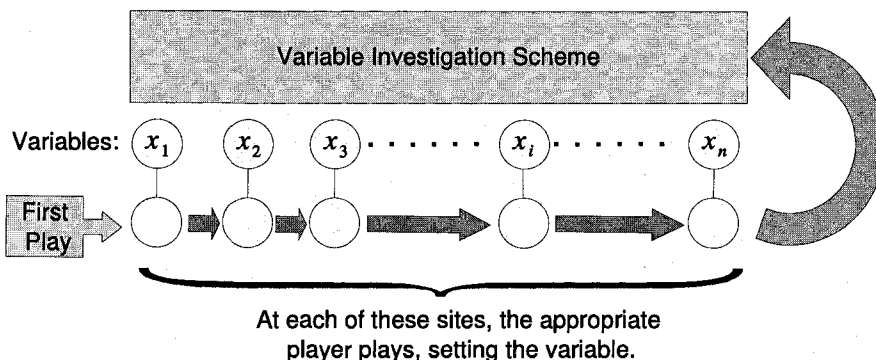


Figure 4.2: Construction Sketch: Setting the Variables. The Variable Investigation Scheme is laid out in Figure 4.1

of a clause and the selection of a literal during investigation). Thus, we will need to display widgets which allow for these logical crossovers to occur.

Our blueprints also seem to defy the rigid structure of the gameboard we are operating on. We require widgets which provide pathways for our prescribed flow of play. GEOGRAPHY is played on a directed graph, so enforcing the flow of play is somewhat more simple. We will need to be very careful that players cannot subjugate design plans by moving in unexpected directions. Also, we need widgets to handle variable assignment, path splitting, and other obstacles to realizing our layout on a Sperner Board. We continue by exhaustively describing these widgets.

#### 4.1.2 Reduction Widgets

Our reduction requires widgets to enforce various moves and allow for appropriate decisions to be made through some moves. In addition, the widgets must be able to connect, allowing us to build the overlying structure by fitting them together. In this section, we describe each of the widgets and specify how they are used.

Many of the widgets are simple and are only pathways to guide the flow of play. For more complex widgets, however, we need to be able to ensure that plays not following this flow correspond to trivially bad choices. This means that any player attempting to go against the prescribed pathway will be vulnerable to an optimal opposing winning strategy

which is easily computable in constant time.

### 4.1.3 Paths

Paths are the most simple of the widgets in our construction, although we have two different versions for different circumstances. Players should not make non-trivial decisions along paths, thus we build them to strongly restrict playing options.

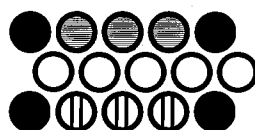


Figure 4.3: A Single-Symbol Path

The first of our two versions is a path in which on any move, a player has the option of playing exactly one symbol without immediately losing. In Figure 4.3, assuming the flow of play comes in from the left, the leftmost circle can and must be filled. Then, the next player is forced to fill the circle to the right, and so on. This path pattern can be extended to any length. Turning widgets for these paths also exist as we describe later.

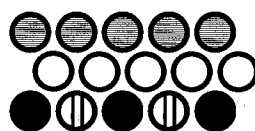


Figure 4.4: A Two-Symbol Path

The other type of path supports a chain of one of two different symbols. In this type of path, shown in Figure 4.4 whichever symbol is played forces the next play to follow suit. If a space is barred, the next play must also be bars. The same is true for filled (we assume again that the flow of play is going from left to right).

This type of path is often the by-product of play leaving other widgets. Since all our widgets use single-symbol paths leading in, it is vital to be able to force the path to switch from a two-symbol path to a single-symbol path.

Figure 4.5 shows a mechanism for this switch. The prescribed flow of play in this widget

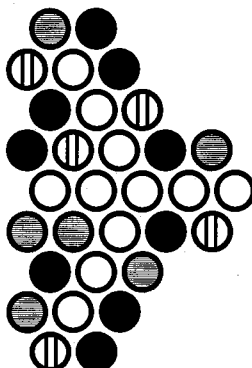


Figure 4.5: A Switch from Two-Symbols to Single-Symbol

in the diagram is again from left to right, strictly horizontal, although there are free spaces both above and below. Indeed, if a player deviates by playing above the horizontal line, there is a winning response, simply by playing further above. The same is true for playing below.

The beginning pattern of the single-symbol path is clear on the right-hand side of the widget, and, as before, no matter which play is made approaching that pattern, a sequence of fill plays can and must be made.

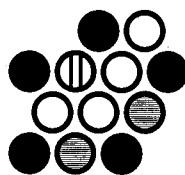


Figure 4.6: A 60 degree turn in a One-Symbol Path.

We must also be able to turn our paths in order to have them line up with other widgets. Figures 4.6 and 4.7 reveal 60-degree turning options to use while performing a reduction. In order to turn further than 60-degrees, we can just pair two or three of these together to attain 120 or 180 degree rotations. Note that in the second example (Figure 4.7) there are two possibilities for playing at the “elbow” of the turn. This does not affect the overall restriction; once further plays after the elbow must return to the original symbol.

Unfortunately for fans of two-symbol paths, we do not bother to create turning widgets

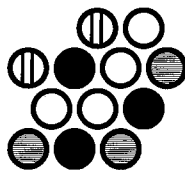


Figure 4.7: Another 60 degree turn in a One-Symbol Path.

for them. Instead, whenever we are presented with a situation where a two-symbol path occurs, we will immediately switch it to a one-symbol path. Thus, none of our other widgets will use two-symbol entrances. This does not present a problem, as only one of our widgets results in an out-going two-symbol path: the variable widget. The out-going paths for these widgets will be followed by the two-to-one symbol switch widget.

#### 4.1.4 Variables

Having described these devices, we are prepared to reveal the widget for modelling variables, presented in Figure 4.8.

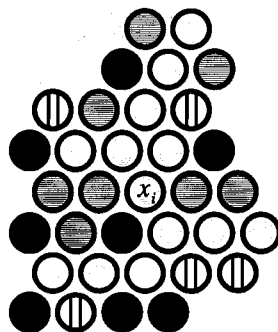


Figure 4.8: Variable Widget

Here the flow of play enters initially from the bottom left and exits through the lower right path. During this time, the “playability” of the location corresponding to some boolean variable  $x_i$  is determined. If that variable is investigated at the end of the game, then the flow of play will enter through the entrance in the upper right corner, and will terminate inside the widget.

The choice of symbol played in the space directly below  $x_i$  determines the playability



of  $x_i$  (and corresponds to the assignment of true or false). Since the plays up to that point must all either be fills, or—in the case of the last space—bars, the deciding play must also either be a fill or bars, and can be either, independent of what the previous play was. Notice now that if a fill is made, the location  $x_i$  will be playable later, whereas a play of bars means that  $x_i$  is unplayable.

After this choice is made, the prescribed flow of play continues rightward, which is clearly forced in the case that  $x_i$  is made unplayable. We now must describe a winning response strategy to a player deviating from the play flow, which occurs when  $x_i$  is played prematurely.

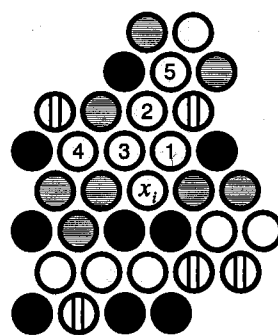


Figure 4.9: A Premature Play at  $x_i$

**Lemma 4.1.4 (Thwarting Cheaters)** *Prematurely playing at  $x_i$  is suboptimal.*

**Proof.** In order to prove this lemma, we must show a winning counter-strategy to a premature play in  $x_i$ . We use the numbers from Figure 4.9 to refer to play locations in this strategy.

Assume that an offending player played prematurely at  $x_i$ . If they have not already lost, then they must have either shaded or filled  $x_i$ . In either case, the winning punishment strategy begins by filling location 1. The offending player will have to respond by either

- a) Filling location 2. A winning response to this play consists of filling 3. Now, the offending player must play at the unplayable location 4.

- b) Filling or shading location 3. The winning response to this is to fill 2. Now, the offending player can only play at 5, which is unplayable.

Thus, playing at  $x_i$  prematurely is a losing strategy, and is suboptimal.

□

Now, assuming the players follow the prescribed flow, the parity of playable spaces in the upper portion of the widget is defined by the playability of  $x_i$ . If  $x_i$  is investigated, then the flow of play will enter at the upper right from a single-symbol shading path. Notice that these incoming plays cause location 1 in Figure 4.9 to be unplayable. Thus, the following sequence of non-suicidal plays is at locations 2, 3, then—if it is playable— $x_i$ .

Thus, if  $x_i$  is playable, a play there after 3 wins by forcing a loss at 1. Otherwise, the play at 3 wins, because all three neighboring spaces are unplayable. In our widget diagram, if  $x_i$  is playable, there are four possible plays when the widget is investigated. In the other case, there are only three.

#### 4.1.5 Splitting and Rejoining Paths

When determining which variable to investigate, players need to be able to make choices to follow different paths. In turn, multiple paths must converge towards the same variable, as multiple clauses can contain the same literal from our instance of QSAT. Thus, we require widgets for splitting and rejoining paths.

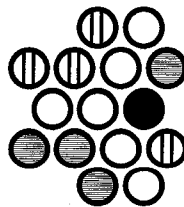


Figure 4.10: This widget splits a path.

The splitter shown in Figure 4.10 is extremely simple to comprehend: the flow of play goes from left to right. Whichever player makes the second play in the widget has the

ability to choose between the two paths. Assuming the first player is forced to fill the first circle (by using a single-symbol fill path to enter this widget) the second player can choose to either take the upper or lower path. Play continues through single-symbol paths. (Note that all plays have to remain fills, independent of which path is chosen.)

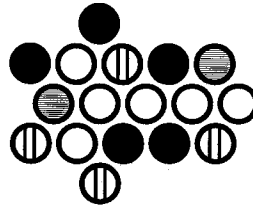


Figure 4.11: This widget converges two paths.

Joiner widgets are slightly more complex, as we have to protect the flow of play (again, left to right in Figure 4.11) and prevent a player from “going” backwards. The widget performs this automatically. Notice that play from the top must begin (in the widget) with a fill, forcing the second play to be a fill. Now, the space below is unplayable, preventing backtracking, and the players must continue to the right. The same phenomenon occurs when play enters from below: forced bars cause the upper entrance to be unplayable. Thus, our joiner widget “works” in the sense that it protects the flow of play from backtracking.

#### 4.1.6 Path Crossing

The graph of paths in our model is not necessarily planar, meaning that we have to be able to handle paths which cross. Thus, we need another widget that acts as a gate, forcing a crossing of two separate possible play flows. Luckily, such a widget exists, as pictured in Figure 4.12.

We keep with the practice of using left-to-right play flow in Figure 4.12, except that here we have two different possible flows: bottom-to-top and top-to-bottom. We will follow one of the path flows and list the sequence of necessary and optional plays, showing that it “flips” sides while passing through the widget. The reader may then investigate play flow from the other entrance.

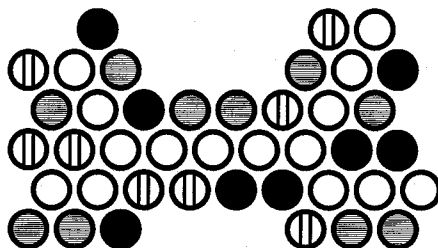


Figure 4.12: This widget crosses two paths.

Assume play enters the lower left side from a single-symbol fill path. Then, the first two plays in the widget diagram must be fills, followed by either a fill or bars at the path intersections. Either of these plays both makes a play upwards unplayable, and forces the next two plays to the right to be fills. The next play must be a shading, followed by another shading in the intersection. Now the lower path is unplayable, and play must continue upwards. By verifying entrance from the upper left path, the reader will see that the crossover widget accomplishes its goal.

In truth, the widget is even more robust than we describe above. As it turns out, threads of play may enter from any adjacent two of the four sides, and the path-crossing property will still hold. In our example construction in Figure 4.14, play enters from either of the bottom holes and exits from one of the top holes (the figure uses an upside-down version of the widget).

#### 4.1.7 Parity Switching

Finally, in order to ensure that the correct player has the chance to play at the variable locations  $x_i$  during investigation, we may need to extend the lengths of some paths. For each different possible path from the splitters representing the choice of a variable by the hero inside a clause and the variable location itself, we need to measure the length of the path to see which player would be playing on the variable site.

If the literal in that clause is negated, then the adversary should be set to play on the variable. Otherwise, it should be the hero's move that lands on the variable site. We can

control this by adding any necessary parity switches in the path segment between the last splitter and the first joiner. A parity switch widget is detailed in Figure 4.13. This device simply inverts the final order of play that would occur in a basic single-symbol path of the same lateral length.

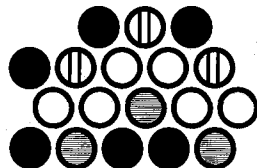


Figure 4.13: This widget effectively switches the order of play in a path.

#### 4.1.8 Sample Reduction

Although we will soon present an algorithmic method for fitting the widgets together, we first provide a sample QSAT instance and equivalent Atropos board that will all fit on one page. Although many readers will be satisfied with this example, those who are still concerned about the validity of the reduction can continue after this section.

We steal our example QSAT instance from a GEOGRAPHY example of Papadimitriou's [14]:  $\exists x : \forall y : \exists z : [(\bar{x} \vee \bar{y}) \wedge (y \vee z) \wedge (y \vee \bar{z})]$ . Although this example contains only two literals per clause, the construction utilizes all of our widgets and correctly portrays the ease with which they fit together.

In Figure 4.14, we have assembled the board using our widgets. We use  $S$  to denote the starting position (we assume that the adversary last colored the circle directly to the left of the  $S$  and that no other uncolored nodes are adjacent to that circle). Some of the other nodes have also been marked to show their correspondence to the formula. Each of the variables  $x$ ,  $y$  and  $z$  are noted in the figure, as well as the clauses  $c_1 = \bar{x} \vee \bar{y}$ ,  $c_2 = y \vee z$  and  $c_3 = y \vee \bar{z}$ . The  $c_i$ 's indicate locations where the hero will choose between one of the literals in that clause. In addition, we have marked the split  $A$  where the adversary makes their first choice between clauses. If they wish to investigate  $c_1$  or  $c_2$ , then they must take the upper path. Otherwise, choosing the lower path will lead directly to  $c_3$ .

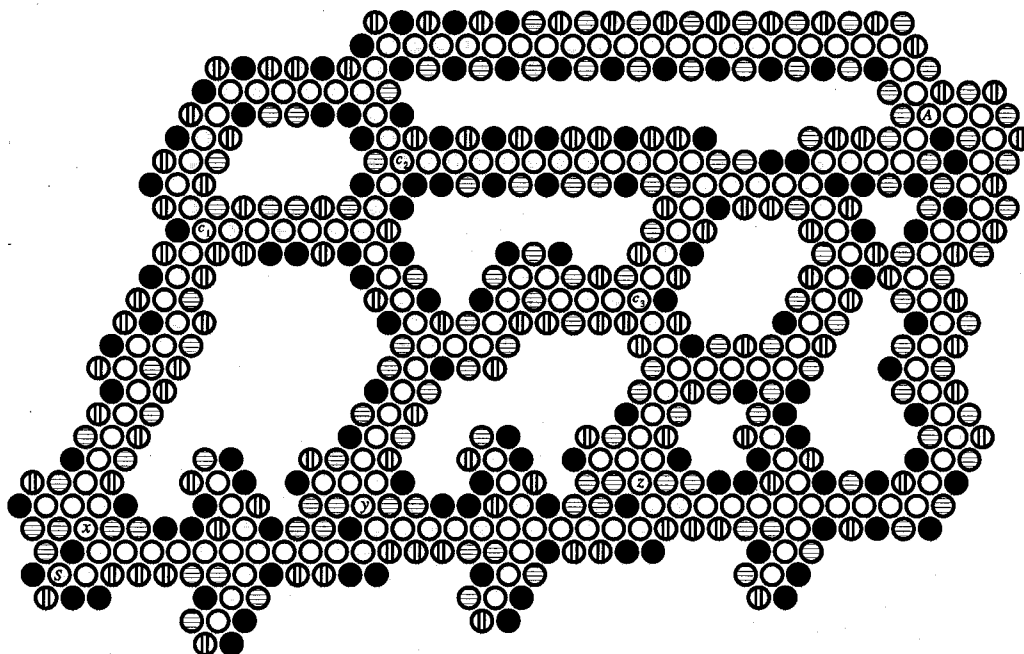


Figure 4.14: Atropos board equivalent to  $\exists x : \forall y : \exists z : [(\bar{x} \vee \bar{y}) \wedge (y \vee z) \wedge (y \vee \bar{z})]$

The attentive reader can count spaces to see that the hero will choose to set  $x$  and  $z$  and will get to choose between literals at each clause. The adversary gets to set  $y$  as well as choose between clauses. Furthermore, after each literal is chosen from each clause, the reader can count to see that the adversary is scheduled to play on the variables when they are negated in the appropriate clause, while the hero is scheduled to play on them when they remain un-negated.

Demonstrating that this is indeed a legal game state, Figure 4.15 shows the holes of our diagram filled in. Recalling earlier discussion, these holes must each simply contain a hexagon of seven colored circles connected to the border. We go a bit further here, and simply fill in the holes, aside from some doomed circles.

Having removed all unnecessary notation from the game board, we must just imagine that this structure exists within our Sperner triangle. Thus, we must only implant our construction within a large-enough initial board.

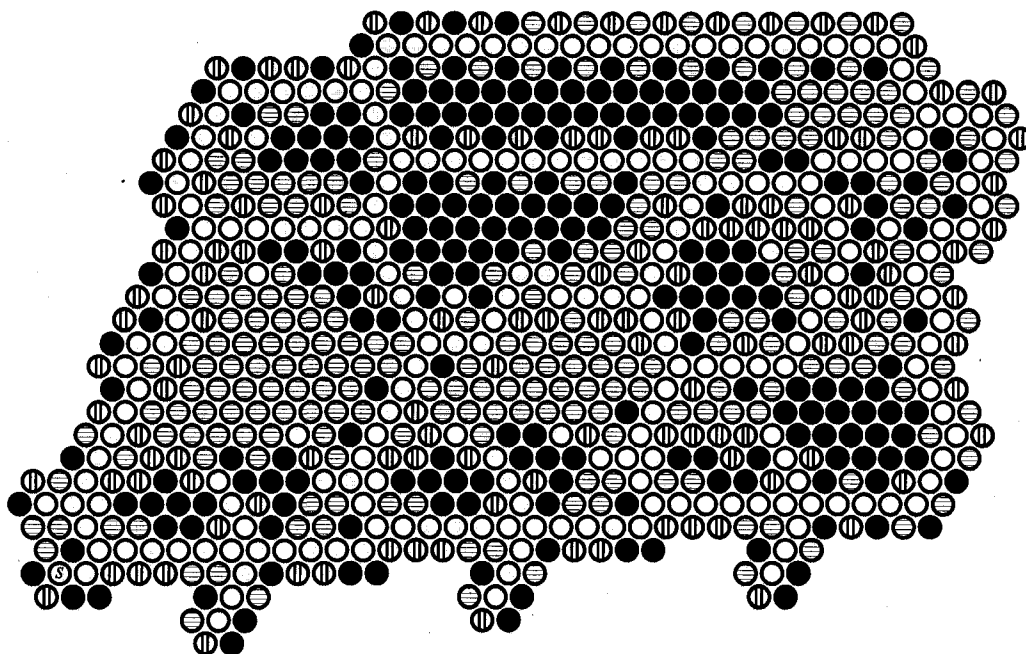


Figure 4.15: Filling in Holes Between the Widgets. We also remove the unnecessary notation from the nodes ( $S$  is still the start node, and must be denoted).

#### 4.1.9 Some Additional Concerns

We have described all our widgets, as well as given mechanisms for them to all fit together. The wary reader may have additional concerns, however. The most obvious of these is fitting our construction inside an actual game board. We have built a very oddly-shaped monstrosity, nowhere near the form of the triangle that is the initial board configuration. This is an easy obstacle to overcome, however: we can simply draw a large triangle around our game board. We can color as many circles as necessary outside our structure, so long as we don't create a bad triangle.

Slightly more troublesome, we need to assert that “islands” inside the structure could exist. These islands, consisting of colored circles surrounded by a connected shell of uncolored circles, can exist in a legal game board: they must simply have been the product of jumps. At some point before we reach the current game state, one of the players must have chosen to start coloring circles of that island. Then, the chain of plays must have created a place where another player was allowed to jump outside of that island.

Ensuring that jumps outside could have occurred is simply a problem of designing large-enough islands. We need to make sure that whenever a jump outside is needed, there must be a place where no neighboring moves were possible. In the simplest case, this requires a hexagonal structure of 7 circles, all of which are colored.

#### 4.1.10 Algorithmic Reduction: Putting the Pieces Together

We have the pieces, but now we need to show how to put them together efficiently. Although many readers might believe at this point that a reduction exists which can be computed in polynomial time using the widgets shown, we prefer to give an explicit construction for the entire board. This will complete the proof of PSPACE-completeness for Atropos.

**Lemma 4.1.5** *We can create an equivalent Atropos board in  $O(n^2)$  time, where  $n$  is the number of literals in the given QSAT formula.*

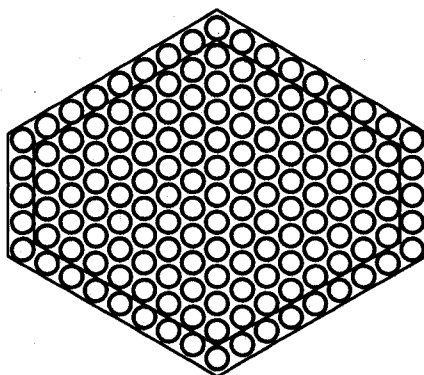


Figure 4.16: An empty hex tile. The border of the tile will overlap with neighboring tiles.

**Proof.**

In order to do this, we will need to put our widgets inside hexagonal tiles (see the empty tile in Figure 4.16) and piece those tiles together into a game board. The borders of each tile (between the lines in Figure 4.16) can overlap with any neighboring tile; patterns along each edge have been standardized so that we don't have to worry about making sure they match. Also, the design of each tile has been laid out so that each piece can be individually



created via a sequence of plays, with or without needing to account for the border. Each side of the structure in each separate tile can be built with a series of plays which ends in a location completely surrounded by colored circles and thus each tile can be created independent of the tiles around it.

We will build our tile structure in layers, in order to correctly sort out the widgets and paths between them. The trickiest of these steps is actually the arrangement of paths from literals to the variable widgets. This takes two steps: first the sorting of those literals, then the combining of the same literal paths into one variable widget.

In order to do this, we will use a structure that is reminiscent of a parallel bubble-sort. At each step, we'll compare two neighboring literals, and if they're out of order, we'll switch them. In each of these steps, there will be two rounds, first to compare the  $2k^{\text{th}}$  and  $(2k + 1)^{\text{th}}$  elements, then next to compare the  $(2k + 1)^{\text{th}}$  and  $(2k + 2)^{\text{th}}$ .

If the two literals need to be switched, then we switch them by having the proper tile contain a crossover widget (see Figure 4.12 for the widget and Figure 4.17 for the tile). Otherwise, if we don't need to switch, then the tile will just have two non-intersecting paths (see Figure 4.18).

Each step in our sorting procedure requires four rows of hexagons: two rows of potential crossovers and two more rows between them and the next step.

Once the literals are sorted, we need to use the joiner widget to bring paths that represent the same variable back together. Again, we use a hexagonal tiling to connect these paths. Figures 4.19 and 4.20 portray these joiner hexagons, which differ only in the direction they exit at the bottom. In the worst case (all literals are instances of the same variable), we will need  $O(n)$  rows of tiles to join all these paths. Each of the tiles in this section have a height of 13.

After the joining, we have each of the variable paths lead into a variable widget. Unsurprisingly, we have installed each of the widgets also into hexagons, each of height 17, as shown in Figure 4.21. This adds only one more row onto the bottom of the hexagonal structure.

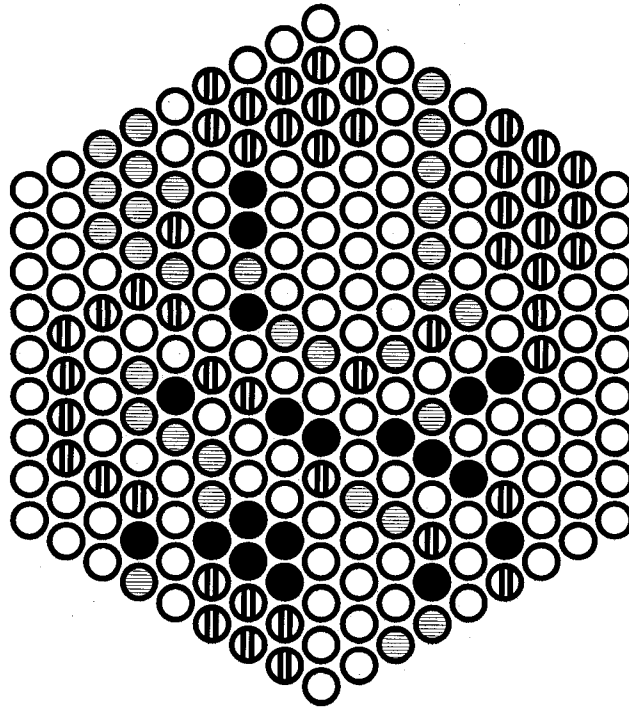


Figure 4.17: Two paths come in from the top, cross paths, and exit through the bottom.

Now we look backwards, building the separator branches that lead into the sorting section. We again use hexagons to systematically hold our separator widgets; this time each hexagon is 10 circles high, as shown in Figure 4.22. In our construction, we need at most  $O(n)$  rows in the splitting section, since in each hexagon, a path either splits or just moves right or left. Thus, in order to reach the left-most literal, we can use  $n - 1$  splitting hexagons, each on a different row to create our tree in this section.

Finally, we just need to connect the right-most exit of our variable row with the entrance to the splitting section. We can do this by simply building a path from the bottom of the structure to the top. With this, our gameboard is nearly complete. The only problem left is that this is not a legal Atropos state; we need our board to be a descendant of an original board. Thus, we embed our structure into a just-large-enough triangle with the starting border configuration.

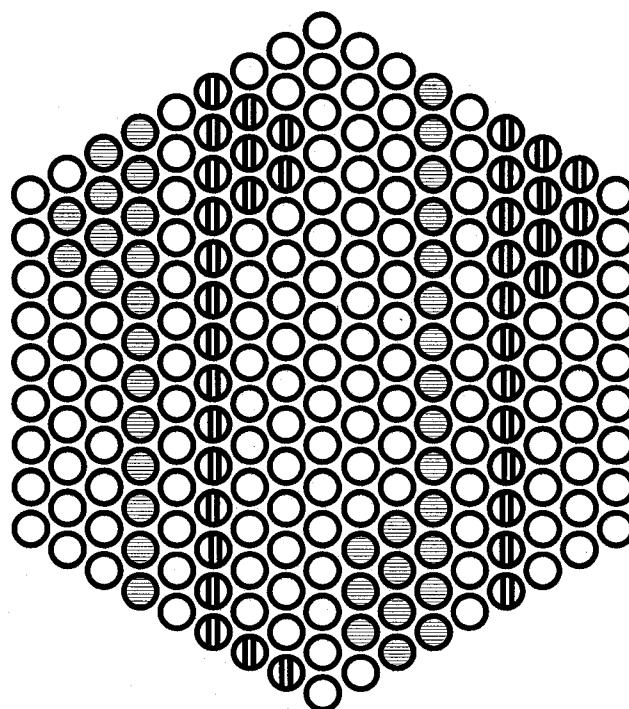


Figure 4.18: These paths don't need to cross and thus just go straight down.

The size of this now depends on the size of our structure. If, in our QBF we had  $n$  literals, then the width of our structure was  $n + 1$  hexagons of circles (the last hexagon contains the path connecting variable widgets to the splitting widgets). Each hexagon has width 17, so the total width is  $17(n + 1)$  circles.

The height of the structure consists of four parts. On top, the splitting section consists of  $n$  rows of hexagons, each of height 10. There are  $n - 1$  splitting hexagons, plus an  $n^{\text{th}}$  to connect to the path leading from the variable widgets. In total, the splitting section contributes  $10n$  rows of circles.

Below these splitters is the sorting section which orders the different paths depending on the variable number of each path's literal. This section mimics a Bubblesort process run in parallel on  $n$  processors. In each step of the process, we compare each literal with literals to the right and left, switching them if necessary, using 4 rows of hexagons, each of height 17. In the worst case, the literal which starts on the far right will need to move all

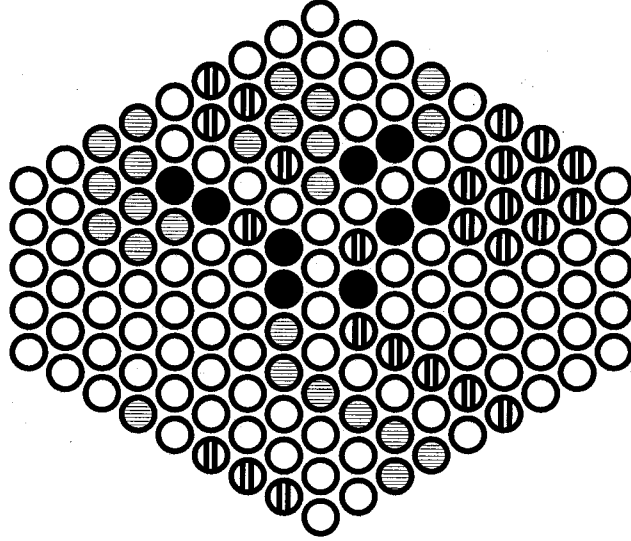


Figure 4.19: This tile joins two paths from above and exits to the right.

the way to the far left (or vice versa) and will only switch once per round. Thus, we need at most  $n - 1$  rounds of the process, or  $4(n - 1)$  rows of hexagons in this section, meaning  $68(n - 1)$  rows of circles.

In the next layer, consisting of joining widgets, in the worst case we need to recombine all paths back to one variable (if all literals refer to the same variable) which requires  $n - 1$  rows of hexagons, each 13 circles high. Thus this section uses  $13(n - 1)$  rows of circles.

The final layer only needs one row, as it just contains all the variable widgets connected together. Each of these hexagons is also 17 circles high, so only 17 rows of circles are contributed by the variables.

Our structure now has a width of  $17(n + 1) = O(n)$  circles and a height of  $10n + 68(n - 1) + 13(n - 1) + 17 = O(n)$  circles. In order to fit this into a triangular Atropos board, we would require a board with  $O(n^2)$  circles.

Thus, the size of the game board is a polynomial in the number of literals in the *QBF* and the reduction can be completed in  $O(n^2)$  time.

□

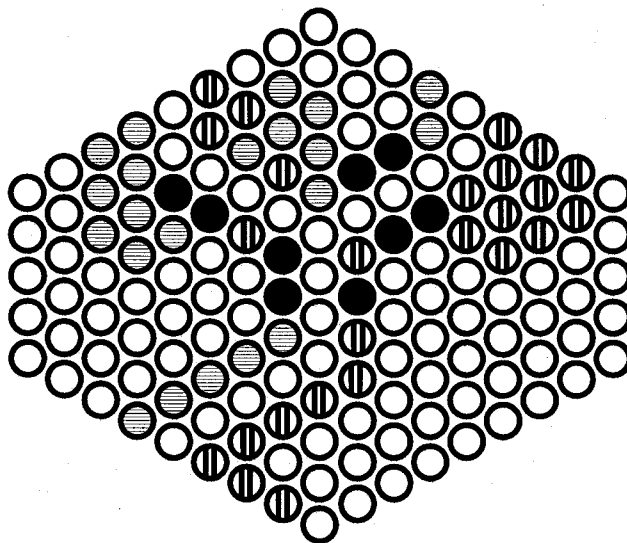


Figure 4.20: This tile joins two paths from above and exits to the left.

This completes our proof of *PSPACE*-hardness.

**Corollary 4.1.6 (Atropos is *PSPACE*-complete.)** *The problem, ATROPOS, of determining whether the current player has a winning strategy is *PSPACE*-complete.*

**Proof.** By Lemma 4.1.2 and Theorem 4.1.3, Atropos is both solvable in *PSPACE* and *PSPACE*-hard. Thus, Atropos is *PSPACE*-complete.  $\square$

## 4.2 Unrestricted Atropos

The movement restriction in Atropos is the property that appears to make this game actually difficult to strategize for. For *most* game configurations, it is a simple matter of determining the parity of uncolored circles to determine who will win. From the starting board configuration, the winning player has to put forth some effort to prevent their opponent from “flipping” this parity, but this is intuitively not difficult to regulate.

At the end of an Unrestricted Atropos game, we reach a state where a play at any uncolored circle on the board will create a three-colored triangle. Almost always, any of

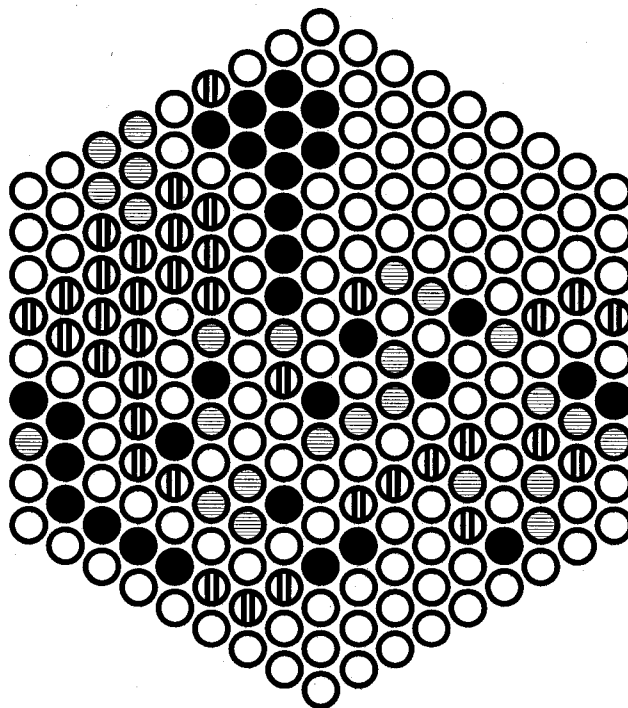


Figure 4.21: This tile contains a variable widget.

these plays will create an *odd number* of bad triangles; only a few configurations result in plays creating an even number (greater than zero) of them (see Figure 4.23). This has a vital impact on the winnability of Unrestricted Atropos due to the following important facet of Sperner's Lemma.

Not only does the lemma state that there will be *at least one* three-colored triangle in a fully-colored Sperner Triangle, but that there will be an *odd number* of them. Thus, if we don't end up with any of these parity switch regions by the end of the game, there will always be an odd number of uncolored circles just before the losing player makes their last move. Assuming either player can prevent these regions from occurring, the next player can win if there are currently an even number of uncolored circles. This parity changes with each switch region added to the board. We begin to argue that it is easy to prevent these regions from being created and relate this to playing this game from the starting position.

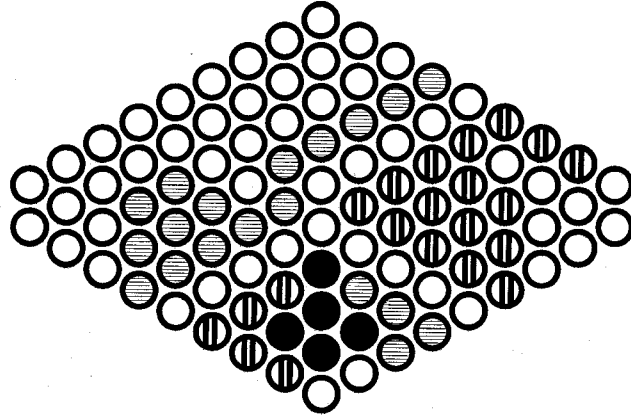


Figure 4.22: In this splitter tile, one path splits into two.

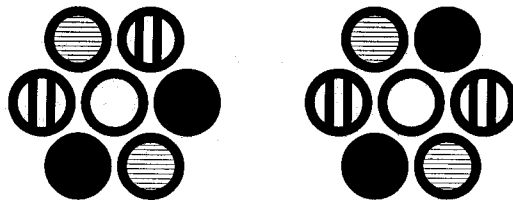


Figure 4.23: Basic parity “switches”. A play in either of these will create an even number of bad triangles. We refer to these as A-Switches.

We will first show all regions which *must* result in an even number of bad triangles (we will refer to these as “parity switches”) then give some intuition for the fragility of these regions.

Although the two regions described in Figure 4.23 (A-Switches) are the smallest possible regions “worth” an even number of bad triangles, other larger regions will result in the creation of one of these two. We describe these pictorially, but using additional symbols. The “null” symbol (see Figure 4.24) means that no color can be played at that circle without causing the game to end. The “box” symbol (see Figure 4.24) with one of the three colors means that while that circle is uncolored, the only non-losing play at that circle is the color inside the box (due to neighboring circles not in the diagram) *and* either all neighboring

circles not in the diagram are already colored or no non-losing play at any of those circles can remove the playability of the color in the box.



Figure 4.24: New Symbols for parity switch diagrams. On the left, the circle does not have any safe colors available. On the right, the circle can only be colored with bars.

As an example, we look at the B-Switches, shown in Figure 4.25. In the example structure on the right, the two open circles correspond to the two “symbolled” circles on the left. The top circle has no safe plays. Meanwhile, the bottom circle has only one safe play—bars—and the only play which could change the playability of bars (filling or shading the top circle) is a losing play.

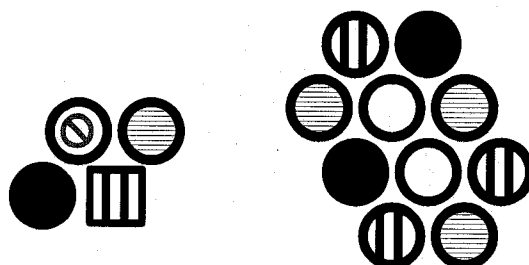


Figure 4.25: B-Switch. On the left is the general form. The right figure is a diagram of an example B-Switch.

We continue by diagramming the remaining switch regions. All these regions (including B-switches) are found by “backtracking” possible moves from the A-Switches, then taking the structures which still enforce the switch. C-Switches are shown in Figure 4.26, D-Switches in Figure 4.27 and E-Switches in Figure 4.28.

We next note that no other structures are parity switches. This can be shown by backtracking further on the switch regions. Any parent of one of the above structures either gives us another already-listed parity switch or is no longer a switch.

The concept behind preventing parity switches, then, is to detect them at least one



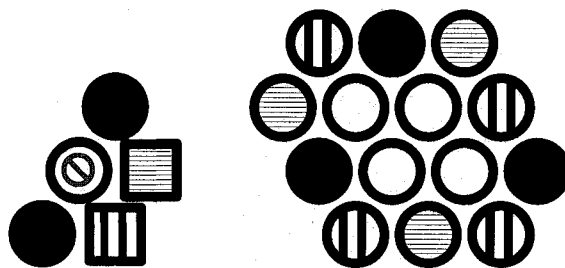


Figure 4.26: C-Switch. On the left is the general form. The right figure is a diagram of an example C-Switch.

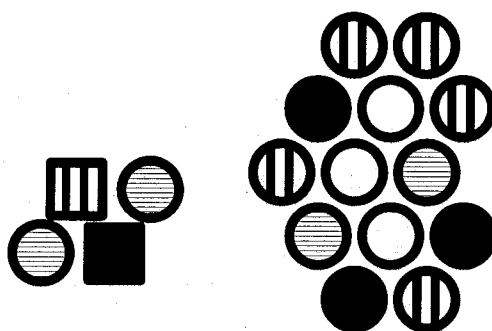


Figure 4.27: D-Switch. On the left is the general form. The right figure is a diagram of an example D-Switch.

move before they occur and make a play to break their structure. (Since most of them require different symbols next to each other, this is usually accomplished by painting a circle the same color as it's neighbor which is a part of a potential parity switch.) By the definition of a parity switch, any region which is not yet a parity switch but is threatening to become one can be "blocked" in just one move (otherwise it would already be a switch). Thus, any single switch can be prevented just one move ahead of time. Unfortunately, multiple near-switches may be neighbors, so one play (by the player hoping to set up a switch) could influence both regions. More specifically, that play could push both potential switch regions into a state where each was only one play away from becoming an actual switch. In this way, the player trying to create a switch might be able to pull that off; the other player can now only prevent one of the regions from realizing itself.

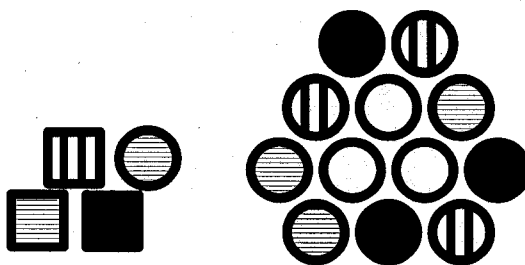


Figure 4.28: E-Switch. On the left is the general form. The right figure is a diagram of an example E-Switch.

Due to the nature of the board, it seems more likely that potential switches need to be recognized at least three moves before they complete. This results from the hexagonal structure of the Atropos board; each circle has six neighbors and thus any one play could influence up to three different threatening switch regions. We conjecture that this is good enough for one player to prevent switches from occurring from the starting position.

**Conjecture 4.2.1 (Initial Unrestricted Atropos is easy)** *The player who would not play in the last uncolored circle if the entire board were to be filled has a winning strategy for Unrestricted Atropos from the starting configuration.*

We see the following potential method for this proof by showing two separate things. First, prove that we can identify parity switches at least three moves in advance. Then, just show that this is adequate to preventing neighboring “four-move-away” switch regions from becoming completed switch regions.

Not starting from the initial configuration, we mention that there are some configurations which we do not know how to efficiently evaluate. Long strands of interconnected potential switch regions can form very complicated game boards where the attention is now paid to the parity of the number of these almost-switches. Players must be mindful of which plays can strengthen or destroy multiple of these regions. In this way, the dependence between neighboring regions can resemble situations from the game Kayles.

Kayles is a game played on an unweighted graph where each turn the current player

chooses a node,  $v$ , then removes  $v$  and all direct neighbors of  $v$  from the graph. The game continues until the last vertices in the graph have been removed; the player who makes this final move wins. This game is known to be *PSPACE*-complete in general [18], but the complexity is not known for more structured versions. Even Kayles on extremely structured graphs, such as star graphs[8], is not yet known.

Due to the planar board structure of any version of Atropos, it seems that a reduction to Kayles would result in planar graphs, but even this would not directly solve the complexity; at the time of this writing there is no efficient solution for planar Kayles.

Alternatively, a polynomial-time strategy *from the initial position* seems more likely; perhaps an algorithm could be designed which protects the parity of the board by destroying any potential switch region before they can grow and develop such interdependencies.

### 4.3 Full Matchmaker

From the previous section, we defined Full Matchmaker to be the matching game where the starting position can be any complete matching on the candidates. We will now show how to determine who will win this game, independent of the choices for moves that they make.

First, for any matching,  $M$  on  $\{1, \dots, n\}$ , we refer to two pairs,  $(a, b), (x, y) \in M$  as being *crossed* if  $(a < x \text{ and } y < b)$  or  $(x < a \text{ and } b < y)$ . Notice that it is always a legal move to uncross these edges, by proposing either  $(a, y)$  or  $(x, b)$ , depending on which two are the more preferred candidates. We refer to such a move as an *uncrossing move*.

We need the following lemma:

**Lemma 4.3.1 (Uncrossing Lemma)** *On an fully-matched board, an uncrossing move will always remove an odd number of crosses from the board.*

**Proof.** Let the two edges we are considering be  $(w_a, e_b)$  and  $(w_b, e_a)$  (without loss of generality, we can assume  $w_a < w_b$  and  $e_a < e_b$ ). Let  $M$  be the matching which contains them both. Thus, we want to show that the move from proposing the pair  $(w_a, e_a)$  removes

an odd number of crosses from the board. The edges  $(w_a, e_a)$  and  $(w_b, e_b)$  will replace the crossed pairs to create  $M'$ .

To prove this, we will show that any other edge  $(w, e)$  crosses  $(w_a, e_b)$  and  $(w_b, e_a)$  with the same parity (odd or even number of times) as it crosses  $(w_a, e_a)$  and  $(w_b, e_b)$ . Thus, the other edges do not affect the total crossing parity, and only the uncrossing of the two edges changes this by 1. We will look at cases for  $(w, e) \in M$ .

We have three cases:  $w < w_a$ ,  $w_a < w < w_b$  and  $w_b < w$ .

Case 1:  $w < w_a$ . We have three subcases:  $e < e_a$ ,  $e_a < e < e_b$  and  $e_b < e$ .

Subcase 1.1:  $e < e_a$ . Here,  $(w, e)$  doesn't cross any of the edges in question. The total change will be 0.

Subcase 1.2:  $e_a < e < e_b$ . Now,  $(w, e)$  crosses  $(w_b, e_a)$  in  $M$  and  $(w_a, e_a)$  in  $M'$ . The total change is also 0.

Subcase 1.3:  $e_b < e$ . Here,  $(w, e)$  crosses both  $(w_a, e_b)$  and  $(w_b, e_a)$  in  $M$  along with  $(w_a, e_a)$  and  $(w_b, e_b)$  in  $M$ . Again, the total change is 0.

Case 2:  $w_a < w < w_b$ . Again, we have three subcases:  $e < e_a$ ,  $e_a < e < e_b$  and  $e_b < e$ .

Subcase 2.1:  $e < e_a$ . This case is equivalent to subcase 1.2. The total change is 0.

Subcase 2.2:  $e_a < e < e_b$ . In this case,  $(w, e)$  crosses both  $(w_a, e_b)$  and  $(w_b, e_a)$  in  $M$ , but neither of the two edges in question in  $M'$ . Thus, the change is 2, and the change in parity is 0.

Subcase 2.3:  $e_b < e$ . This case is equivalent to subcase 1.2. The total change is 0.

Case 3:  $w_b < w$ . The subcases here follow the same as in case 1. The total parity change is still always 0.

Thus, an uncrossing move always removes an odd number of crosses from the board. Note that this proof does not depend on the fact that the board is full; we will use this fact later on with partial matchings.  $\square$

Since the stable situation has an even number of crosses (zero, to be exact), and this position is a loss for the next player, all positions with an even number of crosses are a loss

for the next player. Thus, to determine if the next player wins, one simply must count the number of crosses in the current matching.

**Corollary 4.3.2** *The number of all Full Matchmaker game is always either 0 or 1 and is equivalent to the parity of the number of crossed pairs.*

**Proof.** The terminal Full Matchmaker game (which has number 0) has no crossed pairs. Thus, any parent of this game must have an odd number of crossed pairs and also have number 1. By induction, we can extend this to prove the claim.  $\square$

## 4.4 Basic Dictator

It turns out that it is simple to figure out who is winning a game of Basic Dictator. This realization is based on the following lemma:

**Lemma 4.4.1** *The function  $F$  (which must satisfy IIA) associated with a game board violates Pareto if and only if no dictator can exist in the board.*

**Proof.**

We will prove this by proving both directions of our if and only if statement.

[ $\Leftarrow$ ]

This direction is trivial using Arrow's Theorem: if no dictator can exist in the board, then Arrow's Theorem states that a function,  $F$ , which satisfies IIA cannot also satisfy Pareto.

[ $\Rightarrow$ ]

We will prove this direction by proving the contrapositive: as long as a dictator could still exist in a partial board, Pareto and IIA are satisfied. Thus, assume we have a partially-filled board  $B$ , in which a dictator could still exist. We must show that there exists a social choice function  $F$  which satisfies IIA and Pareto and which is consistent with  $B$ .

Let  $B'$  be a completely-filled descendant of  $B$  with *exactly one* dictator. Let that dictator be the  $i$ th ballot on the board. Then, we choose our social choice function,  $F$ , to

be the dictator function which simply returns the list equivalent to ballot  $i$ . By our choice of the dictator,  $F$  is consistent with  $B'$  and thus the ancestor  $B$ . Also, since  $F$  is a dictator function, it satisfies both Pareto and IIA.  $\square$

Using this lemma, players can easily determine exactly when someone has made a “violating” play. This occurs either when a dictator is created or no dictator can be created. Thus, as play proceeds, players must be mindful of only two objectives: do not create a dictator, but do not rule one out (see Figure 4.29). As ballots are filled in, possible locations for dictators will be removed until the final ballot is completed, revealing the dictator. Thus, when played optimally, the game will end when the last slot is filled on the board, and players need only count the number of plays possible on the board to determine who will win. Even when multiple dictators are possible, all others will be “removed” until only one remains (see Figure 4.30).

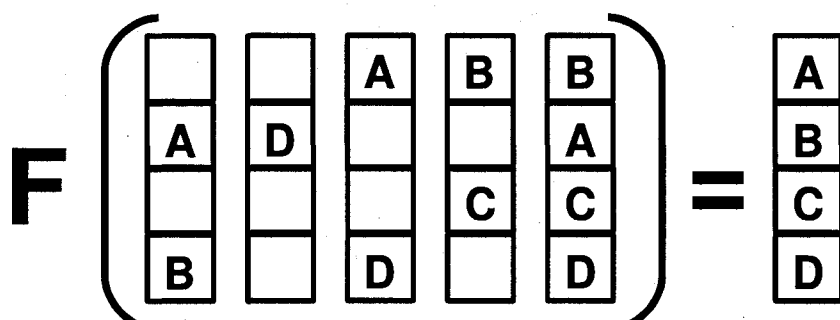


Figure 4.29: In a Basic Dictator game. Only the third ballot could be the dictator. Thus, players must avoid playing in that ballot as long as possible. Any play there will either create the dictator or make the dictator impossible, violating Pareto.

**Corollary 4.4.2 (Basic Dictator is in  $P$ )** *Determining which player wins a game of Basic Dictator is in  $P$ .*

**Proof.** Assuming all players play optimally, the last play will be the losing play. Thus, a player need only count the number of plays remaining. If  $F$  takes  $x$  ballots, and uses  $y$  candidates, and  $z$  slots have already been filled on the board, then the next player will win

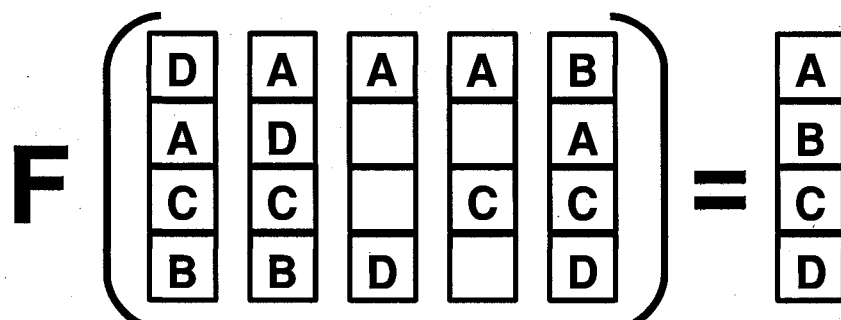


Figure 4.30: Near the end of a Basic Dictator game. Both of the unfilled ballots have the potential to be a dictator. The next player must choose one of them and play so that the ballot is not a dictator. The other player will then lose by either creating the Dictator or violating IIA and Pareto. Even though there are potentially more than one Dictator, this does not change the parity of the outcome.

exactly when  $x(y - 1) - z$  is even. This is exactly the number of remaining plays on the board.  $\square$

## 4.5 Dictator

Dictator has a dual-threat of interesting properties as far as the complexity of the game is concerned. There are two main questions we can ask about the difficulty of playing a game of Dictator. The most obvious, as studied in games throughout this thesis, is, “How difficult is it to determine whether the current player has a strategy to win the game?” This problem is in *PSPACE* and has the potential of being *PSPACE*-complete.

The second question is, given a Dictator game state, “can the current player successfully show a Pareto violation this turn?” In practice (at least, the practice we’ve seen so far) games are often lost when a player does not see a challenge they could have made on their turn. Once more ballots are filled up on the board, it becomes easier to see where violations could exist and is thus costly when players don’t see a viable proof. We now show that answering this problem is in *NP*acronymNPNondeterministic Polynomial time.

**Lemma 4.5.1** *Determining whether a proof of a Pareto violation exists on a Dictator board*

*is in NP*

**Proof.** Assume we have a Dictator board on  $n$  candidates with  $m$  ballots. In our sequence of IIA-manipulations we may need, in the worst case (time-wise), to completely rearrange each ballot. Since each ballot has  $n$  entries this would require  $O(n^2)$  time to rearrange. Since we have  $m$  ballots, the total time is  $O(mn^2)$ .

Note that there is no need to perform a more roundabout reshuffling of the ballots other than a straight-forward reordering. Any extra back-and-forth shuffling steps will not add to the proof and may only be detrimental to the ordering in the board's result.  $\square$

Although we have not yet proven a completeness result for this problem either, Dictator now has the potential to provide two difficult problems in one game. It would be excellent to try to solve an *NP*-complete problem to determine whether to play and a *PSPACE*-complete problem to determine *how* to play!

## 4.6 Other Game Complexities

We note that we have not listed all of the games mentioned in the previous chapter. For many of the variants, we do not yet have results. We will spend the next chapter discussing using computer programs to attempt to solve some complexity issues. Even still, some games remain unanalyzed. We list and discuss open problems and remaining questions in Chapter 9.



## Chapter 5

# Programming to Solve Games

### 5.1 Nimbers for Impartial Games

Already, we can use the children of a game to recursively determine whether that game has a winning strategy.  $G$  has a winning strategy (rather,  $G$  is *winnable*) exactly when there is a child,  $G'$ , of  $G$  which does not have a winning strategy ( $G'$  is *unwinnable*). For impartial games, we can use Sprague-Grundy evaluation (see Definitions 2.1.4 through 2.1.6) to determine whether any children are unwinnable. To do this, we just test whether the number of a game is zero.

From the recursive definition of  $G$ , we can classify the winnability of games using their numbers. Indeed,  $G$  has a winning strategy if and only if  $\mathcal{G}(G) > 0$ . Moreover, this allows us to analyze composite games. That is, if we have two different game states,  $G$  and  $H$ , we denote  $G + H$  to be the composite game, which has children:  $\{G' + H \mid G' \text{ is a child of } G\} \cup \{G + H' \mid H' \text{ is a child of } H\}$ . This definition simply means that each turn, a player may choose to make one of the legal moves in one of the boards. The famous result here is that  $\mathcal{G}(G + H) = \mathcal{G}(G) \oplus \mathcal{G}(H)$  [20][12][4] (where  $\oplus$  is the bit-wise xor operator). Nimbers allow us to find the number (and thus the winnability) of composite games, where only the winnability would not. For example, if we know  $\mathcal{G}(G) = 5$  and  $\mathcal{G}(H) = 3$ , then  $\mathcal{G}(G + H) = 5 \oplus 3 = 4$ , so  $G + H$  is still winnable. Also, if we know that  $G$  is unwinnable,

then  $G + H$  is winnable if and only if  $H$  is winnable, as  $\mathcal{G}(G + H) = \mathcal{G}(G) \oplus \mathcal{G}(H) = \mathcal{G}(H)$ . On the other hand, if we only know that both  $G$  and  $H$  are winnable, we don't actually know whether  $G + H$  is winnable! (The numbers of both could be the same, in which case  $G + H$  is not winnable.)

In order to evaluate games for nimbers, we build a simple Java framework to encapsulate the notion of impartial games in an `ImpartialGame` class. Along with this, we developed a “Nimberator” class to recursively determine the number of any `ImpartialGame`-implementing object. This source code is available online<sup>1</sup>. Unfortunately, this definition-based approach requires an amount of resources exponential in the size of the game board to carry out the computation.

In the following sections, most of our analysis will not just be on whether we can determine the winnability of games, but whether we can determine the number of games.

## 5.2 Atropos

For many combinatoric games, determining the winner from starting a starting position is an often-considered task, even if the game is hard in general. For some impartial games, there is often a simple trick towards proving this initial-playing result. In Chomp, for example, a strategy-stealing argument shows that the first player always has a winning strategy [9]. This result is entirely non-constructive, however; the theorem does not reveal a winning strategy or even a good first move. Instead, there is a special move from that initial play which has all the same other children (not itself) as the initial move. Thus, if none of those other children are unwinnable, that special child must be, as it will have no unwinnable children.

For many other games, it is unknown whether such a trick exists. Often, however, a pattern of winnability emerges as the size of the initial board changes. For the game Sprouts (a popular game in the Netherlands) extensive testing has supported a widely-

---

<sup>1</sup>These Java classes can be found at: <http://cs-people.bu.edu/paithan/thesis/code/java/impartialgames/>

believed pattern. For an initial board of size  $n$ , the first player has a winning strategy exactly when  $3 \leq n \pmod{6} \leq 5$ [13]. The misere version of this game has been studied for sizes up to 16, with results announced as recent as January 8, 2009[15].

After first examinations, Atropos seems to have a similar pattern from initial positions. When we mention the size of an initial Atropos board, we are referring to the number of uncolored circles along each side. Thus, for size 1, there is one uncolored circle along each side. In this board, there is only one open circle on the entire board; thus the first play must create a bad triangle. For size 1, the first player loses.

The initial board of size 2 (3 open circles) has the same result. The first player can play in any of the three corners, and the second player will also be able to play safely in another. The first player will then have to play last, losing again. For size 2, the first player also loses.

For size 3, the first player suddenly has a winning strategy! There is a strategy which forces the game to last until there is only one open circle left. The second player must play there, and will then lose. So far, in these three scenarios, the loser is forced to play in the last remaining circle on the board.

Since the number of circles in an initial board of size  $n$  is  $\frac{n(n+1)}{2}$ , we can easily determine who wins the game if the entire board is filled up during play. For exactly  $1 \leq n \pmod{4} \leq 2$ , the second player wins if the entire board is filled. If, from the starting position, either player can force this situation to occur, then this relationship tells us exactly who will win the game.

We explore this relationship by testing Atropos from starting sizes. These tests are done by running code (available online<sup>2</sup>) to play out entire game trees from starting configurations. Using fortran code, we compute the following results: the first player wins games of size 3, 4 and 7, while the second player wins games of sizes 5 and 6. We hope to improve the code to perform further tests on larger games. So far, however, the pattern seems to hold. Thus, we make the following conjecture.

---

<sup>2</sup>The Atropos fortran code is available at: <http://cs-people.bu.edu/paithan/thesis/code/fortran/> .

**Conjecture 5.2.1 (Atropos Conjecture)** *On a starting Atropos board of size  $n$ , the first player has a winning strategy exactly when  $n \pmod{4} = 0$  or  $n \pmod{4} = 3$ .*

Any solution to this question would be remarkable, as currently the strategy-stealing argument is the most common method for understanding these relationships and it doesn't seem like this principle can be applied to Atropos. In Sprouts, even though the winnability pattern has been shown up through a starting size of 32 (with some other boards up to size 47 proven) a proof is still seemingly out of reach. Luckily, with Atropos, we have a relationship to filling up all the open circles. It may prove useful, then, to consider the following conjecture.

**Conjecture 5.2.2** *From a starting Atropos board, either player can enforce that the game will end by playing at the last open circle.*

### 5.3 Stubborn Matchmaker

Unlike Atropos, adding a restriction to Matchmaker to create Stubborn Matchmaker simplifies the analysis of game states. Whereas in Unrestricted Atropos, it is reasonably easy to determine which of the potential moves is good, the same is not as abundantly clear in Matchmaker. Reducing the number of choices serves to limit the number of options each player has to look at and search.

Although this has simplified the analysis thus far, we still have not been able to fully solve this game. In this section we describe our findings, most of which require the following two conjectures:

**Conjecture 5.3.1 (Uncrossing Effect)** *If matching  $M'$  is derived from  $M$  simply by making one uncrossing move, then  $\mathcal{G}(M') = \mathcal{G}(M) \oplus 1$ .*

The second conjecture (see Figure 5.1) relates two different matchings:  $M$  and a similar match, but with two neighboring edges added, which we call  $M(i, j)$ . Formally,  $\forall i, j \in \{1, \dots, |M|\}$ :

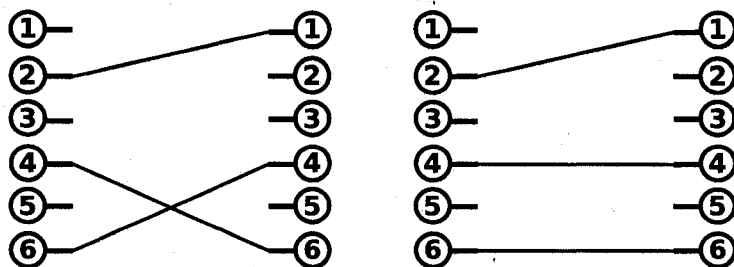


Figure 5.1: We conjecture: the number of the left game is equal to the number of the right game  $\oplus 1$ .

$$\begin{aligned}
 M(i, j) = & \{(i, j), (i + 1, j + 1)\} \cup \\
 & \{(a, b) | (a, b) \in M \wedge a < i, b < j\} \cup \\
 & \{(a, b + 2) | (a, b) \in M \wedge a < i, b \geq j\} \cup \\
 & \{(a + 2, b) | (a, b) \in M \wedge a \geq i, b < j\} \cup \\
 & \{(a + 2, b + 2) | (a, b) \in M \wedge a \geq i, b \geq j\}
 \end{aligned}$$

See Figure 5.2 for a visual description.

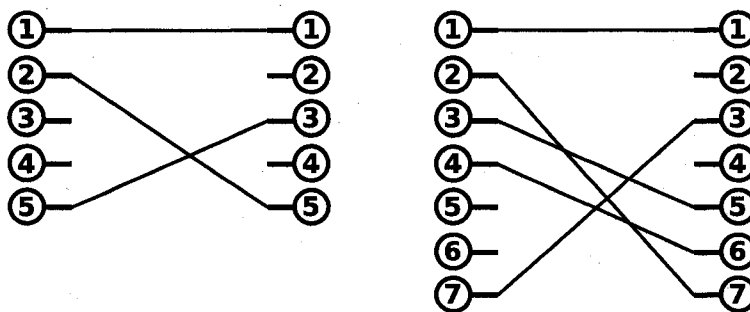


Figure 5.2: Left: a matching,  $M$ . Right:  $M(3, 5)$ . We conjecture that the numbers for these boards are the same.

**Conjecture 5.3.2 (Neighbor Pairs)** For two matchings,  $M$  and  $M(i, j)$  described above,  $\mathcal{G}(M) = \mathcal{G}(M(i, j))$ .

Assuming these two conjectures hold, we are able to perform some transformations on

the game board without affecting its number. First we will show how to perform operations we refer to as “un-Z-ing” and “Z-ing”: For any matching  $M$  and  $a, i, j, z$  where  $a < i$  and  $j < z$  which contains the following three edges:  $(a, j - 1), (i, j), (i + 1, z)$ ,

$$\begin{aligned}
 M_{\text{un-Z}}(i, j) = & \{(a, z - 2)\} \cup \\
 & \{(x, y) \mid (x, y) \in M \wedge x < i \wedge y < j - 1\} \cup \\
 & \{(x - 2, y) \mid (x, y) \in M \wedge x > i + 1 \wedge y < j - 1\} \cup \\
 & \{(x, y - 2) \mid (x, y) \in M \wedge x < i \wedge y > j\} \cup \\
 & \{(x - 2, y - 2) \mid (x, y) \in M \wedge x > i + 1 \wedge y > j\}
 \end{aligned}$$

See Figure 5.3 for a visual example.

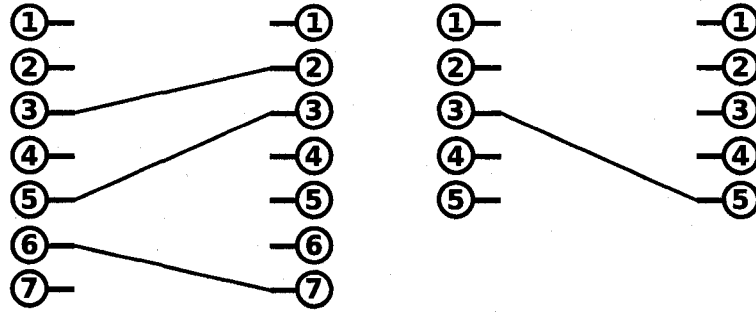


Figure 5.3: Left: a matching,  $M$ . Right:  $M_{\text{un-Z}}(5, 3)$ . We conjecture that the numbers for these boards are the same.

**Lemma 5.3.3 (un-Z)** *For any matching  $M$  and  $a, i, j, z$  where  $a < i$  and  $j < z$ , if  $M$  contains  $(a, j - 1), (i, j)$  and  $(i + 1, z)$ , then the previous two conjectures imply  $\mathcal{G}(M) = \mathcal{G}(M_{\text{un-Z}}(i, j))$ .*

**Proof.** Assume we have a matching  $M$  and  $a, i, j, z$  where  $a < i$  and  $j < z$  and  $(a, j - 1), (i, j), (i + 1, z) \in M$ . Notice that the Uncrossing Effect (the first of our two conjectures) works also for crossing edges, since the operation of xor-ing by 1 is its own inverse. Thus, if we replace the edges  $(a, j - 1)$  and  $(i, j)$  with  $(a, j)$  and  $(i, j - 1)$ , this crosses those edges and the number of this game is also  $\mathcal{G}(M) \oplus 1$ . Now, we have a matching with the two edges  $(a, j)$  and  $(i + 1, z)$ . Crossing these two edges, we replace them with the

two edges  $(a, z)$  and  $(i + 1, j)$ . Let this new matching be  $N$ . Then the number of  $N$  is  $\mathcal{G}(N) = \mathcal{G}(M) \oplus 1 \oplus 1 = \mathcal{G}(M)$ .

This new configuration has the two edges  $(i, j - 1)$  and  $(i + 1, j)$ . Thus, there is some matching  $P$  where  $N = P(i, j - 1)$ . By our Neighbor Pairs conjecture,  $\mathcal{G}(N) = \mathcal{G}(P(i, j - 1)) = \mathcal{G}(P)$ . After removing those two edges from  $P(i, j - 1)$  to  $P$ , we are left with the edge  $(a, z - 2)$ ;  $P$  is exactly the matching  $M_{\text{un-Z}}(i, j)$ . Thus,  $\mathcal{G}(M_{\text{un-Z}}(i, j)) = \mathcal{G}(P) = \mathcal{G}(P(i, j - 1)) = \mathcal{G}(N) = \mathcal{G}(M)$ .  $\square$

With the ability to use these Z's, we now discover the ability to shift two neighboring connected candidates either up or down without changing the game's value. More formally, for any matching  $M$  and an  $a$  where  $\exists(a, i), (a + 1, j) \in M$ ,

$$\begin{aligned} M'_{\text{west}}(a) = & \{(1, i), (2, j)\} \cup \\ & \{(x, y) \mid (x, y) \in M \wedge x > a + 1\} \cup \\ & \{(x + 2, y) \mid (x, y) \in M \wedge x < a\} \end{aligned}$$

**Lemma 5.3.4 (Shifting)** *Assuming the previous two conjectures, if  $\exists(a, i), (a + 1, j) \in$  matching  $M$ , then  $\mathcal{G}(M) = \mathcal{G}(M'_{\text{west}}(a))$ .*

**Proof.** Assume we have matching  $M$  with edges  $(a, i)$  and  $(a + 1, j)$ . Consider the matching  $M(1, i)$ . Using the Neighbor Pairs conjecture,  $\mathcal{G}(M(1, i)) = \mathcal{G}(M)$ . Additionally, this new matching contains the following edges:  $(2, i + 1)$ ,  $(a, i + 2)$  and  $(a + 1, j)$  which form a Z. Applying the Un-Z lemma, we find that  $\mathcal{G}([M(1, i)]_{\text{un-Z}}(a, i + 2)) = \mathcal{G}(M(1, i))$ . After this Un-Z step, the remaining matching is exactly  $M'_{\text{west}}(a)$ . Thus,  $\mathcal{G}(M) = \mathcal{G}(M(1, i)) = \mathcal{G}([M(1, i)]_{\text{un-Z}}(a, i + 2)) = \mathcal{G}(M'_{\text{west}}(a))$ .  $\square$

With this last lemma in place, we see we can categorize boards in the following way. We say that a board belongs to class  $(x, y, z)$  if the number of unmatched candidates in *West* (the same for *East*) is  $x$ , while  $y$  equals the number of matched candidates  $i$  in *West* where  $i - 1$  and  $i + 1$  also in *West* are both candidates and both unmatched, and  $z$  is the same as  $y$ , but for *East* instead of *West*.

	(0,0)	(0,1)	(1,1)	(0,2)	(1,2)	(2,2)	(0,3)	(1,3)	(2,3)	(3,3)
1	1	X	X	X	X	X	X	X	X	X
2	2	2	2	X	X	X	X	X	X	X
3	0	0	0	0	0	0	X	X	X	X
4	1	2	2	2	2	2	2	2	2	2
5	3	4	4	0	0	0	0	0	0	0
6	0	0,1,6	6	0,1,2	2	2	2	2	2	2
7	2	2,3,4	4	2,3,4	4	0	4	4	0	0
8	0	0	0,1,6	0	0,1,6	0,1,2	0,1,6	6	2	2
9	2	2	2,3,4	2	2,3,4	0,2,3,4	2,3,4	4	0,4	0,4
10	0	0	0,1,6	0	0,1,6	0,1,2,6	0	0,1,6	0,1,2,6	2,4,5,6
11	2	2	2,3,4	2	2,3,4	2,3,4,5,8	2	2,3,4	2,3,4,5	4,5
12	0	0	0	0	0	0	0			
13	2	2	2							
14	0									

Table 5.1: Some known results for Stubborn Matchmaker. The left-hand side of the table lists the number of unmatched pairs. Above, the first number in the pair is the minimum number of loners, while the right is the larger number of loners. Listed numbers assume the board has no crosses.

We use this classification to make our evaluation more simple. Previously, it had been necessary to save the evaluation of every game, to speed up the already-exponential-time process. Unfortunately, the solution required exponential space. In storing our numbers based only on classification, this lowers this space requirement substantially.

Unfortunately, it also means that many classes contain games with different numbers. Although we don't worry if two games are off by a parity of 1 (uncrossing edges will not change the classification), there are plenty of examples of classifications with such numbers as 1,2,6 and 7 or 2,3,4 and 5.

Based on the results collected so far, we see the following pattern: for most classes where  $(x - \max(y, z))$  is small ( $< 4$ ), the game either has number 0 or number 2 (with no crosses). Determining which of the two is a simple parity computation! Also, once this difference becomes large enough, (roughly  $> 7$ —we are still running tests) it appears that the same is true, but with the parity reversed. Game positions either have parity 0 or 2, and the same computation (but with opposite results) reveals the actual number!



	(0,4)	(1,4)	(2,4)	(3,4)	(4,4)	(0,5)	(1,5)	(2,5)	(3,5)	(4,5)
5	0	0	0	0	0	X	X	X	X	X
6	2	2	2	2	2	2	2	2	2	2
7	0	0	0	0	0	0	0	0	0	0
8	0,1,2	2	2	2	2	2	2	2	2	2
9	2,3,4	4	0	0	0	4	4	0	0	0
10	0	0,1,6	0,1,2	2	2	0,1,6	6	6	2	2
11	2	2,3,4	0,2,3,4,8	0,4,8	0	2,3,4	4	0,4	0,4	0
12	0					0				2

Table 5.2: More known results for Stubborn Matchmaker. The notation is the same as in the previous table, but we start with five free pairs, as boards with less do not exist at this size.

	(5,5)	(0,6)	(1,6)	(2,6)	(3,6)	(4,6)	(5,6)	(6,6)	(0,7)	(1,7)
6	2	X	X	X	X	X	X	X	X	X
7	0	0	0	0	0	0	0	0	X	X
8	2	2	2	2	2	2	2	2	2	2
9	0	0	0	0	0	0	0	0	0	0
10	2	0,1,2	2	2	2	2	2	2	2	2
11	0	2,3,4	2,3,4	0	0	0	0	0	4	4
12	2	0	0,1,6				2	2	0,1,6	6

Table 5.3: More known results for Stubborn Matchmaker. The notation is the same as in the previous table, but we start with six free pairs, as boards with less do not exist at this size.

Even if this pattern continues, however, this does not solve the entire problem. Indeed, for games with the difference inside that stripe,  $4 \leq (x - \max(y, z)) \leq 7$ , classes have a variety of number options. Also, since each of these classifications,  $(x, y, z)$  has game states with children in  $(x - 1, y - 1, z - 1)$ , also in that stripe, we still may have to perform an exponentially-large search through the stripe in order to determine the number of a game. It appears that in order to determine the number of the game, further analysis will have to be done for games within this stripe!

So far, we have determined many number possibilities, based on our classifications. Our results so far are displayed in Tables 5.1 through 5.4. This data, however, is based off some of the conjectures made earlier in this chapter. Trying to use the brute-force method of

	(2,7)	(3,7)	(4,7)	(5,7)	(6,7)	(7,7)	(0,8)
8	2	2	2	2	2	2	2
9	0	0	0	0	0	0	0
10	2	2	2	2	2	2	2
11	0	0	0	0	0	0	0
12	2,6	2	2	2	2	2	0,1,2

Table 5.4: More known results for Stubborn Matchmaker. The notation is the same as in the previous table, but we start with eight free pairs, as boards with less do not exist at this size.

evaluating requires exponential resources to calculate the numbers.

In order to escape this issue (and get the results we have in the tables), we began to develop two more programs. The first of these, `StubbornMatchmakerNimberizer`, is an evaluator which determines the number of a game without looking at the child values. In order to perform this evaluation, we look for patterns within the classifications, then hard code them into this evaluator. Currently, this evaluator is imperfect, as we do not yet know how to determine the number of any game. We continue to improve this evaluator by adding classifications which it knows the rules to.

In order to check the validity of the `Nimberizer`, we also employ a `StubbornMatchmakerNimberizerVerifier`, which verifies that the `StubbornMatchmakerNimberizer` correctly evaluates a class by recursively checking the classes of children of games within that class. It keeps track of classes verified and stops when it reaches a class which is not correctly evaluated by the `Nimberizer`. With this method, we can continue to write an evaluator without having to store the correct numbers of all descendants. This greatly increases both our ability to analyze classes on larger game boards as well as our attention to potential patterns in coding an evaluator. Hopefully, as we gain a greater understanding of this game, we will be able to find an efficient evaluator for any game state.

## Chapter 6

# “Making it real”

Half of the adventure for these combinatorial games lies in actually playing them. In order to do this, we create both virtual and actual game boards in order to play the games.

### 6.1 Atropos

The first virtual board for Atropos was created in a Java applet, available to play online (see Figure 6.1). Fortunately, Atropos extends naturally to larger game boards, a facet we exploit in our implementation; the size of the triangle is changeable from one game to the next. This presented a bit of an issue in the coding problem: how do we generalize the boundary colors for boards of any size? Not just any alternation of the “right” colors for one side provides the actual desired game board.

#### 6.1.1 Correctly Coloring the Border

Sperner’s Lemma requires that along each of the sides, one of the three colors cannot be played (without losing the game). We want to disallow blue on the left-hand side of the triangle, red along the right-hand side and green along the bottom. Thus, we color each of those sides with alternating circles of the other two colors, directly enforcing Sperner’s rules. The order of this alternation matters, however. If, for example, we colored the

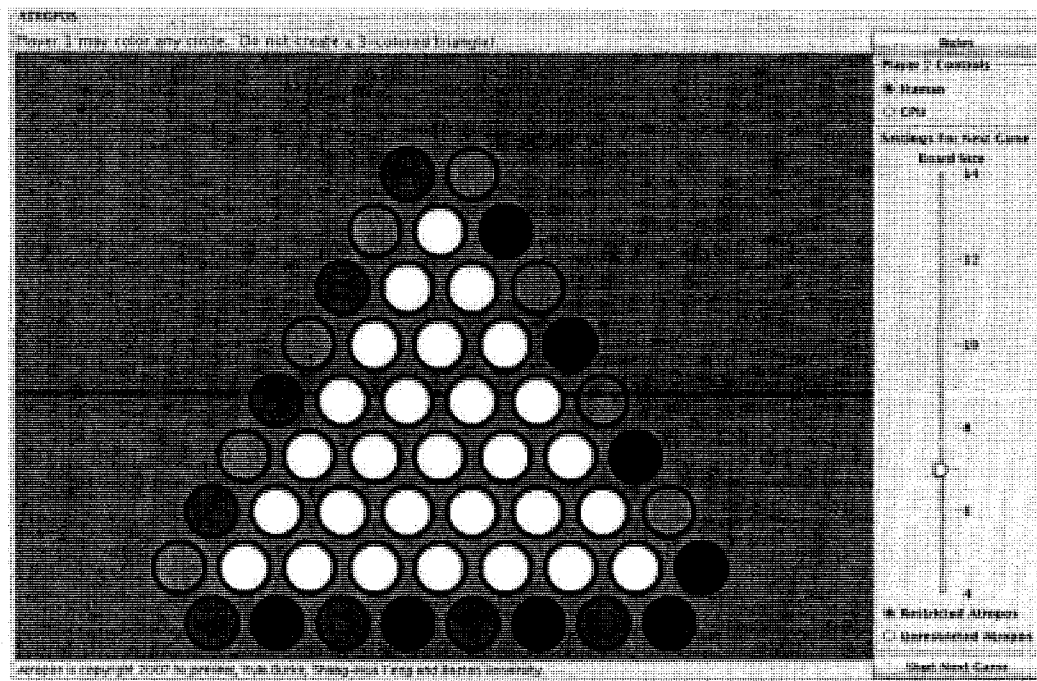


Figure 6.1: The online Atropos applet.

bottom (left-to-right) starting with blue, and colored the left-hand side (bottom-to-top) starting with green, suddenly no color may be played in the bottom-left corner. While it is true that we want to disallow blue and green, a player should be allowed to begin the game by playing red here!

We can remedy this by forcing each corner to have at least one of the two boundary circles painted the “allowed” color (for that corner). Enforcing this proves to be the major roadblock in our board designs.

For our online implementation, we clear this hurdle by choosing to start coloring our boundary edges from the correct place. For the bottom edge, we paint left-to-right starting with red, the allowed color for that bottom left corner. For the right-hand edge, we paint bottom-to-top starting with blue, the allowed color for the bottom right corner. For the left edge, we are less careful, but it turns out not to matter; we color from the bottom, starting

with green. Since we start on the left with green and on the right with blue, in any given horizontal row, either the left boundary circle will be green or the right boundary circle will be green. Thus, at the very top of the triangle, we can never have the configuration with the top colored red and blue, and the two boundary circles below them both green (this is the only configuration where no colors can be played in the top open circle on the game board). The implementation, available at <http://cs-people.bu.edu/paithan/atropos/> as a Java applet, demonstrates the coloration choice for the boundary.

### 6.1.2 Borders in the Physical Prototype

In addition, we created a physical prototype of a game board, pasted to poster board. Here, players play colored poker chips on uncolored (white) circles for their moves. (Instead of using the color green, we use yellow, since this color poker chip is easier to find.) We encountered the same problem: creating a board that could be resized between games. To do this, we provide a large Sperner Triangle with the boundaries colored properly, but also with all but the top four horizontal rows also colored as though they were the bottom row. Without any other props, the game may be played as though on a size-four board. In order to play on a larger board, white poker chips are provided; the number of extra rows desired are added by covering the appropriate pre-colored circles with the white poker chips. (See Figure 6.2 for the color layout, and Figure 6.3 for a photo of the actual prototype.) In this way, one physical board can be used for many different sizes of games of Atropos!

### 6.1.3 Extensions to the Physical Game

In the search for opponents with which to enjoy Atropos, we find many people uninterested in trying to analyze numerous turns ahead. Thus, we added an element of chance to the game, allowing players who lose to assign some of the blame to the randomness.

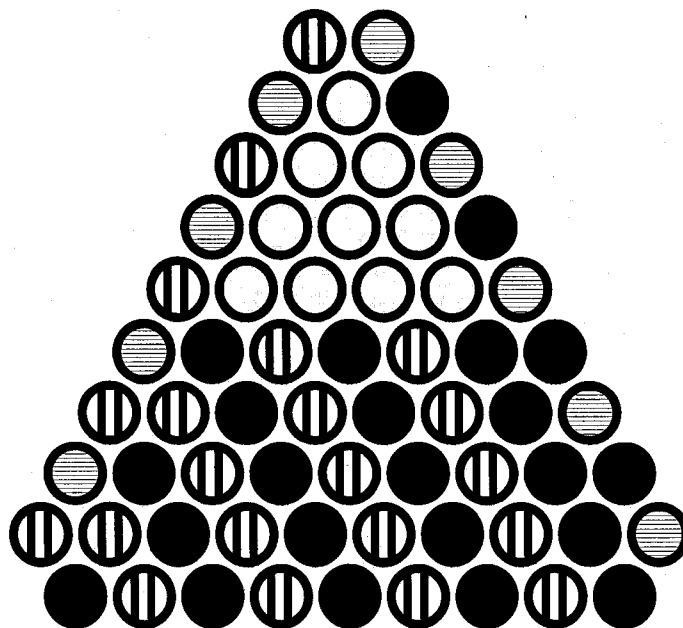


Figure 6.2: Layout of coloration for prototype board. We can cover inside rows with white poker chips to increase the size of the board.

#### 6.1.4 Atropos with Dice

Initially, we included dice to the game, with each side colored with some combination of red, blue and yellow circles. In order to create these dice, I purchased blank white and ivory colored dice, then drew circles on the sides with paint markers. The first of these uses a six-sided die with the coloration  $\{R, B, Y, RB, RY, YB\}$  (one side with a red circle, one with a blue circle, one with a yellow circle, one with red *and* blue, one with red and yellow and one with yellow and blue).

**Definition 6.1.1 (Atropos with a Die)** *Atropos with a Die is the same game as Atropos, except that before each turn, a player rolls a die, then must play one of the colors showing on that side. The die used throughout the game is chosen before the game starts*

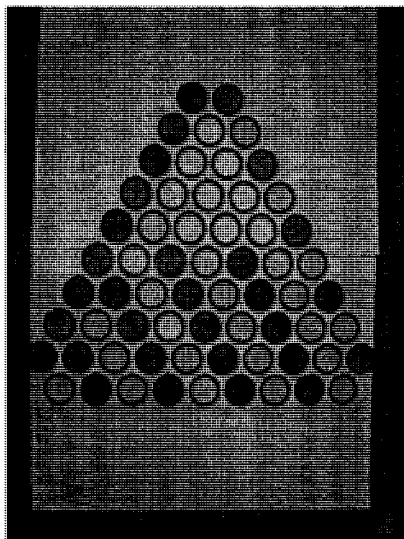


Figure 6.3: Photo of the actual prototype.

*(see the below list for dice used in the game).*

We found some success in this; some players were far more willing to play the die version than regular Atropos. I bought more dice and painted their sides differently. At this point, we have the following die schemes:

- 4-sided:  $\{RB, RY, YB, RYB\}$
- 6-sided:  $\{R, B, Y, RB, RY, YB\}$
- 6-sided:  $\{R, R, B, B, Y, Y\}$
- 8-sided:  $\{R, B, Y, RB, RY, YB, RYB, RYB\}$
- 10-sided:  $\{R, B, Y, RB, RB, RY, RY, YB, YB, RYB\}$
- 12-sided:  $\{R, R, B, B, Y, Y, RB, RY, YB, RYB, RYB, RYB\}$
- 20-sided:  $\{RBY, RBY, RY, RY, RY, RB, RB, RB, YB, YB, YB, R, R, R, B, B, B, Y, Y\}$

Aside from the inherent fun in rolling dice, this game also acts as a canvas for creating simple examples for probabilistic analysis. I have used this as a method for teaching basic probability to students. This can be done by playing the game until you reach a potentially dangerous situation. For example, the only playable circle might be a position where blue needs to be played; either yellow or red will create a three-colored triangle. One can look at the die ( $\{R, B, Y, RB, RY, YB\}$ , for example) and ask the question, “What is the probability that the current player will lose this turn?” before the die is thrown (in this case, that probability is .5). This can lead to more complex examples where you can ask the question, “What is the probability that if the current player does not lose, the next player will lose?” Here that value may rely on the choices that come up on the die, which can change the choices of the next player.



Figure 6.4: An example for the dice version of Atropos. If the next play must take place in the top-most open circle, we can analyze the probability a player losing in the next few turns.

As an example, see Figure 6.4. If we are using the  $\{R, B, Y, RB, RY, YB\}$  die, and the next play must be made in the top-most circle in the diagram, then already the next player has a chance of losing. Playing either  $B$  or  $R$  will suffice, but a play of  $Y$  will force the player to lose. Thus, the current player has a  $\frac{1}{6}$  chance of losing this turn. If the current player colors  $B$ , the next player must roll either  $Y$  or  $B$  to be able to play safely. Otherwise, if instead they play  $R$ , the next player must roll  $B$  to survive. Thus, the current player would prefer to color the next circle  $R$ , which three of the six sides ( $R, BR, RY$ ) of the die allow. For two of the sides ( $B, YB$ ) they will instead play  $B$ .

If the current player plays  $R$ , the next player must roll  $B$  and thus has three chances ( $B, YB, RB$ ) to survive and thus loses with probability  $\frac{1}{2}$ . If the current player plays  $B$ , the next player may roll either  $Y$  or  $B$  and thus loses only if they roll the side  $R$ , which



occurs with probability  $\frac{1}{6}$ .

Thus, in total:

$$\begin{aligned} \Pr[\text{Current Player loses}] &\geq \Pr[\text{Current Player loses this turn}] \\ &= \Pr[\text{Current Player Rolls side 'Y'}] \\ &= \frac{1}{6}. \end{aligned}$$

And

$$\begin{aligned} \Pr[\text{Current Player Wins}] &= \\ &= \Pr[\text{Next Player loses}] \\ &\geq \Pr[\text{Next Player loses next turn}] \\ &= \Pr \left[ \begin{array}{l} \text{Current Player does not lose this turn AND} \\ \text{Next Player loses next turn} \end{array} \right] \\ &= \Pr \left[ \begin{array}{l} \text{Current Player does not lose this turn AND} \\ ((\text{Current Player Plays 'R' AND Next Player loses next turn}) \text{ OR} \\ (\text{Current Player Plays 'B' AND Next Player loses next turn})) \end{array} \right] \\ &= \frac{5}{6} \left( \frac{1}{2} \frac{1}{2} + \frac{1}{3} \frac{1}{6} \right) \\ &= \frac{55}{216} \\ &> \frac{1}{6}. \end{aligned}$$

This analysis suggests that the current player has an advantage, since they have a higher probability of winning next turn than losing this turn.

In tighter board situations, this analysis can be continued into further turns, but when there are too many move options the calculation quickly becomes overly complicated.

### 6.1.5 Atropos with Cards

Although we had success with dice, it was still somewhat limited. In another effort to add interest, we extended the game differently, this time to inject randomness via the use of cards. To do this, we created a deck of approximately forty-three cards. One of the cards has all three colors (red, yellow and blue) on it, while the rest have either one or two colors on them. In the deck there are 7 each of these six cards (red, yellow, blue, red-blue, red-yellow, blue-yellow). At the beginning of the game, the 3-colored card is face up, meaning

that the current player can choose to play any of the three colors. Players may change the current options by playing a card from their hand. Thus, if a player plays a card with a red and blue circle on it, whoever is taking their turn must color the next circle either red or blue. If this would force them to make an illegal play, they lose the game.

More formally, the rules are as follows.

**Definition 6.1.2 (Atropos with Cards)** *Atropos with Cards uses the same basic rules as Atropos, except that we add the use of cards (colored as described above). Rules for including these cards are as follows. At the beginning of the game, all players begin with one randomly-dealt card in hand. The three-colored card begins as the top card in the face-up pile.*

*Each turn consists of two phases:*

- *Card-Playing: beginning with the current player and continuing in turn order, each player may either pass or play one card from their hand. Whenever a card is played, place it on top of the previous face-up card. If all players pass, or if a card is played and then all other players pass, this phase ends.*
- *Coloring: the current player chooses a color displayed on the top-most face-up card and plays in a legal uncolored circle (this changes depending on whether the restricted or unrestricted variant is being used). If that player creates a bad triangle with this coloring, that player loses the game. Otherwise, the current player draws a card and play continues with the next player's turn.*

This game has a nice natural extension to multiple players. If a player creates a bad triangle, they drop out and the game continues with the remaining contestants. The issue here is that Sperner's Lemma only enforces that there will be at least one bad triangle. Thus, we cannot allow the losing player to actually create the three-colored triangle. Instead, we amend the definition of the coloring phase as follows:

**Definition 6.1.3 (Multi-player Atropos with Cards)** *We attain Multi-player Atropos with Cards from Atropos with Cards, by only replacing the “Coloring” rule with the following rule:*

- *Coloring: the current player chooses a color displayed on the top-most face-up card and plays in a legal uncolored circle without creating a three-colored triangle. If this is impossible, instead that player loses the game. All remaining cards in that player’s hand are given to the player who played the last card (or discarded if that player is no longer in the game). The current player drops out of the game and play continues with the next player. If only one player is left, that player wins. If the current player does not lose, that player draws a card and play continues with the next player’s turn.*

This variant is very enjoyable and is highly recommended for casual play with friends. When played face-to-face with other people, the above play order rules become far less strict; game play is both fast and tense.

### 6.1.6 An Automated Atropos Opponent

In order to increase the functionality of the Atropos applet, an automated opponent was added. Since the game is PSPACE-complete, we implemented the simplest possible opponent: our program chooses uniformly from random amongst the possible play locations (which have a non-losing color available) then chooses a random non-losing color to play in that circle. If none of the possible play locations are safe, the program chooses to play red in one of them, losing the game. Despite the lack of strategy in this opponent, it is a good tool for learning how to trap an opponent. In the applet, only the second player can be set as automated. Thus, for the smallest size available in the applet (4, meaning four open circles along each edge) playing “against the computer” means that player two will fill in the last open circle if the game does not end sooner. In this instance, it is very simple for a player to simply survive until the triangle is full and win.

On the other hand, for board sizes of 5 and 6, if the Sperner Triangle is fully-colored

at the end of the game, player one (the human player) will color the final circle and lose. Because of this, a player in this situation must be able to force the program into a trap ahead of time. Since there is less space to work with on the board at size 5, this task is quite difficult. It becomes somewhat more possible on size 6 boards.

As the board size increases, this pattern continues. Boards of 7 and 8 open circles on a side are easier to win because they can simply survive to the end of the game. Boards of 9 and 10 are more difficult because the human player has to create a trap. (These are, however, easier than with sizes 5 and 6, since there is more open board space to build traps with.) See Section 5.2 for more on the winnability of initial boards.

In the fall semester of 2007 at Boston University, Professor Margrit Betke used Atropos as a final programming project for her Artificial Intelligence class. Students worked in teams of two to write code to play Atropos mostly using minimax techniques. At the end of the class, a tournament was run between code all fifteen groups had written.

Most of the algorithms based their evaluations on counting the number of circles adjacent to the last play. The idea here was that locations with less possible neighbors were worse to play from because they had less play options. Using this logic, it seems that many students were frustrated with the results, even when testing against a randomized opponent. This is likely because they prematurely moved into losing positions; the evaluator would rate a position with no neighbors poorly, and thus rating a move to that position highly, even if moving there was a losing play! One of the best programs, however, did use this tactic combined with the avoidance of immediate losses to evaluate plays.

### 6.1.7 Atropos Variants

Our implementation of Atropos allows players to choose between both major variants of the game: the restricted and unrestricted versions. Since Restricted Atropos is PSPACE-complete, this has been set as the default variant. Unrestricted Atropos was added after a request for another version<sup>1</sup>. Although this game may not be as challenging, it is still

---

<sup>1</sup>Dan Spielman asked for this version to play with his children.

pedagogically useful for teaching Sperner's Lemma.

All of the other settings for the game work for both versions, including the automation of the computer opponent.

### 6.1.8 Atropos for Facebook

During the 2008-2009 academic year, I enlisted the help of Boston University students Ryan Fleisher and Robert Solorio to create a version of Atropos for the social networking website Facebook. We plan to finish the application near the end of the academic year and make it available soon afterwards.

## 6.2 Matchmaker

We already have a Java applet for Stubborn Matchmaker<sup>2</sup>. Just as with our Atropos applet, this allows players to change the board size between games (see Figure 6.5). Unlike Atropos, we do not yet have a computer-based opponent.

The Matchmaker applet has some additional features as compared to the Atropos version, all of which concern changing the game settings. In Matchmaker, the settings of the game cannot be changed in the midst of a game. Thus, once the first move has been made, the settings panel is deactivated until the winning move is played. Because of this restriction, we also added an "I Give Up!" button to prematurely end games (see Figure 6.5).

### 6.2.1 Poster Matchmaker

Unlike Atropos, the mechanics behind virtual Matchmaker designs don't carry over directly to actual hard copies. The issue of a physical representation of the edges in the matching presents a problem: how can these connections be represented easily (and inexpensively)? The natural tendency towards using string presents a dual problem of lengths and knotting.

---

<sup>2</sup>The applet is available at <http://cs-people.bu.edu/paithan/matchmaker/>.

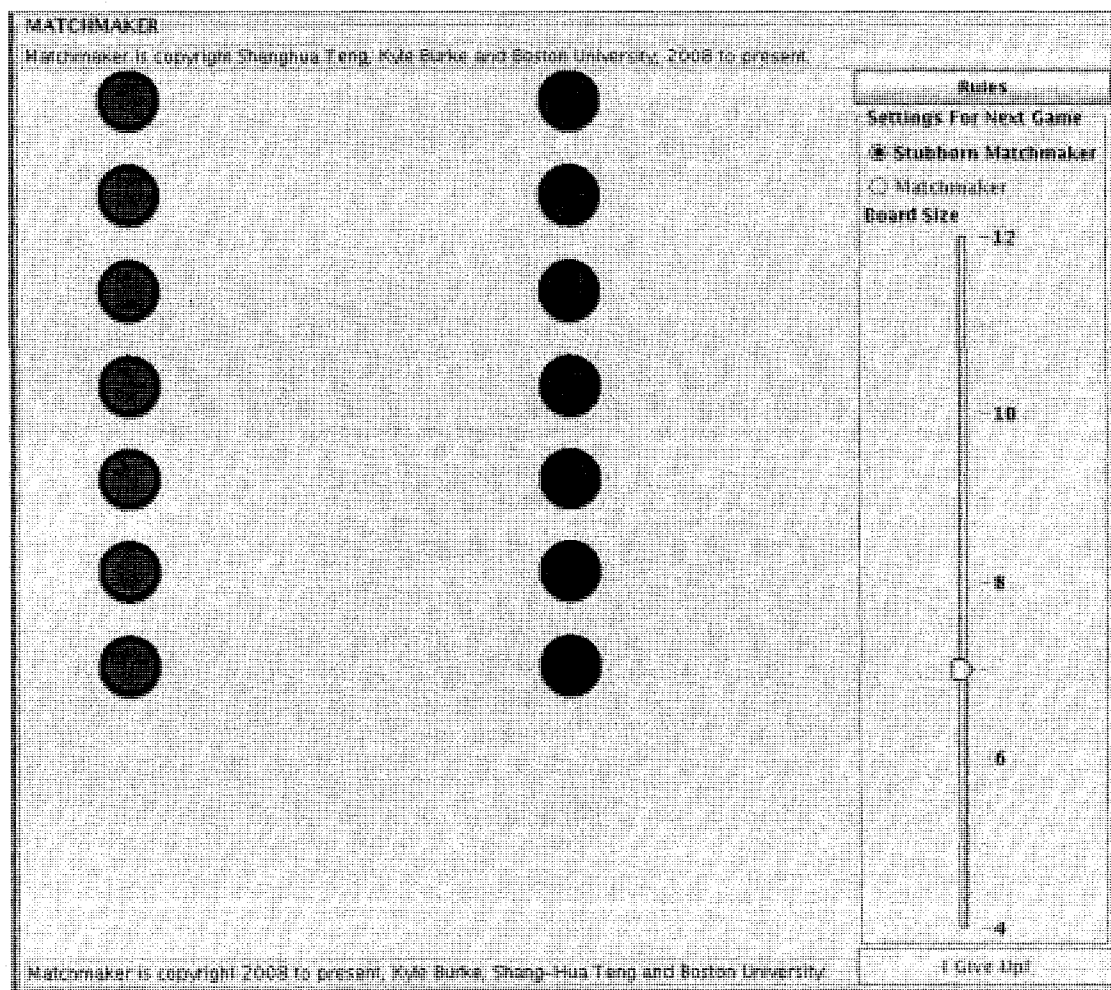


Figure 6.5: Applet for Matchmaker.

The length of the string needed between the top candidates from *West* and *East* is far less than that needed to connect the worst from *West* with the best from *East*. Thus something with far more elasticity than basic string would be necessary; the game state wouldn't be clear if the line between two nearby candidates wasn't taut and instead spaghettied through the rest of the matching. The connections also need to be easy to manipulate. String again fails this test, as it could easily become entangled in the other pieces as the matching changed.

The need to resolve these issues arose when the time came for the annual poster session in

the Computer Science department<sup>3</sup>. The previous year had been a big success with Atropos; the poster had used velcro pieces so that viewers could actually play on an empty board in the middle of the poster. We hoped for a similar interactive set up with Matchmaker.

Luckily, we found a solution to circumvent the entire connector problem. Instead of using any sort of physical connection between matched candidates, we simply printed a number of pairs of tiles, each with a different symbol or picture on them. Now, again using velcro to attach the game pieces to the poster, a pair could be matched by covering the two candidates with two identical tiles. In this way, it was still clear which matches were prevalent, despite the lack of any physical edges between the sides of the board. This allowed players to consider the strategy of a situation rather than the task of untangling the existing edges. Additionally, it adds challenge to the human task of counting crosses between the edges; it's far more difficult to see such intersections when they aren't immediately visible.

### 6.3 Dictator

In the spring of 2007, Atropos was presented as a poster for the computer science department's IAP day and in the following year, Matchmaker had the same honor. In 2009, Dictator had the chance to have a poster form, which required some logistical decisions to properly present this complex game.

Here, instead of having just one game board, a secondary board was needed for players to present their Pareto violation proofs. This aspect made it difficult to squish enough information about the game background and rules onto the poster, but the sacrifice of space paid off. In order to properly explain the game to someone, it is best to play through a sample game. In order to do that, however, the extra space is needed for novices to tinker with proof generation.

After I won the first few games at the open house, visitors quickly realized the need to spend extra time on examining potential proofs. Thus, although the actual turn plays were

---

<sup>3</sup>Boston University Computer Science hosts an annual Industrial Affiliates Program (IAP) day, part of which includes a poster session to present current research amongst the graduate students.

taking place on the game board, the lower board was often updated as a clone of the first. The idea behind this secondary board is that players could start with the current board and describe their series of IIA manipulations without changing the actual state of the game. Thus, if the chain of proof steps didn't work out, the state of the board would be retained and play could continue. In addition, the result of the lower board was malleable, and had rows of velcro instead of just single-space boxes. In this way, the effects of each IIA-manipulation could be recorded, with some candidates possibly losing their strict ordering relationships.

For novices it had the extra benefit of being a testing ground for possible manipulation moves. As visitors learned better how to use IIA in their favor, they were soon able to prove Pareto violations themselves<sup>4</sup>. This two-board model seems like a good format for the game and should likely be used in any future incarnations of Dictator.

Although it would be wonderful to have a working Dictator prototype (be it physical or another applet game) a major obstacle stands in its path. Not only is it difficult for new players to quickly grasp all the rules of the game, but it is the first of our games which is easy to play on a whiteboard. Atropos requires the initial set up of the triangular array of circles with the correct coloration of the border (which is not immediately obvious, as described in Section 6.1.1). Matchmaker, on the other hand, requires frequent changing of the edges between candidates, leading to lots of erasing and redrawing. Dictator, however, requires extremely simple initial set up and only a small amount of erasing while using IIA-manipulations. For this reason, many of the games I've participated in have taken place on a whiteboard and the need for an organized board isn't as great.

## 6.4 Summary

Games are meant to be played, and having working versions of these games has been a great motivator to study their complexity. The true value of these prototypes is the accessibility they allow for game play. Virtual prototypes have the added advantage of ease

---

<sup>4</sup>I lost two games out of about ten.



and availability of play (for our applets, only a Java-enabled web browser is needed) while physical prototypes yield more freedom during playtesting.

Moreover, the existence of playable versions of our games has added to their popularity. With the physical and virtual versions of Atropos and all the development towards injecting randomness into the game, I have played a variety of versions with different groups of people. Furthermore, having a software version of Atropos allowed for use in Margrit Betke's artificial intelligence course (see Section 6.1.6).

With all these bonuses from implementing Atropos, there is little doubt that similar advancements should be made for Matchmaker, especially for creating a useable physical prototype. Although Dictator does have the advantage of being easy to play on a whiteboard, the advantages of creating a software implementation are great enough that this is another future goal for our game development.

One of the great lessons of this work is the importance of implementing games, both for recreation and the natural and automatic study that occurs while playing.

## Chapter 7

# Pedagogical Contributions

By design, our games are furnished from problems and topics often arising in computational economics. Not only do these themes provide interesting boards for the games, they also act as an interactive arena to explore the underlying principles to these fundamentals. We use this chapter to focus solely on the potential contributions of this work towards teaching.

### 7.1 Atropos

Sperner's Lemma has recently joined the limelight in topics covered in advanced game theory and economics courses. This is due in no small part to its relevance in showing the *PPAD*-hardness of finding Nash equilibria[6][7]. In fact, in one of these presentations, Atropos is described as “an lovely pedagogical device” for teaching Sperner's Lemma [17]<sup>1</sup>.

Even a description of the rules of Atropos often inspires a new player to ask, “Wait, what happens if no one loses?” Although one can teach the lemma, it often becomes more instructive to play through an entire game of Unrestricted Atropos to show that not all the circles can be safely colored by the end of the game. Although initially we didn't have the unrestricted version playable online, we added it after it was requested. The apparent simplicity in strategies for Unrestricted Atropos does not detract from its ability to reveal

---

<sup>1</sup>Professor Scarf begins mentioning Atropos about 25 minutes into the video.

the correctness of Sperner's Lemma.

On the other hand, Restricted Atropos lacks a little in teaching the lemma; the game can end well before the board gets filled up. This can, however, be seen as part of the challenge of the game: one player can focus on just trying to fill up the entire board (this is the player who would not have to fill in the last open circle on the board). This player is relying on Sperner's Lemma to win the game, while the other player (who *would* have to play at that last circle) has to set up a trap to end the game more swiftly (while not falling into the trap themselves).

Additionally, the difference between the two versions and the apparent difference in strategy complexity can be used as a tool for describing the intricacies of reduction proofs. The proof of Theorem 4.1.3 is very dependent on the "moving restriction", evidenced in each of the widgets used.

Furthermore, as mentioned in Section 6.1.4, we can use the dice version of Atropos to explore probabilities of winning a game in the next few turns. In that section, we explored a few scenarios, but it could also be used to point out just how swiftly the calculations grow once analysis continues beyond a few turns. I have already successfully used this randomized Atropos variant as an example for explaining basic probability concepts to a student.

## 7.2 Matchmaker

Although the version of Stable Marriage reflected in Matchmaker is not the basic model of the problem, it still carries the basic idea; the only real change is the use of universal preference lists for the candidates. The idea of the stable state of the game is extremely clear after the final Matchmaker move.

Both Matchmaker and Stubborn Matchmaker also carry a lot of interesting baggage for the realm of impartial games. The analysis of game states with Sprague-Grundy results in interesting viewpoints for intuition behind numbers. For example, in *both* games, it appears

that uncrossing two edges changes the number by an xor-factor of 1. The relative intuition is that this uncrossing is the smallest possible move that can be made on a board; any uncross influences the overall parity of the game in the same way.

Stubborn Matchmaker is an interesting realm for teaching numbers, as strategies for play are not immediately obvious (even if the game winds up to be solvable in polynomial-time). Properties of the game lend simplicity as well as complexity to any analysis, each of which has ramifications on the numbers. Too many (non-Nim) examples in the current field are either too simple or too complex. Stubborn Matchmaker lies somewhere inbetween.

### 7.3 Dictator

Arrow's Theorem is an easy enough result to be able to read and reference at parties with politically-curious folk, though the details of the proof are far more hairy. The manipulation of ballots via IIA is not an easy task, and there are few venues to practice with.

Now, using the Dictator board as an arena, the importance of IIA is clear. To win, players must not only understand the statement of Arrow's Theorem, but the fundamentals of the proof techniques. Good players of Dictator will follow a proof of the theorem, even if Arrow's result is a new concept to them!

Unfortunately, unlike Atropos and Matchmaker, Dictator is a game that is likely not playable below a high-school level, and perhaps not below the college level. Atropos and Matchmaker can be played without learning the intricacies of the underlying theorems, but Dictator requires players to prove things during the game. Although this obviously has positive ramifications on understanding Arrow's Theorem, it presents difficulties in introducing players to the game. Sample situations, such as those given in Section 3.4, are instrumental in explaining Dictator to new players. Often many games must be played before novices have a firm grasp of the rules.

## Chapter 8

# Conclusions

In this chapter, we summarize the accomplishments of this dissertation. We mention both complexity and algorithmic results as well as discuss the online implementations we've created. In the end, we see that we have done a great deal of design on the three games and find the current versions of all three to be fun and challenging for players.

### 8.1 Atropos

Sperner's Lemma and its equivalence to the Brouwer Fixed-Point theorem gave us the first inspiration for a new combinatorial game, culminating in the birth of Atropos.

Atropos is a fun, original impartial game based on a colorful topic used in both Computer Science and Economics. We have two different versions of this game, one of which we proved to be *PSPACE*-complete in Theorem 4.1.1. At the same time, we have brute-force studied the winnability of initial game boards and begun to see a repeating pattern, as described in Section 5.2. This pattern is reminiscent of the repeating pattern seen in initial Sprouts configurations [13], although the general complexity of Atropos is known.

In addition to the standard game, we have many variants of Atropos, suitable both for instruction (Unrestricted Atropos) and competitive play (randomized versions). Our randomized variants offer public randomness (Die version, see Section 6.1.4) as well as

imperfect information and rules for more than two players (Atropos with Cards, see Section 6.1.5). These games are all enjoyable, while still carrying the spirit of the original game.

Additionally, for both the standard and unrestricted versions, we provide an online Java applet. This program contains a simple randomized computer opponent which still provides a challenge, especially for novice players.

## 8.2 Matchmaker

Matchmaker is an original impartial game played using the fundamentals of the Stable Marriage problem. The rules are simple and easy to learn. Lemma 4.3.1 shows that Full Matchmaker is solvable in polynomial time. Although this is not the version often played, the other two major variants become instances of Full Matchmaker at the end of the game.

Although the complexity of these other two remains unsolved, we have studied both extensively. To do this, we developed our own Java packages to calculate the numbers of impartial games. These classes are very reusable and can be used with any impartial game (even misere games). Although this code requires an exponential amount of time to perform the number evaluation, we were able to use this data to see the patterns described in Section 5.3 in Stubborn Matchmaker.

To continue analysis on Stubborn Matchmaker, we first created an applet to play it. After making this available online, we upgraded our analysis tools, writing more code that was tailored to evaluate this game. These programming tools are a great resource as our computer-assisted evaluation of Stubborn Matchmaker continues.

## 8.3 Dictator

Teaching the fundamentals of Arrow's Theorem is at the heart of Dictator. This third new impartial game, Dictator has two different challenges that take place each turn. On one hand, players must struggle to try to find a complex proof to show their opponent's violation. Even if they succeed in this, they must still try to determine a strategy to avoid

such a violation themselves.

The dual-nature of these two complexities gives us the first chance to explore complexities beyond the usual *PSPACE*-complete/polynomial-time dichotomy. Though in Lemma 4.4.2, we show that Basic Dictator is solvable in polynomial-time, such a result is not at all clear for standard Dictator. Here we have two separate complexity issues: the difficulty in determining whether there is a violation in the current board (this problem is in *NP*) and the difficulty in determining whether the current player has a winning strategy (this problem is in *PSPACE*). Even if it turns out that the game is *PSPACE*-complete, answering the other question could still be interesting!

As technology used in voting increases, allowing alternative systems to be used, a greater understanding of voting with preference lists could become necessary. Arrow's Theorem is at the heart of these systems, and Dictator uses Arrow's techniques for similar proofs. A competent player of Dictator already has a strong understanding of many of the tenets of modern voting theory.

## 8.4 Overall Conclusions

In all in this work we have defined three games and developed them to try to reach some computational hardness result in order to make them non-trivial for actual play. Atropos, as we showed, is *PSPACE*-complete. Meanwhile, the complexity of Matchmaker is not currently known, although the game play is certainly difficult for human competition. Although we do not yet have complexity results for Dictator, there are two avenues for potential hardness, both of which are exciting facets of this voting game.

On a higher level, however, we see that this design process is a legitimate tactic for developing combinatorial games. Each of these three games began with a different academic topic as inspiration, and went through many phases before reaching their current state. All three began with simple rules before undergoing rounds of analysis challenging their computational complexity. This process both results in a computationally hard game as

well as preserves the simplicity of the rules by only making small changes after each round of analysis. Thus, we continue to adhere to the rule “easy to play, hard to master” as we continue to develop these and more games.



## Chapter 9

# Future Work

In this chapter, we relate future possible avenues for study on these topics. Although this may not be an exhaustive list for inspired directions of research, it should provide good suggestions for further work. Since the games we present in this dissertation cover a wide range of topics, we hope that more readers will be interested in pursuing problems mentioned here. In this section, we relate each open problem and potentially follow it up with a conjecture, but also mention other concerns, including possible implementation plans for the games.

### 9.1 Atropos

In the standard version of Atropos, there is one great open question: which player has a winning strategy from the starting position?

**Open Problem 9.1.1** *At the starting position of an Atropos game, is there an efficient algorithm to determine which player has a winning strategy?*

We mimic the Sprouts Conjecture [2] in our own Atropos Conjecture (see Conjecture 5.2.1), except that our pattern repeats with every four size increments, instead of six. We plan to continue analysis to either support or break this conjecture.

Additionally, similar open questions remain for Unrestricted Atropos. The grandest of these is the complexity of determining which player has the win.

**Open Problem 9.1.2** *Is there an efficient algorithm to determine which of the players in the game of Unrestricted Atropos has a winning strategy?*

This problem is very intertwined with determining the winner of versions of Planar Kayles, and thus we look forward to further results on this topic. Here we hope for a negative result: that Unrestricted Atropos is solvable in polynomial time. If this happens, and the complexity turns out to be different from Atropos, then we can continue investigations by parameterizing the distance restriction between consecutive plays.

On the other hand, a result about the winnability of Unrestricted Atropos from initial positions seems likely in the near future.

**Open Problem 9.1.3** *From the initial position, does one of the players have an efficient algorithm for playing to win a game of Unrestricted Atropos.*

Due to the ease of actually playing this game, we make the following conjecture.

**Conjecture 9.1.4** *One of the two players has an efficient strategy to win a game of Unrestricted Atropos from the initial position. If the starting board has an even number of uncolored circles, then this is the first player. Otherwise, the second player has the winning efficient strategy.*

## 9.2 Matchmaker

With Matchmaker, we also have two different variants with open questions attributed to each of them.

**Open Problem 9.2.1** *Is there an efficient algorithm to determine which player has a winning strategy in a game of Matchmaker?*

This is the motivating problem for all our work on Matchmaker. This led to the creation of Stubborn Matchmaker in the hope of obtaining an easier game with only a slight tweak of the rules. Analysis of Stubborn Matchmaker, however, is also unfinished.

**Open Problem 9.2.2** *Is there an efficient algorithm to determine which player has a winning strategy in a game of Stubborn Matchmaker?*

Although we are uncertain about standard Matchmaker, we are prepared to conjecture about the complexity of Stubborn Matchmaker.

**Conjecture 9.2.3** *There exists an efficient algorithm to determine which player has a winning strategy in a game of Stubborn Matchmaker.*

Whether or not a result here would yield a speedy result for Matchmaker is unclear, but it seems that solving Stubborn Matchmaker will happen before Matchmaker.

In addition to these open questions, we also note that only a Stubborn Matchmaker variant is available for play on our applet. We hope to extend the implementation to include the standard Matchmaker game.

### 9.3 Dictator

Dictator only has one variant in which we have a lot of interest. Here, however, there are many interesting open problems, dealing with the two separate facets of the game: choosing good moves and proving violations. As with Matchmaker, we have the grand problem of the overall game complexity.

**Open Problem 9.3.1** *Is there an efficient algorithm to determine which player has a winning strategy in a game of Dictator?*

It could also be that the complexity of choosing where to play each turn is different than the overall strategy. Thus, the complexity could hinge either on making the best move on the board, being able to efficiently see Pareto violation proofs or some combination of

both. We have already showed that determining whether a proof exists is in  $NP$ , though we haven't yet narrowed down the complexity further.

**Open Problem 9.3.2** *What is the complexity of determining whether a proof of a Pareto violation exists on a Dictator board?*

A hardness result for either of these problems would be preferable and would give us a difficult game! Since Dictator is still very young, even a guess at the complexity of this game is unclear and thus we have not yet made any conjectures.

Aside from these open problems, we still do not have a software implementation of Dictator. This is a big goal for the future, as such a program will be difficult to design, both as far as the interface is concerned as well as the underlying regulation of legal options for a player during their turn. It is likely that initial versions of this software will not regulate the construction of proofs but just be used to maintain the basic game board.

# Bibliography

- [1] M. H. Albert, R. J. Nowakowski, and D. Wolfe. *Lessons in Play: An Introduction to Combinatorial Game Theory*. A. K. Peters, Wellesley, Massachusetts, 2007.
- [2] D. Applegate and D. Sleator G. Jacobson. Computer analysis of sprouts. Technical Report CMU-CS-91-144, Carnegie Mellon University Computer Science, 1991.
- [3] K.J. Arrow. A difficulty in the concept of social welfare. *Journal of Political Economy*, 58(4):328–346, 1950.
- [4] Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. *Winning Ways for your Mathematical Plays*, volume 1. A K Peters, Wellesley, Massachusetts, 2001.
- [5] Kyle W. Burke and Shang-Hua Teng. A PSPACE-complete Sperner triangle game. In Xiaotie Deng and Fan Chung Graham, editors, *Workshop on Internet and Network Economics*, volume 4858 of *Lecture Notes in Computer Science*, pages 445–456. Springer, 2007.
- [6] X. Chen and X. Deng. Settling the complexity of two-player Nash equilibrium. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, Berkeley, California, 2006. IEEE.
- [7] X. Chen, X. Deng, and S.-H. Teng. Computing Nash equilibria: Approximation and smoothed complexity. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, Berkeley, California, 2006. IEEE.

- [8] Rudolf Fleischer and Gerhard Trippen. Kayles on the way to the stars. In H. Jaap van den Herik, Yngvi Björnsson, and Nathan S. Netanyahu, editors, *Computers and Games*, volume 3846 of *Lecture Notes in Computer Science*, pages 232–245. Springer, 2004.
- [9] David Gale. A curious nim-type game. *American Mathematical Monthly*, 81:876–879, 1974.
- [10] David Gale. The game of Hex and the Brouwer fixed-point theorem. *American Mathematical Monthly*, 10:818–827, 1979.
- [11] David Gale and Lloyd Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 1:9–14, 1962.
- [12] P. M. Grundy. Mathematics and games. *Eureka*, 2:198—211, 1939.
- [13] J. Lemoine and S. Viennot. Records for n-spots games, 2007. <http://sprouts.tuxfamily.org/wiki/doku.php?id=records>.
- [14] C. H. Papadimitriou. *Computational Complexity*. Addison Wesley, Reading, Massachusetts, 1994.
- [15] J. Purinton and R. Khorkov. A new verified misere outcome, 2009. <http://www.geocities.com/chessdp/article2009001>.
- [16] S. Reisch. Hex ist PSPACE-vollständig. *Acta Inf.*, 15:167–191, 1981.
- [17] Herbert Scarf. Presentation: Fixed point theorems and economic equilibrium, 2008. [http://www.as.huji.ac.il/schools/econ19/media/econ1\\_5.mov](http://www.as.huji.ac.il/schools/econ19/media/econ1_5.mov).
- [18] Thomas J. Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16(2):185–225, 1978.
- [19] E. Sperner. Neuer Beweis für die Invarianz der Dimensionszahl und des Gebietes. *Abhandlungen aus dem Mathematischen Seminar Universität Hamburg*, 6:265–272, 1928.

- [20] R. P. Sprague. Über mathematische Kampfspiele. *Tôhoku Mathematical Journal*, 41:438—444, 1935-36.
- [21] D. Zeilberger. Chomp, recurrences and chaos. *Journal of Difference Equations and Applications*, 10:1281 – 1293, 2004.

# Curriculum Vitae

**Kyle Webster Burke**

<http://cs-people.bu.edu/paithan/>  
paithan@cs.bu.edu

## Home Address

26 Calvin St. Apt 3  
Somerville, MA 02143  
(617) 312-2485

## Work Address

Boston Univ. Computer Science  
111 Cummington St.  
Boston, MA 02215  
(617) 312-2485

## Education **Boston University**

2003-2009

Ph.D. candidate in Computer Science  
Research Advisor: Shang-Hua Teng, Ph.D.

**Boston, MA**

## **Colby College**

1999-2003

B.A. in Computer Science (with Honors) and Mathematics (with Recognition). Graduated Phi Beta Kappa and Summa Cum Laude (GPA: 3.76), May 2003.

**Waterville, ME**

## Employment **Boston University**

2003-2009

Lecturer: Combinatoric Structures (Fall 2007) and Geometric Algorithms (Spring 2008 and Spring 2009).

Teaching Fellowship: Geometric Algorithms (Spring 2007).

IGERT Trainee, 2005-2008.

GAANN Fellow, 2003-2005.

**Boston, MA**

## **Colby College**

2000-2003

Teaching Assistant: Multivariate Calculus, Linear Algebra.

Lab Assistant: Introduction to Programming, Weaving the Web.

**Waterville, ME**

## Publications

Burke, K. and S. H. Teng. A PSPACE-complete Sperner Triangle game. Workshop on Internet and Network Economics, 2007, pages 445-456.



Bretscher, O. and K. Burke. Linear Algebra with Applications Instructor's Solutions Manual. Pearson Education, 2005.

Smith, M. L, C. E. Hughes, K. Burke. The Denotational Semantics of View-Centric Reasoning. Communicating Process Architectures - 2003. The University of Twente (Enschede, the Netherlands). September 7-10, 2003.

Burke, K. and M. L. Smith. Extending CSP to Investigate Linda Ambiguities. Journal of Computing in Small Colleges, Vol. 18, Issue 5, 2003.

## Presentations

"A Sperner Triangle Game", Combinatorics Seminar, Dartmouth College, 12/2008; also delivered at interviews during Fall 2008 and Spring 2009 semesters.

"A Game on the Sperner Triangle", Mathematics Colloquium, Colby College, 02/2007; Center for Computational Sciences Colloquium, Boston University, 04/2007; 3rd Annual Workshop on Internet and Network Economics (WINE) 12/2007

Regular contributor to departmental Theory Seminar. Topics presented: "The Linial-Saks Decomposition Algorithm", 10/2004; "MAX-CUT Approximation Algorithm and Hardness of Approximation Result", 10/2005; "A Group-Theoretic Approach to Matrix Multiplication", 2/2006; "Games on the Sperner Triangle", 11/2006

Regular contributor to annual departmental Industrial Affiliates Program poster session. Topics presented: "Optimal Aggregation Location of Grid Sensor Networks" (2004); "Building Low-stretch Spanning Trees" (2006); "A Game on the Sperner Triangle" (2007); "Matchmaking: Stable Marriage Game" (2008)

"Optimal Aggregation Location of Grid Sensor Networks" poster presented at Sensor Network Technical Workshop hosted by B. U. Center for Information and Systems Engineering, 05/2004.

## Awards

Awarded Advanced Computation in Engineering and Science (ACES) training for 9/2005 - 8/2007 by Center for Computational Sciences, Boston University.

Awarded Graduate Assistance in Areas of National Need (GAANN) fellowship for 9/2003 - 8/2005 by Boston University Computer Science department.

Won Best Poster award at Boston University Computer Science department Industrial Affiliates Program, 2007 for "Games on the Sperner Triangle".