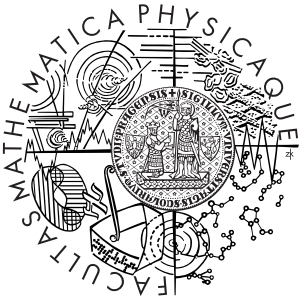
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Jiří Vejmola

Umělá inteligence pro moderní karetní hry

Artificial Intelligence for modern card games

Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: Mgr. Ondřej Sýkora
Studijní program: Informatika, programování

2010

Acknowledgements

I would like to thank my supervisor, Mgr. Ondřej Sýkora, for his guidance. Our meetings and his suggestions and feedback were always helpful. I would also like to thank him and Mgr. Jiří Iša for Seminar on Applied Artificial Intelligence which gave me the opportunity to try some of the techniques in practice. Next, I would like to thank Mgr. Cyril Brom, Ph.D., whose Lecture on Human-like and Animal-like Agents gave me a broader introduction to artificial intelligence in games.

I feel obliged to thank all the people involved in developing Magic: The Gathering. It wouldn't be such a great game without them.

Finally, I would like to thank my family and friends for their support and understanding. They let me work when I needed to and provided me with other activities to keep me sane.

I hereby claim that I have written this bachelor thesis on my own, using exclusively cited sources. I permit the lending of the thesis.

Prague, August 6, 2010 Jiří Vejmola

Contents

List of Figures

List of Tables

Název práce: Umělá inteligence pro moderní karetní hry
Autor: Jiří Vejmola
Katedra (ústav): Katedra teoretické informatiky a matematické logiky
Vedoucí bakalárské práce: Mgr. Ondřej Sýkora
e-mail vedoucího: ondrasej@centrum.cz

Abstrakt: V předložené práci studujeme možnosti implementace umělé inteligence pro moderní karetní hry. Naší snahou je vytvořit různé hráče a na základě jejich vzájemných zápasů si ověřit efektivitu použitých metod, např.: simulace, náhodný přístup, modelování protivníka, daná strategie, atd. Dále bychom rádi zjistili, jaký vliv mají ostatní herní prvky na průběh hry a výkon jednotlivých hráčů. Vycházíme z pravidel karetní hry Magic: The Gathering pro dva hráče, konkrétně z edice Magic 2010 Core Set, kde došlo k několika důležitým změnám. Tyto pravidla jsou dále upraveny nebo zjednodušeny pro potřeby implementace herních mechanik na počítači. Balíky karet použité při testování jsou založeny na již existujících.

Klícová slova: umělá inteligence, karetní hry, Magic: The Gathering, simulace, minimax

Title: Artificial Intelligence for modern card games
Author: Jiří Vejmola
Department: Department of Theoretical Computer Science and Mathematical Logic
Supervisor: Mgr. Ondřej Sýkora
Supervisor's e-mail address: ondrasej@cetrum.cz

Abstract: In this work we study the ways of implementing artificial intelligence for modern card games. We strive to create various players, to put them into matches against each other. By doing so we can verify the efficiency of employed methods, such as simulation, random approach, opponent modeling, fixed strategy, etc. We would also like to find out what influence do other game elements have on the progress of the game and the performance of individual players. The game used in the thesis is based on Magic: The Gathering's rules for two players, specifically on Magic 2010 Core Set, which revised certain parts of the game. These rules are then modified or simplified when necessary for implementing game mechanics on a computer. Decks of cards used in testing are based on the existing ones.

Keywords: Artificial Intelligence, card games, Magic: The Gathering, simulation, minimax

Chapter 1

Introduction

The term "Artificial Intelligence" (AI) covers a wide area of usage in many fields of study. To name a few, there are neural networks, planning, machine learning, path-finding, and much more. Different tasks require different techniques and there are many of both. Computer games and card games are just a small part of the whole, much bigger area. To not cause any confusion, whenever we mention AI or AI techniques, we will mean only the part related to our work and nothing else.

Most card games are set in multi-agent environments with imperfect information and uncertainty. Many classic card games have already been studied. Poker in particular is very popular and it is still of interest to many [5][6][7]. We wanted to do something different, so we decided on Magic: The Gathering (MTG). The game is not even 20 years old and is still expanding.

MTG can be described in fantasy terms and the stories behind the cards and the game itself support it as well. Each player represents a powerful sorcerer. He uses mana, which is his magical energy and resource, to cast spells, summon creatures, and use magical relics. All those things help him to achieve his goal: to defeat his opponents. The bigger territory the player controls, the more mana he has at his disposal, and the more powerful things he can get under his command.

MTG is a trading card game, which means that collecting cards is an important aspect of the game. Players use cards from their collection to build their own unique decks of cards. Even though trading cards and building a deck from available cards are important parts of the game, neither of them will be studied.

Our intention is to create an environment that would implement the core rules of MTG. We will create a series of agents that will use different techniques to play and make decisions. Among those should be one that would decide randomly, one that would use a simple reasoning, and one that would use more advanced or complex techniques. We will then let them duel each other under different circumstances. We will use the results of their matches to compare them and to propose future improvements.

There is also the question of *Skill vs. Luck*. One card can often make a difference between victory and defeat if player knows how to use it. We will provide the same starting circumstances to all agents to better measure their skill. We will then use the same setting and swap the agents to minimize the influence of

luck.

The remaining structure of the thesis is as follows: Chapter 2 gives an overview of the rules of Magic: The Gathering. Chapter 3 discusses the capabilities of the environment we created. Chapter 4 presents what should each agent be able to do and gives a more detailed look at the agents we created. Chapter 5 describes the experiments performed to evaluate the agents and the results we got. Chapter 6 discusses the results we got and the possibilities of future work. Chapter 7 summarizes what we learned. Appendix A contains decklists of the decks we used in experiments. Appendix B describes the contents and structure of the attached CD.

Chapter 2

Magic: The Gathering

In this chapter we talk about the game in general. We then focus on its rules. We summarize them to give rules necessary for understanding the game and this work. We will only explain parts necessary for the thesis. Both basic [14] and comprehensive [15] rules are available from official sources [16].

2.1 Overview

Magic: The Gathering (MTG) is a trading card game being published by American company Wizards of the Coast. The first version of the game was created by Richard Garfield and released in 1993. It quickly became popular among players of role-playing games as well as other people. The essence of the game remained almost unchanged since its beginning, with only three major revisions in 1994, 1999 and 2009 [8]. These changes were done to streamline the game or to make some mechanics more understandable for casual players. They were more like cosmetic touches on a bigger scale, usually for the greater good.

For the popularity of the game also speaks the fact, that it is being expanded several times every year, when new sets of cards are released. The core sets are introductory with the size of 250 cards in average. The base sets have up to 350 cards and bring new ways how to play. The expansion sets follow the base sets and have less than 200 cards. There are many special themed editions consisting of either a handful of cards or containing whole decks to play with. Most sets have their own story for those interested in more than just playing or collecting. MTG was also released in other languages besides English; there are editions in French, German, Japanese, Russian and more.

Based on data from the Gatherer [12], there are almost 12,000 unique cards produced up to date, each with their own art. It's up to everyone interested in MTG whether they will be more of collectors or players but they will surely end up doing both to a certain extent. Players keep collecting cards so that they can use them in battles against others. They can play for fun with their friends or try to compete in tournaments. But in the end, all that matters is to have fun and enjoy the game.

2.2 Rules

According to the Basic Rulebook [14, p. 1],

> *The* Magic: The Gathering® *game is a strategy game played by two or more players, each of whom has customized deck of* Magic™ *cards. Over the course of the game, each player will take turns playing cards such as lands (which enable you to play other cards), creatures, sorceries, and other spells. Each player starts at 20 life. When you reduce your opponent to 0 life by attacking with creatures and playing spells, you win!*

Basics

There are five colors of spells. Each of them has its corresponding basic land that produces mana of its color. Each color represents a different kind of play style.

Color	Symbol	Land	Characteristics
White (The color of Justice)	{W}	Plains	Protection, Order
Blue (The color of Wisdom)	{U}	Island	Deceit, Intellect
Black (The color of Ambition)	{B}	Swamp	Decay, Death
Red (The color of Chaos)	{R}	Mountain	Fury, Chaos
Green (The color of Nature)	{G}	Forest	Life, Nature

Table 2.1: The Colors of Magic

Tapping cards is a widely used action. It is used to produce mana by lands, to attack with creatures, or to use abilities. To tap a card is to turn it sideways.

For an example of a card, along with description of its parts, see Figure 2.1.

Card Types

Every card has one or more types. The type defines when a card can be played and what its main function is. Permanent card is a card that remains in the game when it is cast.

Enchantment represents a stable magical manifestation. It affects other cards. (permanent)

Artifact represents a magical relic. It is the same as *enchantment* but usually colorless. (permanent)

Creature can attack and block. It has power and toughness. (permanent)

Land produces mana. It doesn't have to be cast. (permanent)

Planeswalker represents a powerful character from the world of Magic. It can be affected just like players. It has loyalty counters as its resource. (permanent)

Sorcery represents a magical incantation.

Instant is the same as *sorcery* but it can be cast anytime.

Tribal is a supplement type for others. It indicates what subtypes of creatures it relates to.

Figure 2.1: Card Anatomy, taken from the official MTG website [13]

Zones

Zones are the areas of play. They represent a game board. Figure 2.2 shows a possible arrangement of cards into zones. Each player has his own zone unless stated otherwise.

Library contains yet undrawn cards from shuffled deck.

Hand contains cards drawn from *library*.

Battlefield contains permanents that were cast and resolved successfully. It is shared by both players.

Graveyard is a discard pile. All discarded, destroyed, sacrificed, or countered cards go here. *Sorceries* and *instants* go here when they resolve.

The Stack stores spells and abilities that wait there to resolve. It is shared by both players.

Exile is for cards removed from the game but not destroyed. It is shared by both players.

Abilities

Abilities affect the game when they are used. This effect is either permanent or lasts for some time. There are three types of abilities:

Static abilities affect the game as long as their card is on the battlefield.

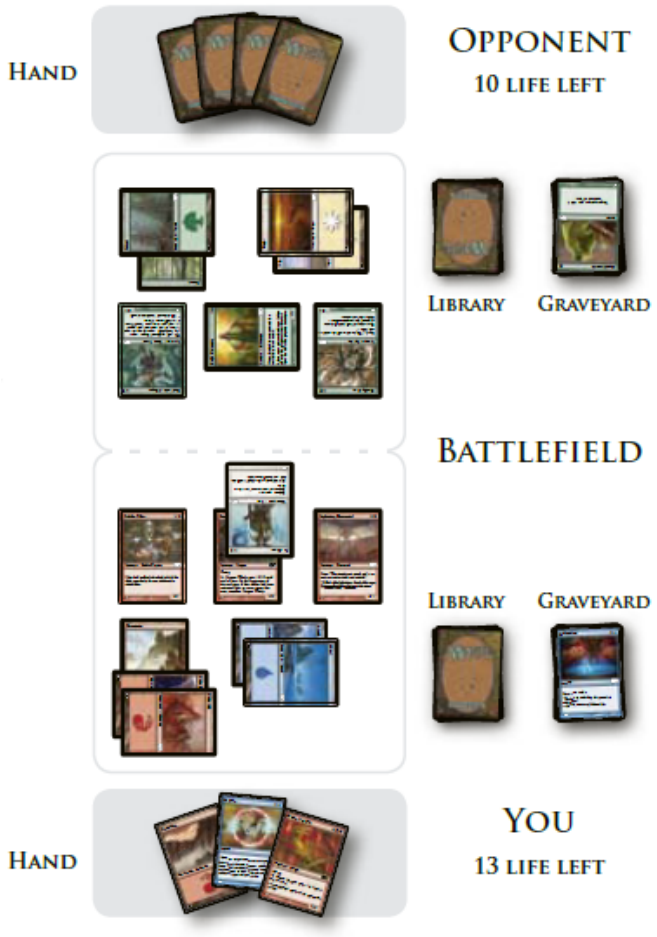


Figure 2.2: Game Zones and example of a game in progress. Taken from the Basic Rulebook [14, p. 7]; Exile *could be another pile of cards between libraries and graveyards.* The Stack *could be in the middle of the battlefield or next to it. The cards turned sideways are tapped.*

Triggered abilities happen automatically when a specific event occurs. They go to *the stack* when triggered.

Activated abilities can be used at any time, as long as a player can pay their cost. They go to *the stack* when used.

There are also *keyword abilities*. They are a special group of abilities that are so common that their description is shortened to a single word or phrase. Most of them are static abilities, but they can be triggered or activated as well.

Dynamics of a Turn

Each turn is divided into phases and steps. They are always performed in their order which can be seen in Figure 2.3. *Active player* (AP) is the player whose turn it currently is. He plays first and he is the player that can attack other players in this turn. Before the next step or phase can start, *the stack* must be empty and

both players must pass. All produced but unused mana is deleted at the end of each step and phase.

Players can cast instants and activate abilities during all steps and phases except the *Untap step* and the *Cleanup step*. Active player untaps all of his tapped permanents during the *Untap step*. He draws a card from *library* during the *Draw step*. He can cast any card and activate any ability during the two *Main Phases*. He can also put one land onto the battlefield in either of his *Main Phase*, but he is limited to one land per turn. At the beginning of *Declare attackers step*, active player decides which of his creatures will attack. At the beginning of *Declare blockers step*, attacked player decides which of his creatures will block. Both players assign the combat damage of their creatures during the *Combat damage step*. During the *Cleanup step*, active player discards enough cards from his *hand* so that he has seven at most. All damage on creatures is then removed. Nothing special happens during the steps that were not mentioned.

1. **Beginning Phase** (skipped at the beginning of the game)
   (a) Untap step (AP untaps his permanents)
   (b) Upkeep step
   (c) Draw step (AP draws a card)
2. **First Main Phase**
3. **Combat Phase**
   (a) Beginning of combat step
   (b) Declare attackers step
   (c) Declare blockers step (skipped if there are no attackers)
   (d) Combat damage step (skipped if there are no attackers)
   (e) End of combat step
4. **Second Main Phase**
5. **Ending Phase**
   (a) End step
   (b) Cleanup step

Figure 2.3: Parts of the Turn

Playing the Game

Each player starts at 20 life. They shuffle their decks and draw seven cards. They can *mulligan* if they aren't satisfied with the cards they have in hand. To do so, they shuffle their hand into library and draw one card less than they had. This can be repeated until they are satisfied with their cards. The first player starts playing according to the structure of the turn when all players agree to keep their hands.

The game uses a system of *priority* to ensure that only one player can make decisions at any given time. A player receives priority at the beginning of his steps and phases (except *Untap* and *Cleanup* steps) or when his opponent passes. When both players pass in a row and there are some items on *The Stack*, then the top item will resolve and the active player will get priority. Game proceeds to the next step if *The Stack* is empty when both players pass.

The player with priority can cast cards or activate abilities. He can only cast or active such cards and abilities, for which he can pay. Their availability also depends on the current step or phase. The played card or ability goes on top of *The Stack*.

To attack, active player announces the creatures he is attacking with and taps them. He can make them target his opponent's planeswalkers if the opponent has any. Only untapped creatures that have been on the battlefield since the beginning of the turn can attack.

The attacked player can block by assigning his creatures to the attacking ones. Only untapped creatures can block. Each creature can block only one attacker but one attacker can be blocked by multiple creatures. The attacking player sets the order in which an attacker damages its blockers. If the attacked player doesn't want to, he doesn't have to block.

With both attacking and blocking creatures declared, both players can assign the damage they deal. There is only one rule to follow when an attacker is blocked by multiple creatures. The attacking creature has to assign enough damage to the first blocking creature in line to destroy it in order to assign damage to the next one, and so on. All of the assigned damage is then dealt simultaneously.

A player wins the game if one of the following happens: *a*) he reduces the life of his opponent to 0 or less; *b*) his opponent has to draw more cards than he has in his library; or *c*) a card says that he wins.

The Golden Rule

When a Magic *card contradicts the rulebook, the card wins.* [14, p. 14]

This is the most important rule in all of Magic: The Gathering. It makes the game both more entertaining and difficult at the same time.

Deck Building

Each player should have his own deck which he builds from the cards in his collection. There are two rules: First, there have to be at least 60 cards, and second, there can't be more than four copies of a single card (except for basic lands). 60-Card decks are considered standard and there are hints for their creation:

- About 2/5 of a deck should be lands.
- There should be enough creatures with varied mana costs.
- The cards should complement each other.

Chapter 3

Environment

In this chapter, we will describe our implementation of the game and changes we've made to the rules in order to implement the game on a computer. We will also describe available evaluation functions and how they should be used.

3.1 Implemented Rules

There are two things that go against implementing the complete rules of MTG. They are partially tied to each other. The first is the ratio of cards that use those specific parts of the rules. Most of the specific rules are limited to certain sets and abilities. As we are using only a few cards (in comparison to the total number of existing cards) in our experiments, it would be unreasonable to implement things that wouldn't be even used. The second thing is *The Golden Rule*, which is quite troublesome. It basically states that we should be prepared to change anything in our structure. To be completely prepared would require going through all the cards. Moreover, these changes can be very complex.

For these reasons, we implemented only a subset of rules. We mostly stayed true to the basics described in section 2.2. We made a few modifications that wouldn't cause any damage to the core of the game but rather simplify it a little. Most of cut-offs are in abilities and the effects they produce. As a result, the rules we have implemented are a subset of the rules introduced in Basic Rulebook of Magic 2010 Core Set [14]. Anything beyond that can be considered missing unless stated otherwise. Multiplayer variants and tournament specific rules aren't implemented.

Core of the Game

The things mentioned here are related to the core of the game.

- Mana costs containing *snow* or *hybrid* mana aren't supported.
- Supertypes and subtypes have only informative value. [15, p. 29–32]
- Players gain priority: *a*) when their opponent passes, casts a spell, or activates an ability; *b*) when the top item on the stack resolves; *c*) at the beginning of the *First Main Phase*, the *Second Main Phase*, and the *End*

of combat step; *d*) after they declare attackers; and *e*) after their opponent declares blockers.
- Multiple creatures blocking one attacking creature cannot be reordered to the attacking player's will.
- Assigning damage during Combat damage step is done automatically by the game. Players cannot influence it.
- Player loses the game when his life drops to 0 or less or when he is forced to draw more cards than he has in his library. The game can end in a draw if both players meet the requirements for losing when one of them gains priority.

Abilities

The abilities cannot use any advanced things like conditions, choosing between effects to produce or linking of abilities.

Static abilities are updated three times per turn: *a*) during the *Untap step*; *b*) at the end of the *First Main Phase*; and *c*) at the end of the *Second Main Phase*. It should only matter for newly created tokens (creatures created by spells or abilities) that should be affected by that ability.

Triggered abilities can trigger at these events: *a*) the beginning and the end of all steps and phases; *b*) dealing damage to cards or players; *c*) moving cards between zones; *d*) executing actions; and *e*) paying mana cost.

Activated abilities cannot be limited to a number of activations per turn. Their cost can be paid with: *a*) mana; *b*) life; *c*) loyalty counters (planeswalkers only); and *d*) actions. Planeswalkers' abilities aren't limited in any way.

Supported keyword abilities, or characteristics taken as such, are as follows: (*a*) deathtouch, (*b*) defender, (*c*) double strike, (*d*) enchant, (*e*) equip, (*f*) fear, (*g*) first strike, (*h*) flash, (*i*) flying, (*j*) haste, (*k*) lifelink, (*l*) reach, (*m*) trample, (*n*) unblockable, and (*o*) vigilance. [15, p. 80, 84–88, 93]

Most abilities aren't keyword abilities. They are defined by effects they produce. The abilities make the cards or players do many things. In our implementation, they can:

- produce a certain amount of colored or colorless mana
- let a player gain a specified amount of life
- force a player to lose a specified amount of life
- force a player to draw or discard a specified amount of cards
- deal a specified amount of damage
- force a player to shuffle his library
- put cards on top, bottom or randomly into a player's library
- move cards from one zone to another
- permanently increase or decrease power and toughness of a creature
- put named counters on a card and remove them
- create specified amount of tokens
- tap or untap permanents
- sacrifice permanents

- destroy permanents
- exile cards
- select a subset of current targets
- counter spells
- increase or decrease power and toughness of a creature for some time
- change the player who controls a card for some time
- change the maximal number of cards on hand for some time
- disable or enable actions for some time
- specify what other creatures can a creature block or be blocked by for some time
- give a new ability to a card for some time

The amounts needed by some abilities can usually be specified in different ways: *a*) as an exact number; *b*) by linking it to the number of certain cards in the game; or *c*) by linking it to the current values of player's or opponent's life or to paid variable mana cost of a card or an ability.

The effects that are supposed to last for some time use the same events for breakpoints as triggered abilities. The breakpoint doesn't have to be specified. In that case the effect lasts as long as the ability that produced it is enabled.

3.2 Evaluation Metrics

Game Score

Game score is a function that evaluates current state of the game. Specifically, it scores both players and returns the difference of their scores ($first - second$). Positive values are in favor of the first player, negative values are in favor of the second player. It takes into account only the most important information even if it might overlook some details.

The score of a player is determined by the elements mentioned below. Each of them uses one or more constants which they are multiplied by or added to. These constatnts aren't based on any real evidence or further research. They are more or less guessed to reflect the importance of each element or its parts.

Life is the main indicator of player's well-being. It affects the score a lot. However, really high values of life aren't important. Therefore values greater than 20 are scored gradually less and less after each set amount.

For example, let's say that the life is multiplied by 4 to get its score. Then each 5 over 20 are multiplied by half the amount as their predecessor. The score would be 72 *for 18 life,* 86 *for 23 life,* 93 *for 28 life, and so on.*

$$\lambda < 1$$
$$\beta > 1$$
$$x = (life - 20) \bmod \beta$$
$$n = 1 + (life - 20 - x)/\beta$$
$$score_L = \begin{cases} life * \alpha_L & \text{if } life \leq 20 \\ 20\alpha_L + \sum_{i=1}^{n-1} \alpha_L \beta \lambda^i + x\beta\lambda^n & \text{if } life > 20 \end{cases}$$

Library takes its size as the base for score. It's almost not needed to include in score but there are still a few reasons to do so. It might be the cause of player's defeat, however seldom it is. It also represents all the cards that player hasn't yet drawn.

$$score_B = librarySize * \alpha_B$$

Hand is similar to library but it is more important. It takes its size as the base for score. It represents cards that can be used and accounted for in future moves. We don't score the cards by themselves because players don't know the contents of each other's hand. By scoring them, it could reveal otherwise unknown information.

$$score_H = handSize * \alpha_H$$

Permanents are another strong indicator of player's well-being besides life. They represent his available resources, support, and offensive and defensive forces. Permanents are defined by their abilities. Planeswalkers have loyalty in addition to that and creatures have power and toughness. We use all these elements and the number of attachments to compute the score. Power and toughness are scored gradually like life. We only differentiate between the types of abilities and the number of their targets. We don't consider how exactly each ability affects the game.

$$score_P = \sum \text{score of each permanent}$$

Defeated Status decreases the overall score greatly if a player is currently considered defeated. Otherwise, it does nothing.

$$score_D = \begin{cases} 1 & \text{if player is not defeated} \\ \delta & \text{if player is defeated; } \delta < 1 \end{cases}$$

$$GameScore = (score_L = score_B + score_H + score_P) * score_D$$

Card Evaluation

Although it is possible to use game score to evaluate cards separately, it is still lacking in some areas. Most importantly, game score assumes that the card is on the battlefield. Card evaluation is made to make up for it. It is designed to evaluate cards in any zone and take it into consideration.

As with score, this also uses constants to multiply or add each of the building elements. Again, they were mostly guessed and therefore might not reflect card's value properly. It just considers more things when making an estimate.

Power, *toughness*, *loyalty* and the *number of attachments* are multiplied by their respective constants. Abilities are evaluated in more detail. In addition to this, costs and produced effects are part of evaluation as well. The sum of all of these makes the value of a card.

Costs reduce the overall value. This applies not only to the mana cost of cards but to the cost of activated abilities as well. Costs on planeswalker's abilities are the only exception. They can either increase or decrease current loyalty counters of their planeswalker card. The costs that increase the loyalty counters don't reduce the value of card but instead increase it. All the costs include their respective amounts of things that are paid with. Those are: *a*) mana; *b*) cards; *c*) life; and *d*) loyalty.

Abilities are considered in more detail than in game score. Keyword abilities are split into groups and evaluated based on those. They are: *a*) always advantageous; *b*) advantageous depending on other cards; and *c*) advantageous only at certain situations. Produced effects, and if they can be used or not, are the main factors for other abilities. Triggered abilities don't consider anything else besides these two. The value of activated abilities is reduced by their cost. The value is reduced significantly if there are not enough targets for the card. Static abilities take into account the number of cards they are currently affecting.

Produced effects are evaluated based on their usefulness and the amount of their respective elements. The value of some effects is influenced by the player who is affected by them. Current situation of the game is not accounted for. The effects only consider how much they will affect the game.

For example, shuffling player's library doesn't change the situation much, so it is not evaluated much. On the other hand, dealing damage has significantly more visible effect. There can be a huge difference between dealing 1 damage and dealing 5 damage.

$eval_A$	...	value of abilities
$eval_{PT}$	...	value of power and toughness
$eval_L$	...	value of loyalty
$eval_S$	...	value of attachments
$eval_C$	...	value of mana cost

$$CardEvaluation = 1 + eval_A + eval_{PT} + eval_L + eval_S - eval_C$$

3.3 Simulation

Simulations are used to determine or predict possible future events and the state of the environment based on currently known information.

The simulation we have implemented into the environment doesn't actually simulate the game. It enables the game to revert into its previous state. It uses a system of stack-like buffers to store reversal actions. The *buffering of actions* is done for all important values as long as at least one buffer is active. For an example see Figure 3.1. Simulation itself is handled by buffers which keep track of possible actions.

procedure `Gain-Life(`*value*`)`
    put `Reverse-Life-Change(`*life*`)` on top of *action-buffer*
    life ← *life* +*value*

---

procedure `Reverse-Life-Change(`*value*`)`
    life ← *value*

Figure 3.1: Example of action buffering

Minimax

In most cases, players take turns making decisions. With simulation used, each player creates a new layer for each set of decisions. Game score is usually used to evaluate current situation. As the simulation goes on, the best option is selected at each layer. It resembles *minimax* in selecting minimum or maximum based on the current layer. When we realized it, we implemented *alpha-beta pruning* to optimize it.

The minimax theorem states [9]:

> *For every two-person, zero-sum game with finite strategies, there exists a value V and a mixed strategy for each player, such that (a) Given player 2's strategy, the best payoff possible for player 1 is V, and (b) Given player 1's strategy, the best payoff possible for player 2 is -V.*

This theorem was established by John von Neumann in 1928 as a contribution to economics. It was later improved and extended to work with games that involve imperfect information or more than two players. A great emphasis was put on it in *Theory of Games and Economic Behavior* written by John von Neumann and Oskar Morgenstern in 1944.[4, p. 142]

Minimax is used to determine the best move for a player. It expects both players to play optimally. Player can only end up better than initially thought if his opponent makes non-optimal decisions.

Figure 3.2 shows a description of a minimax algorithm.

Alpha-beta pruning is an optimization of minimax. It doesn't explore decisions that wouldn't change the result. It gives the same result as minimax alone.

```
function Minimax-Decision(state) returns an action
    v ← Max-Value(state)
    return the action in Successors(state) with value v

function Max-Value(state) returns a utility value
    if Terminal-Test(state) then return Utility(state)
    v ← −∞
    for a, s in Successors(state) do
        v ← Max(v, Min-Value(s))
    return v

function Min-Value(state) returns a utility value
    if Terminal-Test(state) then return Utility(state)
    v ← ∞
    for a, s in Successors(state) do
        v ← Min(v, Max-Value(s))
    return v
```

Figure 3.2: Minimax algorithm

The first ideas about it were by John McCarthy in 1956. It was then independently discovered by a number of people (Hart and Edwards, 1961; Brudno, 1963; Slagle, 1963). Donald Knuth and Ronald W. Moore refined it and proved its correctness in 1975.[4, p. 143]

Figure 3.3 shows a description of alpha-beta search algorithm.

Buffers

The main purpose of buffers is to store reversal actions and to handle simulation. Every buffer represents a layer on a path inside a tree structure. It acts as a breakpoint up to which the reversal is possible.

Each buffer is created with information about the player who created it and the options available at this layer. Options are actions that represent the decisions of a player. By default, the buffer stores all the options and then executes the first one in line. When the next option is required, the buffer evaluates the situation and keeps the better option. It then reverts back to the beginning of current layer and continues with the next option (see figure 3.4). When there are no other options to explore, the best found option is returned and the buffer is removed.

This process serves as the simulation when it is done with multiple buffers. It goes through the tree created by them in depth-first order. It works like a *minimax*.

Apart from the normal buffer mentioned above, there is also an *initial buffer*. The initial buffer is supposed to be at the bottom of the layered structure. It is created like a normal buffer with the addition of initial action and a number defining iterations. The initial action is performed after the creation of the buffer and at the beginning of each iteration. A special inner buffer is used to store the reversal actions created from the initial action. The buffering is done as usual after that. Exploring the next option is similar to the normal buffer with some

```
function Alpha-Beta-Search(state) returns an action
    v ← Max-Value(state, −∞, +∞)
    return the action in Successors(state) with value v
```

```
function Max-Value(state, α, β) returns a utility value
    if Terminal-Test(state) then return Utility(state)
    v ← −∞
    for a, s in Successors(state) do
        v ← Max(v, Min-Value(s, α, β))
        if v ≥ β then return v
        α ← Max(α, v)
    return v
```

```
function Min-Value(state, α, β) returns a utility value
    if Terminal-Test(state) then return Utility(state)
    v ← ∞
    for a, s in Successors(state) do
        v ← Min(v, Max-Value(s, α, β))
        if v ≤ α then return v
        β ← Min(β, v)
    return v
```

Figure 3.3: Alpha-beta algorithm

```
function Next-Option(score) returns an action
    if score is better than bestScore then
        Update-Best(score, current-option)
    remove first item from options
    Undo
    if options is empty then
        return null
    else
        return options[0]
```

```
procedure Undo
    while buffer is not empty do  remove and execute top action on buffer
```

Figure 3.4: Algorithm for trying next option in current buffer

differences (see Figure 3.5). The initial buffer keeps track of the current iteration. The special inner buffer is reverted at the end of each iteration. A new iteration starts when all options are explored and there are still some iterations left to do. The best option is selected using values from all iterations.

```
function Init-Next-Option(score) returns an action
    save score in results for current option and iteration
    Undo
    o ← o + 1                                   // zero-based index of current option
    if not all options explored then
        return options[o]
    else
        UndoInit                                 // same as Undo, but for init buffer
        if not all iterations explored then
            invoke InitAction
            o ← 0
            return options[0]
        else
            return null
```

Figure 3.5: Algorithm for trying next option in init buffer; `Undo` procedure is the same is in figure 3.4.

Observation

Observation is a feature of the normal buffer. It executes each available option, evaluates the situation, and keeps the better option. It reverts after the evaluation so that the next option can be explored. Index of the best option is returned. Observation serves as a secondary one-step simulation that doesn't affect the primary simulation. Figure 3.6 describes the algorithm.

```
function Observe(EvaluationFunction) returns index of best option
    foreach o in options do
        invoke o
        score ← invoke EvaluationFunction
        if score is better than bestScore then
            Update-Best(score, o)
        Undo
    return index of bestOption
```

Figure 3.6: Observation algorithm; `Undo` procedure is the same as in figure 3.4.

Chapter 4

Agents

In this chapter, we will at first describe all decisions the agents make during the game. We will then take a detailed look at each of our agents. We will explain how they work and what the core of their decision system is. Finally, we will state our hypothesis about their performance.

4.1 Decisions

There are five basic decisions an agent can make: a*) Mulligan; b) Play; c) Declare attackers; d) Declare blockers; and e) Select.* Each of them, except *Mulligan*, has own sub-decisions. This allows the agent to use appropriate methods where necessary.

Mulligan: The agent decides whether it should change cards in hand before the beginning of the game. This decision is not used during the game. The agent knows how many cards it would draw the next time and what cards it currently has.

Play: The agent needs to respond each time it receives priority. The agent knows the reason for receiving it (see Figure 4.1), which helps in selecting action to perform. The actions an agent can perform are: *a*) passing; *b*) casting spells (cards); *c*) activating abilities; and *d*) playing lands. Passing is always available and equals to doing nothing and letting the game move forward. The other actions require enough resources to pay their cost or certain part of turn. Each card, ability, or land is considered as separate action and only one of them can be chosen at a time.

Declare attackers: This decision is used by the active player at the beginning of the *Declare attackers step*. It is skipped if the active player has no creatures that can attack. The active player can select the target for each attacking creature. His opponent, and all planeswalkers that are controlled by his opponent, can be a target. There is no upper or lower limit to the number of creatures that can attack. The agent is only limited by the number of creatures it has on the battlefield.

- Next step during own turn.
- Opponent passed.
- Opponent cast spell.
- Opponent activated ability.
- An item on stack resolved.

Figure 4.1: Reasons for gaining priority

Declare blockers: This decision is used by the attacked player at the beginning of the *Declare blockers step*. It is skipped if the attacked player cannot block any of the attacking creatures. The attacked player can select which of his creatures will block each attacking creature. There is no upper or lower limit to the number of creatures that can block one attacking creature.

Select: There are three main subtypes: *a*) select one; *b*) select many; and *c*) select amount. *Select amount* is used for variable mana. The other two are much more versatile. The agent gets a list of items to select from, the context of the current selection, and possibly some extra data (an ability affecting the targets, mana cost to pay, ...). *Select many* specifies how many items should be selected. Context under which selection can be done contains:

- targets of an ability
- target for attackers
- declaring attackers
- creatures to block
- spell to cast
- ability to activate
- playing land
- paying mana cost
- discarding cards
- tapping permanents
- untapping permanents
- destroying permanents
- sacrificing permanents
- exiling cards

4.2 Random

This agent is based on random selection. The agent uses it for every decision it can. The agent uses other random-based methods for the rest. Random number generator uses uniform distribution for all the decisions.

Mulligan: The agent has a 33% chance to mulligan if it would draw more than 4 cards. The agent will keep its hand in other cases.

Play: The agent automatically passes when the top item on the stack is its own or if its opponent passed. In other cases, the agent randomly decides between casting spells (50% chance), activating abilities (30% chance) and passing (20% chance). The agent tries to play a land if the priority is received because of the beginning of the next step.

Declare attackers: Done by using *Select*.

Declare blockers Done by using *Select*.

Select: *Select amount* gets random number from the specified range. *Select one* gets an item at random from the list of items. *Select many* at first randomly chooses how many items from the specified range it should get. It then selects that many items at random from the list of items. Neither of these differentiates between contexts.

4.3 Simple

This agent was made with the intention to play according to a simple strategy. This is achieved through card evaluation (see Section 3.2) and other simple decisions.

Mulligan: This agent always keeps its hand.

Play: The agent tries to play a land as a first action if it received priority because of the beginning of its next step. Then, if the agent can cast any spell, it does so by using *Select*. The agent tries the same with abilities if it couldn't cast any spell. The agent passes if it couldn't cast or activate any spells or abilities. The agent goes through the same sequence, except for playing a land, if the priority was received because the top of the stack resolved. The agent passes in all other cases.

Declare attackers: The agent always attacks with all creatures it can and targets the opponent.

Declare blockers: The agent goes through its creatures that can block and assigns them to block the first attacking creature that: *a*) isn't already blocked; and *b*) whose power is lesser than the blocking creature's toughness.

```
function Declare-Blockers(attackers, blockable) returns list of blockers assignments
    creatures ← get creatures available to block
    if creatures not empty and blockable not empty then
        initialize result-list
        foreach c in creatures do
            if c can block at least 1 attacker then
                get a from attackers such that:
                  a is not blocked and a's power < c's toughness
                add (c, a) group to result-list
        return result-list
    else
        return null
```

Figure 4.2: Simple — Algorithm of Declare blockers decision

Select: *Select amount* always takes the maximum value. *Select one* gets random item, using uniform distribution, when selecting land that should be put on the battlefield. Otherwise, the available items are sorted based on their

value from card evaluation. An item with the least value is selected when sacrificing or discarding. An item with the highest value is selected in other cases. Casting spells is an exception — the best item is taken only if its value is positive. Otherwise, nothing is selected. *Select many* at first specifies the number of items it should get. That number is determined as a minimum of the amount of the available items and an average of the specified amount to select (see Equation 4.1). It then proceeds the same way as in *select one*, but it takes the specified amount of items, not just one.

$$toSelect = \min\left(\frac{lowerBound + upperBound}{2}, itemsCount\right) \tag{4.1}$$

4.4 Advanced

This agent is based on simulation (see Section 3.3). It uses game score (see Section 3.2) to evaluate the results of the simulation. Additionally, it uses an opponent model to properly simulate the progress of the game. We were inspired to use simulation by Patrick Buckland [2] and our experience with Minirisk project [3]. The agent uses a separate simulation for combat.

There are three parameters to define when creating this agent. Using the terms from Section 3.3, they are: *a*) minimal number of layers; *b*) maximal number of layers; and *c*) the number of iterations for the initial buffer. These parameters give us an opportunity to explore another aspect of this agent. Specifically, its performance at different lengths of simulations and the usage of its opponent model.

Decisions

Mulligan: The agent keeps his hand if it gets to four cards. Otherwise, it considers how many lands and creatures it has. The agent mulligans if less than $\frac{2}{5}$ of its hand are lands or if it has no creatures. Equation 4.2 shows the whole condition.

$$toDraw \geq 4 \textbf{ and } \left(lands < (toDraw + 1)\,\frac{2}{5} \textbf{ or } creatures == 0\right) \tag{4.2}$$

Play: The agent plays a land when it is possible. The simulation is then used to determine the next move. If the simulation is not active the agent starts a new one with the initial buffer. All options are explored. The agent creates a new layer with the first few options it has, if the simulation is active and still can continue. Next available option is tried, if the simulation cannot continue anymore. The simulation can continue, if the number of layers is lesser than is the maximal number of layers, and it is not interrupted or the number of layers is lesser then the minimal number of layers. The simulation is interrupted with the beginning of a new turn. Equation 4.3 shows the whole condition.

The options are gathered in the following order: pass, cards to cast, abilities to activate. Cards are sorted by their *game score* value. Abilities are sorted by their *card evaluation* value. There are no duplicates.

$$layers < maxParam \textbf{ and } (notInterrupted \textbf{ or } layers < minParam) \tag{4.3}$$

Declare attackers: The agent at first sorts the possible attacking by the combined value of *game score* of the creature and an average of creature's power and toughness. It is described in Equation 4.4. The agent then tries every combination of possible attackers. The combinations that only use a few creatures are skipped if the total number of possible attackers is high enough. Only some combinations are tried if there are too many possible attackers or if the simulation is active. Those combinations are: (*a*) no creatures; (*b*) all creatures; (*c*) currently unblockable creatures; (*d*) all groups from the start when in line; (*e*) all groups from the end when in line; and (*f*) all inner groups.

For example, the combinations for attackers A, B (unblockable), C, D, and E would be (without duplicates): -, ABCDE, B, A, AB, ABC, ABCD, BCDE, CDE, DE, E, BCD, and C.

Trying consists of using *combat simulation* with selected attackers and all of the creatures the opponent can block with. If the simulation is not active, it is assumed that the opponent will attack next turn with all of his available creatures. *Combat simulation* is used to evaluate this possibility. In the end, the combination with the highest value is selected.

Creatures always target player's opponent.

$$sortValue = score * \frac{power + toughness}{2} \tag{4.4}$$

Declare blockers: The agent uses *combat simulation* to determine how to block.

Select: *Select amount* takes one less than maximum. *Select many* uses random selection for everything except paying mana. In that case, it tries to match the wanted mana cost. It prefers using basic lands over other sources. *Select one* gets random item for everything except discarding and targets of abilities. Discarding selects the card with the least *card evaluation* value. Selecting the target for an ability is done by using observation (see section 3.3). All the possible targets are sorted by the product of their *game score* and *card evaluation* values. All of the random selections use uniform distribution.

```
function Declare-Attackers returns list of attackers
    creatures ← get creatures available to attack
    if creatures empty then
        return null
    else
        sort creatures
        initialize best-attackers
        blockers ← opponent's creatures available to block
        if lot of creatures or simulation active then
            best-attackers ← Inspect-Combat(no creatures, blockers, best-attackers)
            best-attackers ← Inspect-Combat(creatures, blockers, best-attackers)
            if simulation not active then
                foreach sublist of creatures do
                    best-attackers ← Inspect-Combat(sublist, blockers, best-attackers)
        else
            foreach combination of creatures do
                if many creatures and too few in combination then
                    continue
                else
                    best-attackers ← Inspect-Combat(combination, blockers,
                    best-attackers)
        return best-attackers
```

```
function Inspect-Combat(attackers, blockers, best-attackers) returns list of attackers
    sim-result ← Combat-Simulation(attackers, blockers)
    if simulation not active then
        next-att ← opponent's creatures available to attack next turn
        next-bl ← player's creatures available to block next turn
        next-result ← Combat-Simulation(next-att, next-bl)
        sim-result ← combined sim-result and next-result
    if sim-result is better than best-result then
        Update-Best(sim-result)
        return attackers
    else
        return best-attackers
```

Figure 4.3: Advanced — Algorithm of Declare attackers decision

Combat Simulation

Combat simulation is implemented separately from the game environment and has no relation to the main simulation. In that regard, it doesn't take into account any further changes that could alter its results. It works with known attackers and possible blockers to determine the best way to block.

Combat simulation keeps the input setting and results of the last simulation and uses those results if it assumes that the new setting is the same. The comparison isn't detailed enough as it only checks creatures' power and toughness, but not abilities.

The simulation itself uses recursion. It takes the first attacker and tries combinations of blockers. For each of these combinations, it simulates the remaining attackers and blockers.

Combinations are selected in a few steps. The remaining steps aren't explored if current best combination is considered good enough. A combination of blockers is good enough if only the attacker died or if the summed *game score* value of the dead blockers isn't much higher than the *game score* value of the dead attacker.

The first three steps consist of selecting groups of one, two, and three creatures respectively. The final step is done only if the main simulation is not active and the best result is still not good enough. All the remaining combinations are tried if there aren't too many blockers. Otherwise, only the groups from start of the line, from the end of the line and the inner groups are tried; much like during declaring attackers mentioned above.

Each group of one attacking creature and the creatures blocking it is processed and evaluated in a simulated fight. Some of the keyword abilities are accounted for during the process, specifically: *a*) deathtouch; *b*) lifelink; *c*) first strike; *d*) double strike; and *e*) trample. The value is based on the life of both players and *game score* values of killed creatures.

See figure 4.4 for algorithm.

Opponent Model

When the simulation is used, the agent has to predict the moves of his opponent as well. The opponent model is needed for this task. It substitutes the opponent while the simulation is active.

The opponent model doesn't use any complex techniques. It is based on *guessing the hand*. At the beginning of each iteration of the simulation, we give the opponent a new hand of the same size. The new hand contains randomly selected cards, using uniform ditribution, from his library and his current hand. The opponent model makes the same decisions as *Advanced* agent during active simulation.

```
function Combat-Simulation(attackers, blockers) returns a structure of casualties,
list of attacker-blockers groups and score
    if attackers empty then
        return default-result                    // no casualties, no a-b groups, zero score
    result ← default-result
    a ← attackers[0]
    remove first item from attackers
    a-blockers ← sorted blockers able to block a
    if a-blockers not empty then
        foreach single in a-blockers do
            result ← Try-Selected(attackers, blockers, a, single, result)
        if result not optimal then
            foreach pair in a-blockers do
                result ← Try-Selected(attackers, blockers, a, pair, result)
            if result not optimal then
                foreach trio in a-blockers do
                    result ← Try-Selected(attackers, blockers, a, trio, result)
                if simulation not active and result not optimal then
                    if lot of a-blockers then
                        foreach sublist of a-blockers do
                            result ← Try-Selected(attackers, blockers, a, sublist,
                            result)
                    else
                        foreach combination of a-blockers do
                            result ← Try-Selected(attackers, blockers, a,
                            combination, result)
    else
        best-result ← Combat-Simulation(attackers, blockers)
    result ← combine result with best-result
    return result
```

```
function Try-Selected(rem-attackers, all-blockers, attacker, blocking, result) returns
a structure of casualties, list of attacker-blockers groups and score
    temp-result ← Combat-Simulation(rem-attackers, all-blockers except blocking)
    fight-result ← Fight(attacker, blocking)   // returns structure of casualties and score
    if fight-result combined with temp-result is better than best result then
        result ← temp-result
        UpdateBest
    return result
```

Figure 4.4: Combat simulation algorithm

4.5 Hypothesis

We expect *Random* to be worse than others. Although its chances to cast spells and activate abilities are higher than passing, it still doesn't play because of current situation. The more creatures it has the higher probability there is that it will attack with some of them. We can see *Random* winning some duels, but not many.

Simple should play better than *Random*. It tries to cast or activate the "best" available cards and abilities and attacks regularly. However, its selection depends on evaluation that might be faulty. Another of the weak points *Simple* has is blocking. It blocks so that its creatures survive but doesn't care about its own life. Attacking with everything without further investigation might lead to needless deaths of weaker creatures. The process of selection is also not good because it takes the "best" item, which is not always the best choice. *Simple* should play good if it has a lot of creatures and mostly static or triggered abilities. Otherwise, it can't use its skills.

Advanced should be a lot better than both *Random* and *Simple*. However, we aren't that sure about its performance against another *Advanced* agents with different parameters; or rather the influence of parameters on the performance of agents. It should be able to plan ahead and make simple strategies. Though, it is possible that the agent will be making decisions that aren't actually important just because they lead to better score. Its handling of combat can make the agent more cautious than it is needed. But all in all, we expect it to play like an average casual player.

We would like to verify these things with the results from experiments:

1. Random should hardly win 1 out of 5 duels against other agents.
2. Random should be visibly worse than other agents.
3. Simple should win 2 out of 5 duels at most against other agents.
4. Simple should be better than Random but it should not be close to Advanced.
5. There should be large differences between Advanced agents with distinct parameters.
6. Higher maximum limits for Advanced agents should worsen their performance.
7. More iterations for Advanced agents should worsen their performance.

Chapter 5

Experiments

In this chapter, we present the settings used for experiments and the results we got from them. Each part of the settings will be explained. Results are organized in tables. They show the amount of won duels for different settings.

5.1 Settings

One game consists of five duels and keeps the same setting during all of the duels. The settings of each game are determined by three things: *a*) the agents that control the players; *b*) the decks the players can play with; and *c*) the seed for the random number generator. Placing different agents against each other is our plan from the beginning. Using various decks helps in finding out how they play with different cards. Having the seed for the random number generator preset or leaving it unset creates new situation. With these things changing we can observe the behavior of the agents under different circumstances.

There are seven agents in total; all listed in Figure 5.1. Five of them are *Advanced*, each with different parameters. They are set to similar values so that we can better understand how they influence their performance. The first and second parameters represent the minimum and maximum simulation depth respectively. The third one represents how many times it tries to guess the opponent's hand.

- Random
- Simple
- Advanced 2–16–1
- Advanced 4–12–1
- Advanced 4–12–5
- Advanced 6–6–5
- Advanced 6–6–10

Figure 5.1: Experiments settings — Agents

We have three decks at our disposal. They are listed in Figure 5.2, together with the arrangement for duels. Their differences will be explained later.

- Elves
- Goblins
- Liliana Vess

- Elves *vs.* Goblins
- Goblins *vs.* Elves
- Goblins *vs.* Liliana Vess
- Liliana Vess *vs.* Goblins
- Liliana Vess *vs.* Elves
- Elves *vs.* Liliana Vess

Figure 5.2: Experiments settings — Decks

Using different seeds give us an opportunity to start the games from the same starting point. This is then good when making comparisons. On the other hand, using unset seeds allow us to explore more cases. We have ten cases in total, six with set seeds and four with unset seeds. They are listed in Figure 5.3.

- 0
- 5
- 20
- 60
- 2010
- 13,860
- unset A
- unset B
- unset C
- unset D

Figure 5.3: Experiments settings — Seeds

5.2 Decks

All three decks we are using are standard 60-card decks. They are based on existing decks — Elves (ED) and Goblins (GD) from *Duel Decks: Elves vs. Goblins [10]* and Liliana Vess (LD) from *Duel Decks: Garruk vs. Liliana [11]*. Not all the cards in them are fully functional, as they were modified where it was necessary. Decklists of our ED, GD, and LD can be found in Appendix A. Each deck has different play style.

ED uses mainly creatures and a lot of tokens. A good portion of cards takes advantage of that. Be it healing, increasing power/toughness or creating even more tokens. With the right cards in play, there can be many strong creatures very quickly and player's life shouldn't be a problem.

As GD is designed to match ED, it also uses mainly creatures and tokens. It's good to have a good amount of tokens to use as they tend to be sacrificed by many abilities. There are a few ways to increase power/toughness. The main strategy for this deck is direct damaging of the opponent and his creatures.

LD doesn't rely on creatures as much as the previous two. Making an opponent discard cards or lose life while gaining it in return is very common with this deck.

It can damage or destroy creatures and in some cases use them as their own. It also has a big offensive advantage against the other two — creatures with flying. Neither ED nor GD can block those creatures.

5.3 Results

The results are organized in tables containing all variations of games between agents. The results presented in this section are either summed deck-specific or totals of a bigger group. Table 5.1 shows the number of duels for different settings.

In one game	5
Total	12600
Per agent	3600
Between two agents	600
Per deck setting	2100
Per deck-seed setting	210
Duels ended in a draw	4
Duels won through library	12

Table 5.1: Numbers of duels

In this section, we will use the following abbreviations to denote the agents:

- R — Random
- S — Simple
- A1 — Advanced 4–12–5
- A2 — Advanced 6–6–5
- A3 — Advanced 4–12–1
- A4 — Advanced 6–6–10
- A5 — Advanced 2–16–1

The cells in each table have the following meaning:

X-Y cell shows results of the game between agents X and Y where X played first. Agent X plays with the first deck and agent Y with the second deck. Numbers in format *x:y* show the amount of duels each of them won. When $(x + y) \bmod 5 > 0$, then some duels ended in a draw.

X-X cell shows how many times agent X won playing first and second against all other agents (shown as $\begin{matrix}\textbf{first}\\\textbf{second}\end{matrix}$).

Cells in last column show summed results of each agent playing first against others (shown as $\begin{matrix}agent\\others\end{matrix}$).

Cells in last row show summed results of each agent playing second against others (shown as $\overset{others}{agent}$).

Bottom right corner cell shows total results across all duels (shown as $\mathbf{\overset{first}{second}}$).

Table 5.2 shows results across all seeds using Elves vs. Goblins setting. We will use this table to describe how to read in all of the tables. The subsequent tables will not contain such a thorough description.

	R	**S**	**A1**	**A2**	**A3**	**A4**	**A5**	
R	**86** **21**	19:31	13:37	15:35	13:37	11:39	15:35	86 214
S	44:6	**174** **67**	24:26	23:27	26:24	28:22	29:21	174 126
A1	43:7	44:6	**233** **123**	35:15	40:10	33:17	38:12	233 67
A2	49:1	43:7	33:17	**239** **123**	40:10	40:10	34:16	239 61
A3	47:3	43:7	38:12	41:9	**239** **109**	36:14	34:16	239 61
A4	48:2	45:5	37:13	33:17	35:15	**233** **120**	35:15	233 67
A5	48:2	39:11	32:18	30:20	37:13	32:18	**218** **115**	218 82
	279 21	233 67	177 123	177 123	191 109	180 120	185 115	**1422** **678**

Table 5.2: Summed results of Elves vs. Goblins across all settings of the seed; For each game, the agent on the row plays first with Elves and the agent on the column plays second with Goblins. The cells where the same agent is on both row and column show the number of duels the agent won playing first and second against others. The cells in the last column show the number of duels the agent on each row won playing first against others. The cells in the last row show the number of duels the agent on each column won playing second against others. The bottom right cell shows the number of duels won with Elves and Goblins.
The cells in $X : Y$ format should be read as $first : second$ or $elves : goblins$. The cells in $\overset{X}{Y}$ format should be read as $\overset{first}{second}$ or $\overset{elves}{goblins}$.
To be concrete: The first player has Elves, the second player has Goblins. R won 86 duels playing first and 21 duels playing second. S won 24 duels playing first against A1, but only 6 duels when playing second against the same agent. A3 won 109 duels in total against all other agents when playing second. A5 won 218 duels in total against all other agents when playing first. Overall, agents playing first won 1422 duels and agents playing second won 678 duels. There were no duels that ended in a draw.

	R	S	A1	A2	A3	A4	A5	
R	**37** **37**	13:37	6:44	4:46	10:40	1:49	3:47	37 263
S	42:8	**138** **96**	20:30	22:28	16:34	16:34	22:28	138 162
A1	45:5	40:10	**173** **188**	30:20	23:27	19:31	16:34	173 127
A2	44:6	35:15	23:27	**168** **188**	23:27	22:28	21:29	168 132
A3	44:6	35:15	16:34	19:31	**167** **186**	26:24	27:23	167 133
A4	43:7	40:10	22:28	15:35	23:27	**163** **195**	20:30	163 137
A5	45:5	41:9	25:25	22:28	19:31	21:29	**173** **191**	173 127
	263 37	204 96	112 188	112 188	114 186	105 195	109 191	**1019** **1081**

Table 5.3: Summed results of Elves vs. Liliana Vess across all settings of the seed; For each game, the agent on the row plays first with Elves and the agent on the column plays second with Liliana Vess.

	R	S	A1	A2	A3	A4	A5	
R	**18** **89**	7:43	5:45	1:49	1:49	1:49	3:47	18 282
S	27:23	**68** **172**	8:42	6:44	8:42	7:43	12:38	68 232
A1	42:8	24:26	**113** **237**	9:41	13:37	15:35	10:40	113 187
A2	36:14	26:24	12:38	**114** **244**	17:33	7:43	16:34	114 186
A3	37:13	26:24	13:37	12:38	**117** **242**	18:32	11:39	117 183
A4	37:13	24:26	13:37	14:36	6:44	**106** **239**	12:38	106 194
A5	32:18	21:29	12:38	14:36	13:37	13:37	**105** **236**	105 195
	211 89	128 172	63 237	56 244	58 242	61 239	64 236	**641** **1459**

Table 5.4: Summed results of Goblins vs. Elves across all settings of the seed; For each game, the agent on the row plays first with Goblins and the agent on the column plays second with Elves.

	R	S	A1	A2	A3	A4	A5	
R	**40** **69**	16:34	4:46	6:44	2:48	8:42	4:46	40 260
S	33:17	**136** **108**	20:30	23:27	21:29	24:26	15:35	136 164
A1	35:15	38:12	**144** **190**	19:31	16:34	23:27	13:37	144 156
A2	42:8	35:14	25:25	**161** **195**	19:31	21:29	19:31	161 138
A3	40:10	31:19	20:30	20:30	**151** **200**	22:28	18:32	151 149
A4	41:9	33:17	20:30	17:33	27:23	**160** **183**	22:28	160 140
A5	40:10	38:12	21:29	20:30	15:35	19:31	**153** **209**	153 147
	231 69	191 108	110 190	105 195	100 200	117 183	91 209	**945** **1154**

Table 5.5: Summed results of Goblins vs. Liliana Vess across all settings of the seed; For each game, the agent on the row plays first with Goblins and the agent on the column plays second with Liliana Vess.

	R	S	A1	A2	A3	A4	A5	
R	**47** **35**	10:40	8:42	8:42	6:44	10:40	5:45	47 253
S	33:17	**103** **127**	12:38	16:34	19:31	11:39	12:38	103 197
A1	47:3	30:20	**200** **169**	30:20	31:19	34:16	28:22	200 100
A2	47:3	33:17	31:19	**209** **151**	36:14	36:14	26:24	209 91
A3	44:6	34:16	26:23	33:17	**190** **145**	31:19	22:28	190 109
A4	50:0	32:18	26:24	27:23	35:15	**202** **140**	32:18	202 98
A5	44:6	34:16	27:23	35:15	28:22	38:12	**206** **175**	206 94
	265 35	173 127	130 169	149 151	155 145	160 140	125 175	**1157** **942**

Table 5.6: Summed results of Liliana Vess vs. Elves across all settings of the seed; For each game, the agent on the row plays first with Liliana Vess and the agent on the column plays second with Elves.

	R	S	A1	A2	A3	A4	A5	
R	**79** **32**	18:32	10:40	9:41	13:37	16:34	13:37	79 221
S	33:17	**126** **105**	20:30	18:31	21:29	19:31	15:35	126 173
A1	45:5	40:10	**208** **152**	31:19	30:20	32:18	30:20	208 92
A2	47:3	34:16	32:18	**199** **165**	33:16	24:26	29:21	199 100
A3	48:2	37:13	26:24	26:24	**199** **144**	33:17	29:21	199 101
A4	46:4	32:18	27:23	25:25	28:22	**184** **146**	26:24	184 116
A5	49:1	34:16	33:17	25:25	30:20	30:20	**201** **158**	201 99
	268 32	195 105	148 152	134 165	155 144	154 146	142 158	**1196** **902**

Table 5.7: Summed results of Liliana Vess vs. Goblins across all settings of the seed; For each game, the agent on the row plays first with Liliana Vess and the agent on the column plays second with Goblins.

	R	S	A1	A2	A3	A4	A5	
R	**199** **172**	58:122	25:155	29:151	28:152	33:147	26:154	199 881
S	128:52	**463** **409**	65:115	68:112	73:107	68:112	61:119	463 617
A1	158:22	129:51	**659** **617**	99:81	95:85	87:93	91:89	659 421
A2	158:22	117:62	95:85	**646** **623**	105:75	89:91	82:98	646 433
A3	152:28	126:54	88:91	90:90	**651** **597**	103:77	92:88	651 428
A4	158:22	116:64	94:86	81:99	99:81	**645** **611**	97:83	645 435
A5	154:26	124:56	95:85	90:90	83:97	89:91	**635** **631**	635 445
	908 172	670 409	462 617	457 623	483 597	469 611	449 631	**3898** **3660**

Table 5.8: Summed results of games with set seed (0, 5, 20, 60, 2010, 13860) across all settings of the decks; For each game, the agent on the row plays first and the agent on the column plays second.

	R	S	A1	A2	A3	A4	A5	
R	**108** **111**	25:95	21:99	14:106	17:103	14:106	17:103	108 612
S	84:36	**282** **266**	39:81	40:79	38:82	37:83	44:76	282 437
A1	99:21	87:33	**412** **442**	55:65	58:62	69:51	44:76	412 308
A2	107:13	89:31	61:59	**444** **443**	63:56	61:59	63:57	444 275
A3	108:12	80:40	51:69	61:59	**412** **429**	63:57	49:71	412 308
A4	107:13	90:30	51:69	50:70	55:65	**403** **412**	50:70	403 317
A5	104:16	83:37	55:65	56:64	59:61	64:56	**421** **453**	421 299
	609 111	454 266	278 442	276 443	290 429	308 412	267 453	**2482** **2556**

Table 5.9: Summed results of games with unset seed (A, B, C, D) across all settings of the decks; For each game, the agent on the row plays first and the agent on the column plays second.

	R	S	A1	A2	A3	A4	A5	
R	**307** **283**	83:217	46:254	43:257	45:255	47:253	43:257	307 1493
S	212:88	**745** **675**	104:196	108:191	111:189	105:195	105:195	745 1054
A1	257:43	216:84	**1071** **1059**	154:146	153:147	156:144	135:165	1071 729
A2	265:35	206:93	156:144	**1090** **1066**	168:131	150:150	145:155	1090 708
A3	260:40	206:94	139:160	151:149	**1063** **1026**	166:134	141:159	1063 736
A4	265:35	206:94	145:155	131:169	154:146	**1048** **1023**	147:153	1048 752
A5	258:42	207:93	150:150	146:154	142:158	153:147	**1056** **1084**	1056 744
	1517 283	1124 675	740 1059	733 1066	773 1026	777 1023	716 1084	**6380** **6216**

Table 5.10: Summed results of all games across all settings used; For each game, the agent on the row plays first and the agent on the column plays second.

Chapter 6

Discussion

In this chapter, we compare the performance of our agents based on results of experiments. We then discuss possible improvements and future work.

6.1 Performance

Let's see if our hypothesis from Section 4.5 turned out to be true or if we were wrong:

1. *Random should hardly win 1 out of 5 duels against other agents.*
   Random won 590 duels out of 3600 which is about 16.39%. It didn't get to the 1 out of 5 but it wasn't that far.

2. *Random should be visibly worse than other agents.*
   We can say that this is true. Looking at the numbers, it is clear that he is the worst one.

3. *Simple should win 2 out of 5 duels at most against other agents.*
   Simple won 1420 duels out of 3600. That is about 39.4%. We were close and Simple was almost better than we expected.

4. *Simple should be better than Random but it should not be close to Advanced.*
   This is most likely true. Simple has more than twice as many victories as Random which definitely makes Simple better. It doesn't play as badly as Random against Advanced agents but the difference is still visible. Simple is not close to Advanced agents.

5. *There should be large differences between Advanced agents with distinct parameters.*
   We were wrong with this one. Looking at the number of their victories in total, there are only a little differences. The worst one has 2089 victories and the best one has 2156 victories. The difference is only in 67 duels, only 1.86%. That's almost no difference.

6. *Higher maximum limits for Advanced agents should worsen their performance.*

By comparing only the ones with the same number of iterations, we see no real difference in total results. It seems that simulation length isn't as important as we thought.

7. *More iterations for Advanced agents should worsen their performance.*
   By comparing only the ones with the same simulation length limits, we got opposing results. We compared 1 versus 5 iterations and 5 versus 10 iterations. The agents with 5 iterations were a little better in both cases. However, it wasn't by much.

Random turned out a lot like we expected him to. *Simple* did quite good as well, although we expected that it to be a little worse. We think that we helped them achieve better results by specifying the targets of abilities. We made it so that they give the most suitable choices, like damaging only the opponent or his creatures and not the creatures of player. While it is the logical choice to do, the real cards aren't specified like that. They usually don't specify whose creature to damage. Both agents would probably end up worse if we left it like that for experiments.

Advanced is more interesting. Simulations proved to be the right method to use. At first, it is surprising that the parameters don't influence the performance much. On second thought, it is understandable. The only two things they do differently is the number of guessing the hand of the opponent and the length of simulations. Given that the simulations last only to the end of turn doesn't add to the importance of their length either.

Inspecting the logs should reveal us more about the actual behavior of *Advanced* agents during the game. The differences between them are more visible there. The Combat itself is quite good. The agents block quite reasonably and they even use spells or abilities that would help them if they have them. Attacking is a little worse but still reasonable for the most part. The agents sometimes attack in a way that kills both the attacking and blocking creatures. Unfortunately, the blocking creature doesn't always have lower value. What the agents have in common is that they occasionally use certain cards or abilities ahead of time or completely unreasonably. Those are mostly cards and abilities that boost the score of the player for some time. Understandably, this occurs more for the agents with shorter simulations or less hand guessing because they don't realize that the boost of the score is only temporary.

Another thing we can see from results is how effective the decks are against each other. The *Goblins* deck (GD) is the weakest. It is really bad against the *Elves* deck (ED). It does better against the *Liliana Vess* deck (LD) but it still loses overall. The original Elves and Goblins decks were designed to be comparable with each other. However, we modified a number of cards for our implementation. Significantly more cards were modified in GD than in ED which is probably the reason why GD is so weak. ED and LD are comparable, with the latter being slightly better than the former. The superiority of LD is probably caused by the flying creatures it contains because neither ED nor GD has any defense against them.

6.2 Future Work

There are a lot of things that could be improved. Both game score and card evaluation are among the ones widely used. The way both of them are computed could be refined to give more precise values. Especially card evaluation should consider current situation and the effects produced by abilities much more.

The opponent model used in the thesis is very primitive and possibly weak. It would certainly benefit from learning a strategy of a concrete agent. Barto and Sutton [1] present reinforcement learning techniques that might be applicable for this task. If nothing else, they could lead to the elimination of unnecessary actions that are always considered.

The combat simulation is quite good. It gives reasonable results in most cases. However, it has a major flaw of being virtually separated from the game. The combat simulation doesn't consider anything happening after the blockers are declared, be it the actions of players or the triggered abilities. Even if these things don't always cause serious problems, they should still be addressed.

The environment should be refined as well. Although it works as it should, there are a few special cases where it could be improved. The cards the agents play with should have been taken into consideration much earlier. That way the cards could be more functional and used to their full potential.

Chapter 7

Conclusions

We successfully implemented the rules of Magic: The Gathering and developed an environment in which they are used. We used this environment to develop three agents with different levels of skill. We have found out that even though luck can ensure victory, it is not something a player should depend on. Refining one's skill is a better way. This is showed by our agents. Random strategy shouldn't be used for major decisions as it can be easily defeated by skilled players. A simple strategy based on the evaluation of cards can be effective. It can achieve at least average results if the evaluation function is good enough. Simulation turned out to be the technique to use. It's able to consider many possibilities that might happen. It can then alter its decisions and play accordingly. Magic: The Gathering is a complex game and our results are in no way definite. However, we believe that it is a viable test-bed environment when implementing artificial intelligence for modern card games.

Bibliography

[1] Barto A., Sutton R. S.: *Reinforcement Learning: An Introduction*, MIT Press, 1988.

[2] Buckland P.: *Duels of the Planeswalkers: All About AI*, Daily MTG, `http://www.wizards.com/Magic/Magazine/Article.aspx?x=mtg/daily/feature/44`, June 22, 2009 (accessed August 4, 2010).

[3] Iša J.: *Minirisk*, `http://artax.karlin.mff.cuni.cz/~isa_j1am/projects/minirisk/`, April 4, 2007 (accessed August 4, 2010).

[4] Russel S., Norvig P.: *Artificial intelligence: A Modern Approach*, Prentice Hall, 1995.

[5] Risk N. A.: *Using Counterfactual Regret Minimization to Create a Competitive Multiplayer Poker Agent*, M. Sc. Thesis, `http://webdocs.cs.ualberta.ca/~games/poker/publications/abourisk.msc.pdf`, 2009 (accessed August 4, 2010).

[6] Schnizlein D. P.: *State Translation in No-Limit Poker*, M. Sc. Thesis, `http://webdocs.cs.ualberta.ca/~games/poker/publications/schnizlein.msc.pdf`, 2009 (accessed August 4, 2010).

[7] : Waugh K., Bard N., and Bowling M.: *Strategy Grafting in Extensive Games*, Advances in Neural Information Processing Systems 22 (NIPS-09), `http://webdocs.cs.ualberta.ca/~games/poker/publications/NIPS09-graft.pdf`, 2009 (accessed August 4, 2010).

[8] Wikipedia contributors: *Magic: The Gathering*, Wikipedia, The Free Encyclopedia, `http://en.wikipedia.org/w/index.php?title=Magic:_The_Gathering&oldid=376337570`, July 30, 2010 (accessed August 4, 2010).

[9] Wikipedia contributors: *Minimax*, Wikipedia, The Free Encyclopedia, `http://en.wikipedia.org/w/index.php?title=Minimax&oldid=375069801`, July 23, 2010 (accessed August 4, 2010).

[10] Wizards of the Coast LLC: *Duel Decks: Elves vs. Goblins*, `http://www.wizards.com/magic/tcg/productarticle.aspx?x=mtg_tcg_elvesvsgoblins_productinfo`, 2007 (accessed June 15, 2010).

[11] Wizards of the Coast LLC: *Duel Decks: Garruk vs. Liliana*, http://www.wizards.com/Magic/TCG/ProductArticle.aspx?x=mtg/tcg/garrukvsliliana/productinfo, 2009 (accessed July 20, 2010).

[12] Wizards of the Coast LLC: *Gatherer, The Magic Card Database*, http://gatherer.wizards.com/Pages/Default.aspx, 2009.

[13] Wizards of the Coast LLC: *Getting Started, Card Anatomy*, http://www.wizards.com/Magic/TCG/NewtoMagic.aspx?x=mtg/tcg/newtomagic/gettingstarted (accessed July 27, 2010).

[14] Wizards of the Coast LLC: *Magic: The Gathering Basic Rulebook*, http://www.wizards.com/magic/rules/EN_Magic_Basic_Rulebook_20090710.pdf, 2009 (accessed July 12, 2009).

[15] Wizards of the Coast LLC: *Magic: The Gathering Comprehensive Rules*, http://www.wizards.com/magic/comprules/MagicCompRules_20090708.pdf, 2009 (accessed July 12, 2009).

[16] Wizards of the Coast LLC: *The Multiverse: Magic: The Gathering*, http://www.magicthegathering.com.

Appendix A

Decklists

Decks represented by these decklists were used in our experiments. They are based on the existing decks of the same name. Certain cards are different from the original ones.

A.1 Elves

This deck uses mainly creatures and a lot of tokens. A good portion of cards takes advantage of that. Be it healing, increasing power/toughness or creating even more tokens. With the right cards in play, there can be many strong creatures very quickly and player's life shouldn't be a problem.

#	**Card Name** **Type Line** ***Abilities***	**Mana Cost** **[Power/Toughness]**
1×	**Ambush Commander** Creature — Elf *{1}{G}, Sacrifice an Elf: Target creature you control gets +3/+3 until end of turn.*	{3}{G}{G} [2/2]
1×	**Allosaurus Rider** Creature — Elf Warrior *You may exile two green cards from your hand rather than pay Allosaurus Rider's mana cost.* *Allosaurus Rider's power and toughness are each equal to 1 plus the number of lands you control.*	{5}{G}{G} [1+*/1+*]
2×	**Elvish Eulogist** Creature — Elf Shaman *Sacrifice Elvish Eulogist: You gain 1 life for each Elf card in your graveyard.*	{G} [1/1]
1×	**Elvish Harbinger** Creature — Elf Druid *When Elvish Harbinger enters the battlefield, search your library for an Elf card, then shuffle your library and put that card on top of it.* *{T}: Add {G} to your mana pool.*	{2}{G} [1/2]
3×	**Elvish Warrior** Creature — Elf Warrior	{G}{G} [2/3]
2×	**Gempalm Strider** Creature — Elf *{2}{G}{G}, Discard Gempalm Strider: Draw a card. Elf creatures get +2/+2 until end of turn.*	{1}{G} [2/2]
1×	**Heedless One** Creature — Elf Avatar *Trample* *Heedless One's power and toughness are each equal to the number of Elves on the battlefield.*	{3}{G} [*/*]
2×	**Imperious Perfect** Creature — Elf Warrior *Other Elf creatures you control get +1/+1.* *{G}, {T}: Put a 1/1 green Elf Warrior creature token onto the battlefield.*	{2}{G} [2/2]

3×	**Llanowar Elves** Creature — Elf Druid *{T}: Add {G} to your mana pool.*	{G} [1/1]
2×	**Lys Alana Huntmaster** Creature — Elf Warrior *Whenever you cast an Elf spell, put a 1/1 green Elf Warrior creature token onto the battlefield.*	{2}{G}{G} [3/3]
1×	**Stonewood Invoker** Creature — Elf Mutant *{7}{G}: Stonewood Invoker gets +5/+5 until end of turn.*	{1}{G} [2/2]
1×	**Sylvan Messenger** Creature — Elf *Trample*	{2}{G} [2/2]
1×	**Timberwatch Elf** Creature — Elf *{T}: Target creature you control gets +X/+X until end of turn, where X is the number of Elves on the battlefield.*	{2}{G} [1/2]
1×	**Voice of the Woods** Creature — Elf *Tap five untapped Elves you control: Put a 7/7 green Elemental creature token with trample onto the battlefield.*	{3}{G}{G} [2/2]
2×	**Wellwisher** Creature — Elf *{T}: You gain 1 life for each Elf on the battlefield.*	{1}{G} [1/1]
1×	**Wirewood Herald** Creature — Elf *When Wirewood Herald is put into a graveyard from the battlefield, search your library for an Elf card, put it into your hand, then shuffle your library.*	{1}{G} [1/1]
1×	**Wirewood Symbiote** Creature — Insect *{T}, Untap target creature: Return an Elf you control to its owner's hand.*	{G} [1/1]
2×	**Wood Elves** Creature — Elf Scout *When Wood Elves enters the battlefield, search your library for a Forest card and put that card onto the battlefield. Then shuffle your library.*	{2}{G} [1/1]
1×	**Wren's Run Vanquisher** Creature — Elf Warrior *Deathtouch*	{3}{G} [3/3]
1×	**Elvish Promenade** Tribal Sorcery — Elf *Put a 1/1 green Elf Warrior creature token onto the battlefield for each Elf you control.*	{3}{G}
2×	**Giant Growth** Instant *Target creature you control gets +3/+3 until end of turn.*	{G}
1×	**Harmonize** Sorcery *Draw three cards.*	{2}{G}{G}
1×	**Wildsize** Instant *Target creature you control gets +2/+2 and gains trample until end of turn. Draw a card.*	{2}{G}
3×	**Moonglove Extract** Artifact *Sacrifice Moonglove Extract: Moonglove Extract deals 2 damage to your opponent or target opponent's creature.*	{3}
1×	**Slate of Ancestry** Artifact *{4}, {T}, Discard your hand: Draw a card for each creature you control.*	{4}
1×	**Wirewood Lodge** Land *{T}: Add {1} to your mana pool.* *{G}, {T}: Untap target Elf you control.*	
2×	**Tranquil Thicket** Land *When Tranquil Thicket enters the battlefield, tap it.* *{T}: Add {G} to your mana pool.* *{G}, Discard Tranquil Thicket: Draw a card.*	
19×	**Forest** Basic Land — Forest *{G}*	

A.2 Goblins

This deck uses mainly creatures and tokens. It's good to have a good amount of tokens to use as they tend to be sacrificed by many abilities. There are a few ways to increase power/toughness. The main strategy for this deck is direct damaging of the opponent and his creatures.

#	Card Name Type Line *Abilities*	Mana Cost [Power/Toughness]
1×	**Siege-Gang Commander** Creature — Goblin *When Siege-Gang Commander enters the battlefield, put three 1/1 red Goblin creature tokens onto the battlefield.* *{1}{R}, Sacrifice a Goblin: Siege-Gang Commander deals 2 damage to target opponent's creature or your opponent.*	{3}{R}{R} [2/2]
1×	**Akki Coalflinger** Creature — Goblin *First strike* *{R}, {T}: Attacking creatures gain first strike until end of turn.*	{1}{R}{R} [2/2]
1×	**Clickslither** Creature — Insect *Haste* *Sacrifice a Goblin: Clickslither gets +2/+2 and gains trample until end of turn.*	{1}{R}{R}{R} [3/3]
3×	**Emberwilde Augur** Creature — Goblin Shaman *Sacrifice Emberwilde Augur: Emberwilde Augur deals 3 damage to your opponent. Play this ability only as a sorcery.*	{1}{R} [2/1]
1×	**Flamewave Invoker** Creature — Goblin Mutant *{7}{R}: Flamewave Invoker deals 5 damage to your opponent.*	{2}{R} [2/2]
1×	**Gempalm Incinerator** Creature — Goblin *{1}{R}, Discard Gempalm Incinerator: Gempalm Incinerator deals X damage to target opponent's creature, where X is the number of Goblins on the battlefield. Draw a card.*	{2}{R} [2/1]
3×	**Goblin Cohort** Creature — Goblin Warrior	{1}{R} [2/2]
1×	**Goblin Matron** Creature — Goblin *When Goblin Matron enters the battlefield, search your library for a Goblin card and put it into your hand. Shuffle your library.*	{2}{R} [1/1]
1×	**Goblin Ringleader** Creature — Goblin *Haste*	{2}{R} [2/2]
1×	**Goblin Sledder** Creature — Goblin *Sacrifice a Goblin: Target creature you control gets +1/+1 until end of turn.*	{R} [1/1]
1×	**Goblin Warchief** Creature — Goblin *Other Goblin creatures you control have haste.*	{1}{R}{R} [2/2]
1×	**Ib Halfheart, Goblin Tactician** Legendary Creature — Goblin Advisor *Sacrifice two Mountains: Put two 1/1 red Goblin creature tokens onto the battlefield.*	{3}{R} [3/2]
1×	**Mogg Fanatic** Creature — Goblin *Sacrifice Mogg Fanatic: Mogg Fanatic deals 1 damage to target opponent's creature or your opponent.*	{R} [1/1]
2×	**Mogg War Marshal** Creature — Goblin Warrior *When Mogg War Marshal enters the battlefield, put a 1/1 red Goblin creature token onto the battlefield.* *When Mogg War Marshal is put into a graveyard from the battlefield, put a 1/1 red Goblin creature token onto the battlefield.*	{1}{R} [1/1]
2×	**Mudbutton Torchrunner** Creature — Goblin Warrior *When Mudbutton Torchrunner is put into a graveyard from the battlefield, it deals 3 damage to target opponent's creature or your opponent.*	{2}{R} [1/1]

2×	**Raging Goblin** Creature — Goblin Berserker *Haste*	{R} [1/1]
1×	**Reckless One** Creature — Goblin Avatar *Haste* *Reckless One's power and toughness are each equal to the number of Goblins on the battlefield.*	{3}{R} [*/*]
2×	**Skirk Drill Sergeant** Creature — Goblin	{1}{R} [2/1]
1×	**Skirk Fire Marshal** Creature — Goblin *Tap five untapped Goblins you control: Skirk Fire Marshal deals 10 damage to each other creature and each player.*	{3}{R}{R} [2/2]
1×	**Skirk Prospector** Creature — Goblin *Sacrifice a Goblin: Add {R} to your mana pool.*	{R} [1/1]
1×	**Skirk Shaman** Creature — Goblin Shaman *Skirk Shaman can't be blocked except by artifact creatures and/or red creatures.*	{1}{R}{R} [2/2]
1×	**Tar Pitcher** Creature — Goblin Shaman *{T}, Sacrifice a Goblin: Tar Pitcher deals 2 damage to target opponent's creature or your opponent.*	{3}{R} [2/2]
2×	**Boggart Shenanigans** Tribal Enchantment — Goblin *Whenever another Goblin you control is put into a graveyard from the battlefield, Boggart Shenanigans deal 1 damage to your opponent.*	{2}{R}
1×	**Spitting Earth** Sorcery *Spitting Earth deals damage equal to the number of Mountains you control to target opponent's creature.*	{1}{R}
3×	**Tarfire** Tribal Instant — Goblin *Tarfire deals 2 damage to target opponent's creature or your opponent.*	{R}
1×	**Forgotten Cave** Land *When Forgotten Cave enters the battlefield, tap it.* *{T}: Add {R} to your mana pool.* *{R}, Discard Forgotten Cave: Draw a card.*	
1×	**Goblin Burrows** Land *{T}: Add {1} to your mana pool.* *{1}{R}, {T}: Target Goblin creature you control gets +2/+0 until end of turn.*	
22×	**Mountain** Basic Land — Mountain {R}	

A.3 Liliana Vess

This deck doesn't rely on creatures as much as the previous two. Making an opponent discard cards or lose life while gaining it in return is very common with this deck. It can damage or destroy creatures and in some cases use them as their own. It also has a big offensive advantage against the other two, creatures with flying, which they cannot block.

#	Card Name Type Line *Abilities*	Mana Cost [Power/Toughness] or (Loyalty)
1×	**Liliana Vess** Planeswalker — Liliana *+1, {T}: Your opponent discards a card.* *-2, {T}: Search your library for a card, then shuffle your library and put that card on top of it.* *-8, {T}: Put all creature cards in all graveyards onto the battlefield under your control.*	{3}{B}{B} (5)
1×	**Deathgreeter** Creature — Human Shaman *Whenever another creature is put into a graveyard from the battlefield, you gain 1 life.*	{B} [1/1]
2×	**Ghost-Lit Stalker** Creature — Spirit *{4}{B}, {T}: Your opponent discards two cards. Activate this ability only any time you could cast a sorcery.* *{5}{B}{B}, Discard Ghost-Lit Stalker: Your opponent discards four cards. Activate this ability only any time you could cast a sorcery.*	{B} [1/1]
2×	**Vampire Bats** Creature — Bat *Flying* *{B}: Vampire Bats gets +1/+0 until end of turn. Activate this ability only any time you could cast a sorcery.*	{B} [0/1]
1×	**Drudge Skeletons** Creature — Skeleton	{B} [1/1]
1×	**Ravenous Rats** Creature — Rat *When Ravenous Rats enters the battlefield, your opponent discards a card.*	{1}{B} [1/1]
2×	**Fleshbag Marauder** Creature — Zombie Warrior *When Fleshbag Marauder enters the battlefield, sacrifice a creature.* *When Fleshbag Marauder enters the battlefield, your opponent sacrifices a creature.*	{2}{B} [3/1]
2×	**Phyrexian Rager** Creature — Horror *When Phyrexian Rager enters the battlefield, you draw a card and you lose 1 life.*	{2}{B} [2/2]
1×	**Urborg Syphon-Mage** Creature — Human Spellshaper *{2}{B}, {T}, Discard a card: Your opponent loses 2 life and you gain 2 life.*	{2}{B} [2/2]
1×	**Wall of Bone** Creature — Skeleton Wall *Defender*	{1}{B} [1/4]
1×	**Faerie Macabre** Creature — Faerie Rogue *Flying*	{1}{B}{B} [2/2]
1×	**Howling Banshee** Creature — Spirit *Flying* *When Howling Banshee enters the battlefield, each player loses 3 life.*	{2}{B}{B} [3/3]
1×	**Keening Banshee** Creature — Spirit *Flying* *When Keening Banshee enters the battlefield, target creature gets -2/-2 until end of turn.*	{2}{B}{B} [2/2]
2×	**Twisted Abomination** Creature — Zombie Mutant *{2}, Discard Twisted Abomination: Search your library for a Swamp card and put it into your hand. Then shuffle your library.*	{4}{B} [5/3]

1×	**Skeletal Vampire** Creature — Vampire Skeleton *Flying* *When Skeletal Vampire enters the battlefield, put two 1/1 black Bat creature tokens with flying onto the battlefield.* *{3}{B}{B}, Sacrifice a Bat: Put two 1/1 black Bat creature tokens with flying onto the battlefield.*	{4}{B}{B} [3/3]
1×	**Genju of the Fens** Enchantment — Aura *Enchant creature; Enchanted creature gets +1/+1.* *{1}{B}: Target creature you control gets +1/+1 until end of turn.* *When Genju of the Fens is put into a graveyard, you may return it from your graveyard to your hand.*	{B}
1×	**Bad Moon** Enchantment *Black creatures get +1/+1.*	{1}{B}
2×	**Sign in Blood** Sorcery *Your opponent draws two cards and loses 2 life.*	{B}{B}
1×	**Vicious Hunger** Sorcery *Vicious Hunger deals 2 damage to target opponent's creature and you gain 2 life.*	{B}{B}
1×	**Ichor Slick** Sorcery *Target opponent's creature gets -3/-3 until end of turn.* *{2}, Discard this card: Draw a card.*	{2}{B}
1×	**Hideous End** Instant *Destroy target opponent's nonblack creature. Your opponent loses 2 life.*	{1}{B}{B}
1×	**Snuff Out** Instant *Destroy target nonblack creature.*	{2}{B}
2×	**Tendrils of Corruption** Instant *Tendrils of Corruption deals X damage to target opponent's creature and you gain X life, where X is the number of Swamps you control.*	{3}{B}
1×	**Mutilate** Sorcery *All creatures get -8/-8 until end of turn.*	{2}{B}{B}
1×	**Rise from the Grave** Sorcery *Put target creature card in a graveyard onto the battlefield under your control.*	{4}{B}
2×	**Corrupt** Sorcery *Corrupt deals damage equal to the number of Swamps you control to target opponent's creature or your opponent. You gain life equal to the damage dealt this way.*	{5}{B}
1×	**Enslave** Enchantment — Aura *Enchant creature; You control enchanted creature.* *At the beginning of your upkeep, Enslave deals 1 damage you.*	{4}{B}{B}
2×	**Polluted Mire** Land *When Polluted Mire enters the battlefield, tap it.* *{T}: Add {B} to your mana pool.* *{2}, Discard Polluted Mire: Draw a card.*	
23×	**Swamp** Basic Land — Swamp {B}	

Appendix B

CD Contents

The following structure is used:

- [**docs**]
  - *user_doc.pdf* — user documentation
  - *prog_doc.pdf* — programmer's documentation
  - *EN_Magic_Basic_Rulebook_20090710.pdf* — Basic Rulebook of Magic 2010
  - *MagicCompRules_20090708.pdf* — Comprehensive rules
- [**experiments**]
  - *experiments.rar* — binaries and data files as they were used for testing
  - *logs.rar* — complete logs from eperiments [473.3 MB after extraction]
  - *results_only.rar* — processed logs, contains only results of duels and games
  - *individual_results.pdf* — results of individual games
- [**install**]
  - *maid.rar* — application's binaries and data
  - *dotnetfx35setup.exe* — Microsoft .NET Framework 3.5 Service Pack 1
  - *dotnetfx35.exe* — Microsoft .NET Framework 3.5 Service pack 1 (Full Package)
- [**source**]
  - *source.rar* — source code of the environment and agents
- *vejmola_bc-thesis.pdf* — the thesis in PDF format