Faculdade de Engenharia da Universidade do Porto

High-Level Language to build Poker Agents

Pedro Daniel da Cunha Mendes

Report of Dissertation
Masters in Informatics and Computer Engineering

Supervisor: Luis Paulo Gonçalves dos Reis

July 2008

High-Level Language to build Poker Agents

Pedro Daniel da Cunha Mendes

Report of Dissertation
Master in Informatics and Computing Engineering

Approved in oral examination by the comittee:

Chair: Armando Jorge Miranda de Sousa, Ph. D.

External Examiner: João Balsa, Ph. D.

Internal Examiner: Luis Paulo Gonçalves dos Reis, Ph. D.

July of 2008

Abstract

The game of Poker has been, in the past years, one of the most interesting subjects in Artificial Intelligence because it is a game were players don't have complete information about the entire state of the game and where the element of chance is always present. With its continuous growth, it also became clear the importance that Artificial Intelligence can have in the game of poker. Building poker autonomous agents that are able to play against human players at a competitive level may be a very good way of for humans to learn to evolve and strengthen up the player's game abilities. However, Artificial Intelligence is a subject that is not accessible to the normal poker player because it requires years of study.

The purpose of this work was to fill the gap between human poker players and Artificial Intelligence in poker, enabling normal users to build their own poker playing agents without needing Artificial Intelligence knowledge.

In this work, the first step was to create a high-level language of poker concepts, PokerLANG that enables the construction of poker agents with a "language" that normal poker players would comprehend. The language follows a format similar to the RoboCup Coach Language (Coach Unilang), a language developed to enable online coaches to change the behaviour of simulated soccer players during games in the Simulated League of the robotic soccer international competition - RoboCup.

An application, with a simple graphical interface was created in order to support and help the users to create poker strategies based on the poker language developed. This application was developed in Adobe Flex enabling the creation of a simple and pleasant graphical interface to support the users.

The last step of this work was to create a poker agent that supports the poker strategies built in the Flex application. This agent is able to read a given poker strategy developed in PokerLANG, parsing it into its internal data structures and use a standard communication protocol to connect and play, using the pre-defined strategy, in common poker servers.

The results achieved prove the usability of the application and the usefulness of the supporting language. The example agents created showed interesting results against agents created by experts in the area, proving the simplicity of making a poker agent suited for competition using the application developed.

Resumo

O poker tem sido, nos últimos anos, um dos mais interessantes temas de estudo em Inteligência Artificial porque é um jogo em que os jogadores não têm informação perfeita do estado do jogo e aonde o elemento de casualidade está sempre presente. Com o seu crescimento contínuo, ficou à vista a importância que a área de Inteligência Artificial pode ter no jogo. Construir agentes autónomos de poker, capazes de jogar contra humanos num nível competitivo, pode ser uma boa maneira dos jogadores evoluir e fortalecer as suas capacidades. Mas a área de Inteligência Artificial é uma área que não está acessível ao comum jogador de poker porque necessita de anos de estudo.

O objectivo deste trabalho foi diminuir a distância entre os jogadores de poker e a área de Inteligência Artificial no jogo, permitindo aos utilizadores, construir os seus próprios agentes de poker sem precisarem de conhecimentos na área de Inteligência Artificial.

Neste trabalho, o primeiro passo foi construir uma linguagem de alto nível de conceitos de poker para permitir a construção de agentes de poker com uma "língua" que os jogadores pudessem compreender. A linguagem segue a formatação da RoboCup Coach Language (Coach UniLANG), uma linguagem desenvolvida para permitir que treinadores "online" mudassem os comportamentos de jogadores de futebol simulado durante os jogos da Simulated League da competição internacional de futebol robótico - RoboCup.

Com uma simples interface gráfica, foi criada uma aplicação para suportar e ajudar os utilizadores a criarem estratégias de poker baseadas na linguagem de conceitos desenvolvida. Esta aplicação foi desenvolvida em Adobe Flex permitindo a criação de uma interface simples e agradável para facilitar o trabalho dos utilizadores.

O último passo deste trabalho foi criar um agente de poker que suporta as estratégias de poker criadas na aplicação. Este agente é capaz de ler a estratégia de poker desenvolvida na linguagem PokerLang, gravando a mesma em estruturas de dados internas e usando o protocolo de comunicação para se ligar e jogar, usando a estratégia pré-defenida, nos servidores de poker.

Os resultados alcançados, provam a usabilidade da aplicação e a utilidade da linguagem que a suporta. Os exemplos de agentes criados mostraram resultados interessantes contra agentes desenvolvidos por outros especialistas na área, provando a simplicidade com que se pode fazer um agente de poker, suficientemente competitivo, na aplicação.

Acknowledgements

I would like to thanks to all my colleagues who in any way contributed to this project's success, either by clearing some doubts or even giving feedback and suggestions which allowed me to improve my project.

I would also like to thank Dinis Felix and Mauro Gomes for their participation in this project as well as my girlfriend Helia Gajo for her support and her patience.

Special thanks to my supervisor Luis Paulo Reis for his continued support and expertise provided concerning this project's domain as well as all the ideas that he gave.

Contents

List of Figures

List of Tables

Chapter 1

1. Introduction

Poker is the most played card game in the world. Played by millions around the world, poker has become a very profitable business. Giving millions and millions of dollars in prizes, in almost every country, it became a case study in different subjects such as Mathematics, Artificial Intelligence, Sociology, among others. As a game of incomplete knowledge, risk management, opponent modeling and dealing with unreliable information, it transformed poker as an important research in Computer Science, especially Artificial Intelligence.

With the recent online poker boom, a large number of tools have been created to assist players in their game. Most of the tools created are statistic programs that save information of the games you play, creating statistical knowledge of your opponents, to help you make the right moves. As Gus Hansen said "Poker is a game which needs a lot of study, but the most important thing is to play. You need to play, to gain experience." [Han08]

The goal of this work is to create a tool that helps players around the world to improve their game to achieve perfection, while playing. A tool that would allow users to create specified opponents has Artificial Intelligence Agents.

Agents can simulate behaviors, but the downside is that Artificial Intelligence is a complicated subject, that requires a lot of knowledge of computer programming, knowledge that most of the poker players don't have. So the tool must create poker agents in a language that any player could understand.

The tool created in this project, is a tool that uses a language of poker concepts to create poker agents. The idea is to create a simple method of creating poker rules using these concepts, to become accessible to any user.

1.1 Artificial Intelligence and Games

The term Artificial Intelligence was created in 1956 by John McCarthy at the Massachusetts Institute of Technology. Artificial Intelligence includes many subjects such as:

- Robotics: programming computers to react to sensory stimulation like seeing and hearing.

- Expert systems: computer systems design to assist humans in real-life decisions (Medicine, Economics).
- Natural language: computer systems design to understand human languages.
- Game playing: computer programming to play games against human players.

At the present moment, Artificial Intelligence is not yet sufficiently developed to perform perfect human behavior in most domains. The area with most advances is game playing. In games such as chess and checkers, computer programs are able to beat humans in a regular basis. In May, 1997, an IBM super-computer called Deep Blue [New96] successfully defeated the world chess champion Gary Kasparov.

But games like chess and checkers handle with perfect information. Just by looking to the board, each player has access to all the information. In games like Poker the job becomes more difficult. Each player has hidden cards, so you can't access all the information which becomes imperfect knowledge. Another problem is that poker has elements of chance. The cards dealed to the table are random and change the game state dramatically. Games with these elements are called stochastic games with incomplete information.

Artificial Intelligence researchers have used games as research tests for years. Because games have well-defined rules, they become easy to explore comparing to other problems in Artificial Intelligence. Despite the fact that their research focused on games of complete information (Deep Blue), in recent years researchers start to focus on stochastic games. They offer many new interesting challenges not present in the traditional perfect information games.

1.2 Motivation

The company MECN recently made a study that analyses one of the hottest topics in the gambling industry – online poker. As the co-author of the study, explains "Online poker is the perfect symbiosis of two mega trends in gambling: online gambling and poker." [Oel05]. It began with a triumphant entrance in USA, but in the latest years, it leaded to a global Poker wave on the web. In this study, MECN achieve some conclusions [Oel05]:

- The amount gambled on poker websites around the world in 2005 is estimated to be more than USD 60 billion.
- More than 60% of the industry experts surveyed believe that online poker will be the dominant offer in online gambling in 2-3 years.
- 75% of the industry experts surveyed believe that the global annual rake/commission in the 2-3 years will be more than USD 4 billion.

Despite the fact that this movement started in USA, 52% of the experts believe that Europe will be potentially the future of online Poker. The actual market leader, PartyPoker, is already launching its presence on television, following the steps of the USA competition.

In the same way online poker achieved a financial growth, poker tools have also increased in number, creating an income around USD 60 million. Most of the tools, as refer before, are statistical tools (Poker Office, Win Hold'em). The area of creating your own opponents is yet to be explored, and it would represent an enormous aid for players to perfect their game and increase their abilities. Making Artificial Intelligent Agents with a language accessible to everyone is a challenge in computer programming worth trying for, as well as potentially financially rewarding.

1.3 Goals

The main goal of the project is to create a powerful tool, capable of creating Poker Agents, through rules of concepts so that any user, even without computer programming knowledge, can easily use the tool to create competitive autonomous poker players.

The first goal is to create a language of concepts, which includes the main ideas behind poker strategies, moves and behaviors. This language should be complete enough to create poker strategies enabling to play at a professional level but yet to be easily transformed into logical rules for being interpreted by an agent.

Secondly, the goal is to build a graphical interface for this language, which allows the user to create rules, defining complete strategies in a simple and straightforward way.

After the first two goals are achieved, a poker agent should be developed, capable of interpreting the rules created by the user. With the help of a poker server application this agent performance should be tested against other agents and human players.

1.4 Summary of contents

This project is structured in 7 chapters.

The first chapter contains the introduction to the thesis. Games and Artificial Intelligence are presented in a simple approach, and it is explained the motivation for this kind of work.

In the second chapter the game variant used in this thesis (Texas Hold'em) is explained in detail and some previous work done in Artificial Intelligence applied to the game of Poker is presented. It is also presented some information about the existing tools that may be applied to the game.

The third chapter shows the language of concepts created - PokerLANG. An explanation of each poker concept used is available, as well as, examples of rules constructed using this language.

The fourth chapter describes the interface implemented to use the language, its possibilities and a detailed description of the features accessible to the user.

The fifth chapter gives an explanation the Agent built to interpret and follow the rules created by the user previously.

In Chapter 6, some test and results are shown based on the agents created using the main application. Some tables and graphs are presented to explain the work done and the results achieved.

The conclusions of the project are presented in the seventh chapter, as well as, the next steps to be made in order to increase the potential of this project.

Chapter 2

2. State of Art

In Artificial Intelligence (AI), games have proven to be challenging and rewarding for research in the subject. There are many interesting successes with computer programs defeating humans in many different games. Stories like Chinook (checkers) [Sch92], Logistello (Othello) [Bur97], Deep Blue [New96] and Hydra [Don05] (chess), and TD Gammon (backgammon) [Tes95], represent the advances made in Artificial Intelligence. But the fact is, all these research work were created for games with perfect information. The agents may access all the information in the game by just looking at the board which turns the problem a lot easier to solve with AI techniques.

In poker, everything changes. There is imperfect information because each player owns cards that are not accessible to other opponents. Another factor is that, poker is a stochastic game. Because of the shuffle of the deck, it introduces the element of chance.

But these reasons are what turn poker into a great challenge in Artificial Intelligence. Trying to predict the opponent's cards is a major adversity in the game, but once again, this characteristic turns poker into an excellent game to explore.

2.1 Poker

Poker is a very popular type of card game, played by millions around the world. It contains many variations (Texas Hold'em, Stud, Five-Card, among others) but any of them results in the same purpose. Each player bet on the value of the card combination ("hand") in their possession, by putting a certain bet in the table. The "hand" is the combination of five cards and the winner of the game is the player who has the best combination in the table, accordingly to an established hand rankings hierarchy. Another way to win is to be the only player remaining in the hand at the end of the betting rounds (the player o makes an un-called bet).

The amount of information a player can access in poker, depends on which poker variant is played. Some poker variants use only hole cards (cards that only the player can see), some use community cards (cards placed in the center of the table that can be used by any player to create the best combination) and some use unconcealed cards (cards that belong to each player but can be seen by the opponents). The mix of these 3 types of cards creates all the variations existing in poker.

Poker is also a game of order. In each round, there is a designated “dealer” (the player who distributes the cards among the other players). It defines the last position on betting which is called the “button”. The right to deal a hand passes by in the end of each round, rotating clockwise among the players. In a casino, a house dealer handles the cards in each round, but the dealer “button” is represented by a white plastic disk, that rotates around the table.

The order of betting is determined by the dealer “button”. The two players on his left are forced to make a symbolic bet called the “blinds”, to create an initial stake for the players to contest. The player right after the dealer is forced to bet a Small Blind and the other player right after him is forced to bet a Big Blind (which is twice the value of the Small Blind). The blinds can change value depending on the type of game played. In poker tournaments, along the time, the blinds value rises to create a larger stake to contest.

The dealer, shuffle the cards and deals the appropriate number of cards among the players, one at the time. Cards may be dealt face-up or face-down, and the number of cards dealt depends on the variation of poker. After an initial deal, it starts the first betting round (the number of betting rounds depend on the variation). Between the rounds, the player’s hand changes, often by dealing extra cards or replacing the old cards. At the end of each round, all the bets made by the players are gathered at the center of the table. The gathering of the bets is usually refereed as “the pot”.

At any time, during the betting round, if one player’s bet is not called, the hand terminates immediately and “the pot” is awarded to the un-called bettor, and no cards are required to be shown. This introduces the most appealing concept of poker, the bluff. Bluffing is a primary feature of poker, and it happens when you “force” every player to “fold” (not call your bet) when you don’t have a top hierarchic hand. At the end of the last betting round, the players remaining in the game are required to show their hand, in the order of the betting. When a players show a hand, if the next player’s hand is not enough to beat the shown hand, this player is not required showing his hand.

The hierarchy of the hands in poker is defined as follow (from the highest to the lowest):

Royal Flush: An Ace, King, Queen, Jack and Ten in the same suit.

In the event of a tie: Two or more Royal Flushes split the pot.

Fig 1: Royal Flush

Straight Flush: Five cards in sequence, of the same suit.

In the event of a tie: Highest rank at the top of the sequence wins.

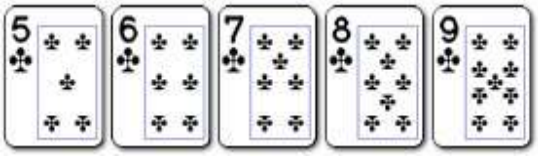

Fig 2: Straight Flush

Four of a Kind: Four cards of the same rank, and one side card.

In the event of a tie: Highest four of a kind wins. In community card games where players have the same four of a kind, the highest fifth side card ("kicker") wins.

Fig 3: Four of a Kind (poker)

Full House: Three cards of the same rank, and two cards of a different, matching rank.

In the event of a tie: Highest three matching cards wins the pot. In community card games where players have the same three matching cards, the highest value of the two matching cards wins.

Fig 4: Full House

Flush: Five cards of the same suit.

In the event of a tie: The player holding the highest ranked card wins. If necessary, the second-highest, third-highest, fourth-highest, and fifth-highest cards can be used to break the tie.

Fig 5: Flush

Straight: Five cards in sequence.

In the event of a tie: Highest ranking card at the top of the sequence wins (the Ace may be used at the top or bottom of the sequence, and is the only card which can act in this manner).

Fig 6: Straight

Three of a kind: Three cards of the same rank, and two unrelated side cards.

In the event of a tie: Highest ranking three of a kind wins. In community card games where players have the same three of a kind, the highest side card, and if necessary, the second-highest side card wins.

Fig 7: Three of a Kind

Two pair: Two cards of a matching rank another two cards of a different matching rank, and one side card.

In the event of a tie: Highest pair wins. If players have the same highest pair, highest second pair wins. If both players have identical pairs, highest side card wins.

Fig 8: Two Pair

One pair: Two cards of a matching rank, and three unrelated side cards.

In the event of a tie: Highest pair wins. If players have the same pair, the highest side card wins, and if necessary, the second-highest and third-highest side card can be used to break the tie.

Fig 9: One Pair

High card: Any hand that does not qualify under a category listed above.

In the event of a tie: Highest card wins, and if necessary, the second-highest, third-highest, fourth-highest and smallest card can be used to break the tie.

Fig 10: High Card

2.2 Texas Hold'em

Texas Hold'em is the most popular poker game, and is the variant that will be considered in this project.

After a slow introduction in the poker world, Texas Hold'em had its boost in 1998 with the release of the film "Rounders" [Dah98]. In the film it is told the fancy history of a poker player (represented by famous actor Matt Damon) and his adventures in the underground poker world. Thanks to the dramatic representation of the swings that a poker player faces in this game, it became very appealing to people all around the world. On the next two years, helped by the sudden interest in the game, a lot of literacy had come to the stores, available to everyone. Suddenly Texas Hold'em had become the most played card game in the world, covered by television and growing on internet.

Texas Hold'em is a community card game. Each player is dealt two hole cards, followed by five community cards. With all the seven cards (2 hole + 5 community) each player creates the best possible 5-card combination. Because of this, Texas Hold'em becomes an opportune game for strategic analysis, including mathematical analysis.

Texas Hold'em is usually played using small and big blind bets. A dealer button is used to represent the player in the top position. In each hand, this button rotates around the table in clockwise order. With this change of the dealer button, the blinds position also change in each hand. The player at the left of the dealer is the small blinder and the next player the big blinder. These players are "forced" to make this bets in order to build a stake to contest. The value of the blinds change along the time, increasing in a define

amount of time (in live poker is usually in each hour, in online is usually after 10 minutes). In figure 11, we can easily see how this works:

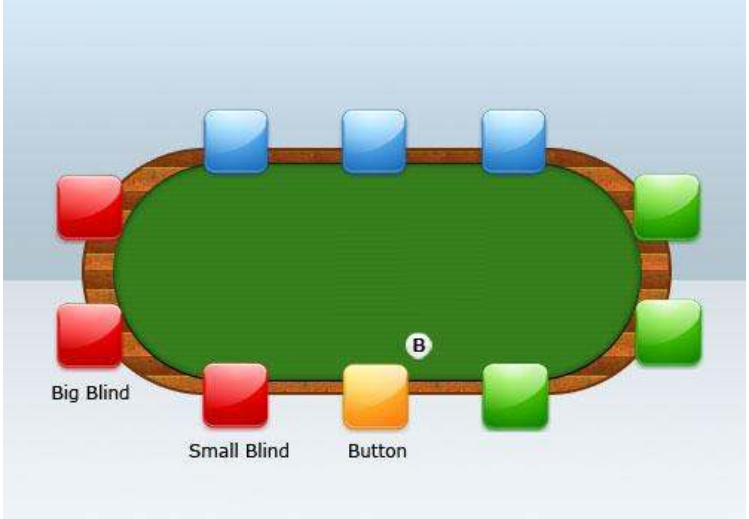

Fig 11: The positions at the table

After all the players have their two hole cards, the initial betting round starts, in turn, starting by player at the left of the big blinder. At his turn, each player can decide to stay in the hand (check/call), increase the bets (raise) or abandon the hand (fold). When the initial bet round is over, 3 community cards are placed in the center of table. These 3 cards are called "the flop". Then a new betting round starts and the first player to bet is the small blinder. If the small blinder already abandoned the hand then the player at his left starts this betting round.

A fourth card comes after the end of the second betting round. This card is refereed as "the turn", followed by a third betting round. At the end, a last fifth card, called "the river", is placed at the center of the table, initiating the final bet round.

When all betting rounds are over and all the 5 community cards are at the table, the showdown starts, Each player shows their two hole cards and the player who has the best 5-card combination out of seven cards, wins the hand and of course all the bets.

Here's an example of a showdown, from the Doyle Brunson's book "Super System - A Course in Power Poker" [Bru79], between 4 players:

Fig 12: Sample showdown

Each one of the players plays the best 5 cards with the seven cards available (2 in the hand and 5 on the board).

The combinations of this showdown could be seen in figure 14.

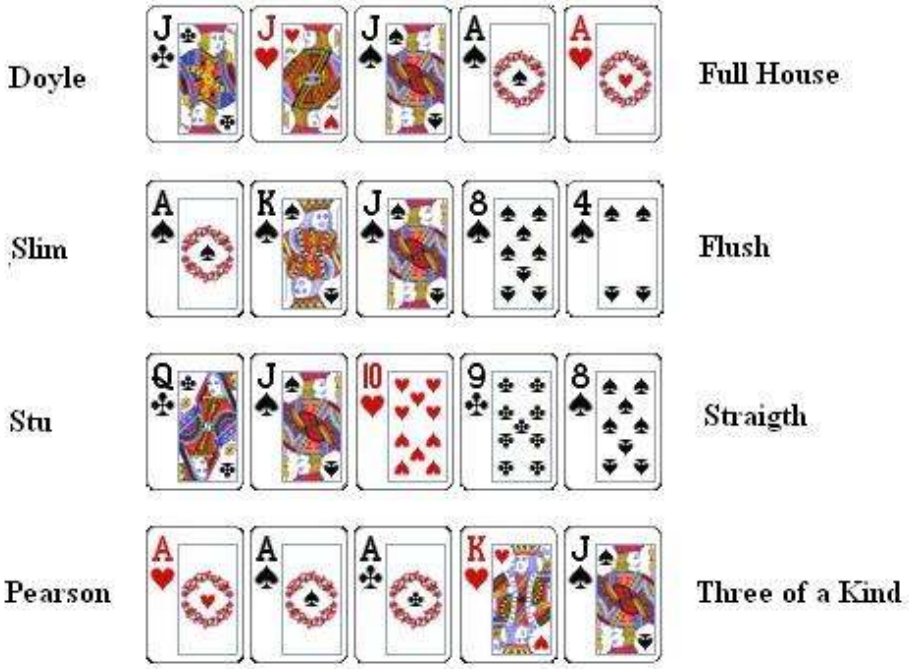

Fig 13: Combinations of cards at the showdown

In this case, Doyle's full house is the best hand, with Slim in 2nd (flush), Stu in 3rd (straight) and Pearson last (three of a kind).

2.3 Related Work

This project is an evolution for the advances made in Artificial Intelligence related to Poker. It is also a project of a helping tool for poker players to perfect their game playing abilities and creates a new area for exploration in support tools for poker, specifically Texas Hold'em. The research in the area started with creation of the University of Alberta (U.A.) Computer Poker Research Group [Una08]. Several tools were created in the follow years, mostly statistical tools (Poker Office [Off08]), as well as in creating very simplistic agents to test the players game (Poker Academy [Aca08]).

This project wants to strengthen up the area of creating agents and also creating a new bunch of possibilities for future advances.

2.3.1 The U.A. Computer Poker Research Group

The University of Alberta CPRG (Computer Poker Research Group) is the major contributor to the academic literature on poker game-playing using Artificial Intelligence [Una08].

The CPRG was formed in 1997, leaded by Darse Billings who had been a professional poker player from 1996 to 1999, after several years of studying and extending poker theory.

The thesis "Algorithms and Assessment in Computer Poker" [Bil06] from its leader, describes the work done at University of Alberta. The main topics that influenced this work are:

- Pre-flop strategies;
- Hand strength, hand potential and effective hand strength;
- Opponent Modelling.

One of the greatest advances made by CPRG, are the betting strategies. They are divided in two different types of betting (pre-flop and after the flop betting). The pre-flop betting approach is extremely simple, because the player has only to consider its two hole cards. The after the flop betting is more challenging because the computer must analyze all possible holdings of the opponents and combine them with the community cards already at the center of the table.

Before the flop there are 1326 possible hands that a player can hold. In University of Alberta's work, the value of each hand is obtained with a technique called a roll-out simulation. Basically is a simulation that plays several million games where all players called the first bet. This provides an approximation to the relative values of the hands with reasonable accuracy for the given situation [Pap98] [Pen99].

With these calculations it could be observed which hands are more powerful. The interesting factor in this work is that it was possible to correlate the results with the Sklansky groups. Sklansky is a professional poker player who ranked the initial 2-cards that a player can hold into different groups, according to their poker value. The calculations made by CPRG fitted on the Sklansky groups as followed:

Table 1: CPRG Values VS Sklansky Groups

Group 1		**Group 2**		**Group 3**		**Group 4**	
+2112	AA	+714	TT	+553	99	+481	T9s
+1615	KK	+915	AQs	+657	JTs	+515	KQo
+1224	QQ	+813	AJs	+720	QJs	+450	88
+935	JJ	+858	KQs	+767	KJs	+655	QTs
+1071	AKs	+718	AKo	+736	ATs	+338	98s
				+555	AQo	+449	J9s
						+430	AJo
						+694	KTS
Group 5		**Group 6**		**Group 7**		**Group 8**	
+364	77	+304	66	+214	44	-75	87o
+270	87s	+335	ATo	+92	J9o	+87	53s
+452	Q9s	+238	55	+41	43s	+119	A9o
+353	T8s	+185	86s	+141	75s	+65	Q9o
+391	Kj0	+306	KTo	+127	T9o	-129	76o
+359	QJo	+287	QTo	+199	33	-42	42s
+305	JTo	+167	54s	-15	98o	-83	32s
+222	76s	+485	K9s	+106	64s	+144	96s
+245	97s	+327	J8s	+196	22	+85	85s
+538	A9s			+356	K8s	-51	J8o
+469	A8s			+309	K7s	+206	J7s
+427	A7s			+278	K6s	-158	65o
+386	A6s			+245	K5s	-181	54o
+448	A5s			+227	K4s	+41	74s
+422	A4s			+211	K3s	+85	K9o
+392	A3s			+192	K2s	-10	T8o
+356	A2s			+317	Q8s		
+191	65s						

Despite the similarity to the Sklansky groups, the CPRG preferred to use their own values, first because they were quantity values instead of qualitative values like Sklansky's and majorly because the purpose of the work was to avoid any use of human knowledge whenever was possible.

After the flop, things were a bit more complicated. In the work developed, the CPRG based their formula in four major calculations:

- Hand Strength (HS);
- Positive Potential (PPot);
- Negative Potential (NPot);
- Effective Hand Strength (EHS).

The hand strength (HS), is the probability that a given hand is better than that of an active opponent. Suppose an opponent is equally likely to have any possible two hole card combination. All of these opponent hands can be enumerated, identifying hand is better (+1), tied (+ 1/2), or worse (0). Taking the sum and dividing by the total number of possible opponent hands gives the (un-weighted) hand strength.

After the flop, there are two community cards yet to come (the Turn and the River). These cards can affect tremendously on the value of player's hand. So CPRG calculated the Positive Potential and the Negative Potential of these cards.

The positive potential (PPot) is the chance that a hand that is not currently the best improves to win at the showdown. The negative potential (NPot) is the chance that a currently leading hand ends up losing.

PPot and NPot are calculated by enumerating over all possible hole cards for the opponent, like the hand strength calculation, and also over all possible board cards. For all combinations of opponent hands and future cards, we count the number of times the agent hand is behind, but ends up ahead (PPot), and the number of times hand is ahead but ends up behind (NPot).

But calculating all the effect possibilities of the Turn and the River can be very expensive, given the real-time constraints. So this calculation was separated and the betting only considers the next card to come. It was the best alternative since when your opponents have possibilities of improving their hand against your current top hand. The best way is to avoid these opponents by throwing them out of the hand with your bets.

The effective hand strength (EHS**)** combines hand strength and potential to give a single measure of the relative strength hand against an active opponent.

One simple formula for computing the probability of winning at the showdown is:

$$Pr(win) = Pr(ahead) \times Pr(opp\ not\ improve) + Pr(behind) \times Pr(we\ Improve) \quad (Eq\ 1)$$

$$Pr(win) = HS \times (1 - NPot) + (1 - HS) \times PPot \quad (Eq\ 2)$$

In practice, the correct act is bet when having the best hand, regardless of negative potential, so that an opponent with a marginal hand must either fold, or pay to draw. Hence, NPot is not as important as PPot for betting purposes. Since the interest is the probability of the hand is either currently the best, or will improve to become the best, one possible formula for EHS sets NPot = 0, giving:

$$EHS = HS + (1 \times HS) \times PPot \quad (Eq\ 3)$$

This has the effect of betting a hand aggressively despite good draws being possible for opponent hands, which is a desirable behaviour.

For n active opponents, this can be generalized assuming that the same EHS calculation is the same for all opponents:

$$EHS = HS^n + (1 - HS^n) \times PPot \quad \text{(Eq 4)}$$

With this methodology, the CPRG was able to create different kinds of agent that choose different moves considering the values of these calculations.

2.3.2 The LIACC's Texas Hold'em Simulator

The Artificial Intelligence and Computer Science Laboratory (LIACC) is a research center at the University of Porto (UP), created in 1988. This center contains several researchers from the Faculty if Sciences (FCUP) and Faculty of Engineering (FEUP).

The center is involved in a large number of projects with national and international cooperation and it covers several subjects of investigation like Natural Languages Processing, Robotics and Distributed Artificial Intelligence among others.

Previously, this center has developed a Poker Simulator system called "LIACC's Texas Hold'em Simulator, which supports one of the goals of this work. The software was developed in C/C++ and it is based on a Server that communicates with several Clients through sockets under a predefined protocol. In this simulator, two versions of the Client were developed. One of the versions may be controlled by humans and another is an autonomous agent that runs with an incorporated simples AI poker playing algorithm.

This AI algorithm is mostly based on the betting strategy described by Billings [Bil06].

The implemented Server is capable to support ten different Clients (using a standard poker protocol based on TCP/IP) and allows the user to define all major characteristics of a poker game (initial stack, the value of the blinds). The number of simulations and the name of the log file that will save all hands played are also available to decide.

The protocol used by this simulator is compatible with the one developed at University of Alberta Computer Poker Research Group. This protocol will be explained in more detail in Chapter 5. Although this is not explored on this work, using the same protocol makes it possible to use the agents developed against other agents developed in distinct projects and to participate in competitions like "2007 AAI Poker Competition" [Smi07]that will be introduced further more in this thesis.

In figure 2.14, we can see an example of LIACC's Texas Hold'em Simulator interface:

Fig 14: LIACC's Texas Hold'em Simulator Interface

2.3.3 AAAI Computer Poker Competition

There are several poker agents competitions available enabling agents developed by distinct researchers and universities to participate but probably the most notorious is the "AAAI Poker Competition" [Smi07]. It is a competition sponsored by the University of Alberta [Una08] and the known AAAI conference that started in 2006.

In the first year, there were only five participants, three designed at universities and two by individuals. The three participants that were designed by universities were the "Hyperborean" [Bil03] developed by "University of Alberta", the "GS2" [Gil06] designed at "Carnegie Mellon University" in Pittsburgh, US and the last one, the "Monash BPP" [Kor99] was created at "Monash University" in Victoria, Australia.

The other participants were developed individually: the "BluffBot" created by Teppo Salonen in Irvine, US and the "Teddy" developed by Morten Lynge in Ikast, Denmark.

In 2006 the competition ran on 16 Windows machines in the Computer Labs at the University of Alberta (one server machine, 14 clients machine and one extra in cas of problems).

On the next year, the competition grew. In 2007, the competition had 15 participants from 7 countries and 43 different bots. This time, the matches were played in 32 machines running for a whole month, playing over 17 million hands of poker.

In this year, the competition is again being held at the University of Alberta, played from June 1st to July 15th of 2008.

2.3.4 Poker Academy

The Poker Academy [Aca08] is probably the most advanced Poker tool in the market. Besides giving statistical support to the players, it also includes several tutorials for a player perfect his game. It provides an interesting tool that creates poker agents with user specified values, entering in competition with the purpose of this work. But the major problem with this feature is that the user can define with a lot of specifications the pre-flop strategy of the agent. After the flop, things change and it only gives the user the possibility to define human characteristics to the agent (aggressiveness, tightness among others) as we can see in figure 2.15.

Fig 15: Poker Academy Interface

With these definitions, the user can't know the rules of behavior that are behind these specifications, leaving the user in the dark.

The objective of this work is to give full control to the user, so he can define all the rules that the agent will obey. This difference for the newbie's in Poker doesn't affect that much, but to the major players already in the circuit it creates great damage. Not being able to build poker agents with very specific actions creates a sense of useless to this kind of tools.

2.3.5 Poker Office

Poker Office [Off08] is the most popular statistic tool available to online poker players. It has several protocols with major poker online rooms with full connection to their applications. Most of the statistical tools existing in the market collect their information with processing image techniques. This limitation collides with several online poker rooms that not let you customize the layout of their interfaces.

Poker Office gives the user statistical support of all the players they play against along the way. It contains a very good feature that gives this type of information at runtime when the player is effectively playing at a table. The Live Tracker connects to the table, registries the opponents and gives the player past information he has about them as well as keep gathering more information. It provides the user with statistics such as percentages of calling bets, patterns of raises among other important information. Figure 2.16 shows an example of Poker Office interface.

Fig 16: Poker Office Interface

Has can be seen in the image, it also provides graphical information, making it easier for the user to read the information given.

2.3.6 Online Poker Rooms

As previously refereed, online poker is an industry with a continuous growth all around the world. There are several major poker rooms with a lot of protocols with supporting tools (Poker Office) and top poker players on their staff. At the time, the three most popular online poker rooms are:

- PokerStars [Sta08]
- Full Tilt Poker [Ful08]
- Party Poker [Par08]

These poker rooms have television coverage and sponsor tournaments with million dollars in prize money all over the globe.

PokerStars [Sta08] was created in 2000. Over the years it had a continuous growth until 2003 when it had its boost. It was the year when one of the players qualified and won

the World Series of Poker (WSOP), the major poker tournament in the world. This victory was conquered by Chris MoneyMaker that with a starting investment of USD $39 won the ultimate prize of the WSOP of 2.5 million USD. After this victory, PokerStars conquered a large number of new users with the dream of being the next "MoneyMaker". PokerStars is also the official sponsor of the European Poker Tour (EPT), a European circuit that goes by in all major casinos across the continent.

Fig 17: PokerStars Interface

Party Poker [Par08] in a well known online poker room created in 2001. It gain instantly success becoming the major online poker room in the world. But when the money transactions between U.S.A. and Europe became forbidden, the Party Poker closed the entrance to American players which represented a great loss of competitively. With new investors and an intensively search for new markets, Party Poker regains its place as one of the top online poker rooms.

Fig 18: Party Poker Interface

Finally, the online poker room on top of the hierarchy at the present time is the Full Tilt Poker [Ful08]. Created in 2004, Full Tilt Poker had a rocket entrance in the industry. It sponsors a list of great professional players (Howard Lederer, Phil Ivey, Chris Ferguson, among others) who are also available to support and teach the newbie's on their site. It provides games in all variants of Poker, with online tournaments capable of gathering 4.000 players in each one.

Fig 19: Full Tilt Poker Interface

To better understanding of the differences of each online poker room, a table with each one's characteristics can be seen in Table 2.2:

Table 2: Comparison of poker online rooms

	PokerStars	**Party Poker**	**Full Tilt Poker**
Software	Excellent	Excellent	Very Good
Interface	Excellent	Very Good	Very Good
Opponent Level	Good	Good	Good
Suports Poker Tools	Yes	Yes	No
Live Statistics	Yes	Yes	Yes
Hand History	Yes	Yes	Yes

By the comparison made, PokerStars is the best online poker room available. The possibility of using Poker Tools is a great advantage to an online poker room. It represents a great flaw on Full Tilt Poker, not being able to support these tools.

2.4 Conclusions

This chapter described the game of poker, its variants and rules with emphasis on Texas Hold'em poker. It also overviewed some research work on AI applied to poker and presented some information about the existing tools that may be applied to the game.

This overview enables to conclude that several tools exist that help players to play by giving statistical information and opponent characteristics. There are also several agents created by distinct universities capable of playing poker at a basic level. However, there is no simple interface enabling normal players to create poker agents capable of playing at a good level.

This conclusion leads to the development of a simple language that enables the definition of simple poker strategies that will be the basis for the development of our application for creating poker playing agents.

Chapter 3

3. Language of Concepts

Many players of different games would like to build their own opponents, but building computer agents is a subject that requires a lot of knowledge and practice on computer programming.

A way to avoid this limitation is to make High-Level Languages supported with simple interfaces. With these programs users would have the possibility to "program" Artificial Intelligence without the need to learn computer programming.

The first step to build such programs is to create a high-level language of concepts about the subject, in the case of this work, Poker. Using well-known concepts of poker, it was made a language to help the users to create simple rules that determine the behavior of the poker agent. This language, called PokerLANG is based on the grammar of Coach UniLANG [Rei02].

3.1 Coach UniLANG and CLANG

Coach UniLANG is a standard language grammar created by Luis Paulo Reis and Nuno Lau with two main objectives [Rei02]: to coach FC Portugal 2001 team [Rei01a] [Rei01b][Lau01] and as a proposal to be used in Fukuoka 2002 RoboCup coach competition [Fuk02].

"This language enables high-level and low-level coaching through coach instructions. High-level coaching includes changing tactics, formations used in each situation and changing player behavior. Low-level coaching includes defining formations, situations, player behavior and positioning with high detail. The language also enables the coach (functioning like an assistant coach) to send opponent modeling information and game statistical information to the players." [Rei02].

After the creation of this standard language, in 2003 several experts in the area, including Reis and Lau, developed the RoboCup Coach Standard Language (CLANG) [Che03]. It was developed to enable online coaches to change the behavior of simulated soccer players during games in the Simulated League, one of the competition types of RoboCup.

With a simple grammar to be followed, the language of concepts created in this work (PokerLANG) was based in Coach UniLANG and CLANG grammar.

3.2 PokerLANG

Poker is, in a professional level, a strategy card game. It requires a group of different tactics to the different moments of the game. A tactic, in poker, is used under certain conditions of the game. Those conditions are different concepts of the game like stack, opponents at the table, and your position at the table, among others. Using CLANG definitions, one of the purposes of this work was to build a language of poker concepts that would permit to create simple rules to define a poker agent. The language developed in this work is called PokerLANG.

3.2.1 Main Definition

The language definition starts in defining the top two settings to build (Strategy and Tactic). Strategy is composed by several tactics where each one is a consequence of the sum of one or more concepts of poker. The concepts main definition are defined using Bakus Naur form [Nau60], like:

<*STRATEGY*>::= {<*ACTIVATION_CONDITION*> <*TACTIC*>}

<*ACTIVATION_CONDITION*>::= {<*EVALUATOR*>}

<*TACTIC*>::= <*PREDEFINED_TACTIC*> | <*TACTIC_NAME*><*TACTIC_DEFINITION*>

<*PREDEFINED_TACTIC*>::= *loose_agressive* | *loose_passive* | *tight_agressive* | *tight_passive*

<*TACTIC_NAME*>::= *[string]*

<*TACTIC_DEFINITION*>::={<*BEHAVIOUR*> <*VALUE*>}

<*BEHAVIOUR*>::= {<*RULE*>}

<*RULE*>::= {<*EVALUATOR*> | <*PREDICTOR*>} <*ACTION*>

<*EVALUATOR*>::= <*NUMBER_OF_PLAYERS*> | <*STACK*> | <*POT_ODDS*> |
<*HAND_REGION*> | <*POSITION_AT_TABLE*>

<*PREDICTOR*>::= <*IMPLIED_ODDS*> | <*OPPONENT_HAND*> | <*OPPONENT_IN_GAME*> |
<*STEAL_BET*> | <*IMAGE_AT_TABLE*>

<*ACTION*>::= {<*PREDEFINED_ACTION*><*PERC*> | <*DEFINED_ACTION*><*PERC*>}

<*PREDEFINED_ACTION*>::= <*STEAL_THE_POT*> | <*SEMI_BLUFF*> |
<*CHECK_RAISE_BLUFF*> | <*SQUEEZE_PLAY*> |
<*CHECK_CALL_TRAP*> | <*CHECK_RAISE_TRAP*> |
<*POST_OAK_BLUFF*>

Fig 20: PokerLANG Main definition

Like is definition, the Strategy concept is made by a group of tactics, each one having an activate condition. This condition is composed by a group of evaluators that will be described later.

The tactics, as explained, are defined by a set of rules. These rules are the sum of poker concepts that leads to defined actions. The concepts used in the rules are basically the information you have at a poker table. But there are different kinds of information. In PokerLANG the information was separated in two types of information. The “certain” information (Evaluators) which is the information you can gather just by looking at a poker table, and the “uncertain” information (Predictors), information that is predicted by statistical data and is not fully reliable. The sum of these two types of information leads to a consequences (Actions) that are well-known poker moves used by professional poker players.

3.2.2 Evaluators

The Evaluators are basically the information a player can gather by just looking the surroundings. Each evaluator is a concept of poker that describes where you "stand" at a poker table:

```
<EVALUATOR>::= <NUMBER_OF_PLAYERS> | <STACK> | <POT_ODDS> |
                <HAND_STRENGTH> | <HAND_REGION> | <POSITION_AT_TABLE>
```

Fig 21: Evaluators Definition in PokerLANG

Number of Players

The first evaluator to be described is the number of players at the table. The size of the table is an important matter in poker because your actions efficiency depends on the number of players at the game.

```
<NUMBER_OF_PLAYERS>::=<PLAYER_VALUE>
```

Fig 22: Number of Players definition in PokerLANG

For example:

- The player has medium cards and he is looking to steal the pot (making a bet that forces everyone to fold)
- the probability of forcing everyone to fold will differ by the number of players in game (many opponents-low probability, few opponents-high probability)

Stack/M

The next evaluator is the Stack or M. The stack is the amount of chips that a player owns. In this definition, instead of using the exact number of chips (we can not determinate the accuracy of number of chips), it will be used the M formula:

$$M = \frac{Stack}{SB + BB + Antes} \quad \textbf{(Eq 5)}$$

The M represents the player situation in a table, depending on the value of the blinds and antes. But M can still be a problem because it can have large values, so to prevent this situation it's created a qualitative table with distinct regions of M. There are 5 predefined regions, but the user can create their own regions, defining the limits of each on of them.

```
<STACK>::= <PREDEFINED_STACK_REGION> | <STACK_REGION_DEFINITION>
  <PREDEFINED_STACK_REGION>::= green_zone | yellow_zone | orange_zone | red_zone | dead_zone
<STACK_REGION_DEFINITION>::= <STACK_REGION_NAME> <STACK_INTERVAL>
<STACK_REGION_NAME>::= [string]
<STACK_INTERVAL>::= <MIN_STACK> <COMP> <STACK_VALUE> <COMP> <MAX_STACK>
<MIN_STACK>::= <STACK_VALUE>
<MAX_STACK>::= <STACK_VALUE>
```

Fig 23: Stack definition in PokerLANG

An example of user defined regions:

Table 3: User defined Stack Regions

Name	Stack/M
DGreen Zone	M>40
Green Zone	40>M>20
Yellow Zone	20>M>10
Orange Zone	10>M>5
Red Zone	5>M>1
Dead Zone	M<1

Pot Odds

The Pot Odds is the main concept of poker for calling bets. It's calculated in a situation where you don't have the winning hand but it can improve, someone bets on you and you have to decide whether you call or fold. The Pot Odds is the comparison of the odds of your hand to improve against the odds the pot gives to you.

```
<POT_ODDS>::= <REAL>
```

Fig 24: Pot Odds definition in PokerLANG

To explain it here is a small example. Suppose a:

Player has 3000 chips and the opponent has 4250, the pot is currently 1200 chips. The player has the following hand:

Fig 25: Pot Odds Example – Player Cards

The flop comes with the following cards:

Fig 26: Pot Odds Example: - Flop Cards

The opponent bets 200 chips (probably has a King), should the player call or fold?

Currently the pot odds are 7-1 (the pot is 1400 chips and it costs you 200 to see another card 1400/200=7). What are the player hand odds? He does not have the best hand but it can improve to be the best hand with some cards. These cards are the outs the player has. To be able to decide it is necessary to calculate how many these outs are?

9 hearts for a flush:

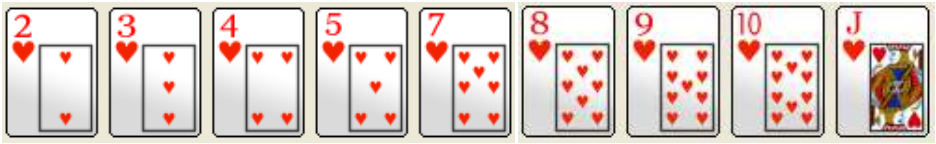

Fig 27: Pot Odds Example: - Cards for flush

3 aces for a pair:

Fig 28: Pot Odds Example: - Cards for pair

The player has 12 outs to improve your hand. There are 47 cards left (player has 2 cards plus 3 from the flop), so the hand odds are 4-1 (47/12=4). If the pot odds are bigger than the hand odds the player should call, if not he should fold. In this case 7>4 so the player should definitely call.

Hand Region

Some of the greatest moves in poker are made without true value in the hole cards, but the truth is that your game depends dramatically on the cards you own. This is the greatest concept in poker. It is true that the occasional bluff is a good weapon in a game but winning hands is what makes you go on top. In this definition, the possible combinations of cards that you can hold are separated in distinct regions. Each region represents a set of hands, grouped by value. There are 5 predefined regions, but the user can create their own regions, filling them with the hands that he chooses.

<HAND_REGION>::= <PREDEFINED_HAND_REGION> | <HAND_REGION_DEFINITION>

<PREDEFINED_HAND_REGION>::= a | b | c | d | e

<HAND_REGION_DEFINITION>::= <HAND_REGION_NAME> {<HAND>}

<HAND_REGION_NAME>::=[string]

Fig 29: Hand Region definition in PokerLANG

The predefined regions are based in Dan Harrington's groups [Har04]:

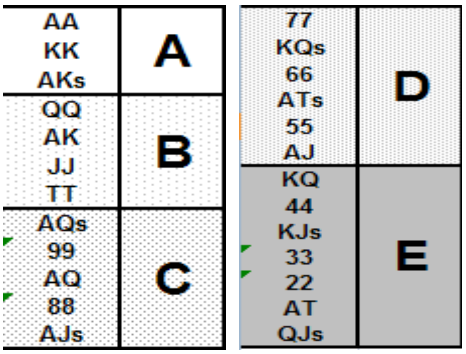

Fig 30: Dan Harrington's Groups

Position

Phil Gordon, one of the most respected poker players in the world, has said "You have to remind about 3 things in poker: position, position and position". When we talk about position, is where you stand at a table relatively to the current Small Blind and Big Blind. The order of bets is determined by this, so if your last to act you will have advantage because you will bet after all the players has made their moves (very important information). In this language definition, we will have a formula that gives us the quality of the player position. There are two special positions that will not be counted in the formula (Small Blind and Big Blind), therefore:

- Position will be counted from the big blind (in a table of 10 players the player on the left of the Big Blind will have the number 1 and the player on the right of the Small Blind will have the number 8).

- For each aggressive player in game or yet to act we will subtract 1 to the position.
- For each tight player in game with a raise we will subtract 1 as well.

Position quality = Position – (Number of aggressive players + Number of tight players)

The range of position quality depends on the number o players in the following proportion:

Range = [-(Number of players-2), (Number of players-2)] **(Eq 6)**

In a table with 10 players, for example, the range will be [-8, 8].

<POSITION_AT_TABLE>::= <PREDEFINED_POSITION_REGION> |

<POSITION_REGION_DEFINITION>

<PREDEFINED_POSITION_REGION>::= bad_position | normal_position | good_position

<POSITION_REGION_DEFINITION>::= <POSITION_REGION_NAME>{POSITION}

<POSITION_REGION_NAME>::= [string]

<POSITION>::= <MIN_POS> <COMP> <POS_VALUE> <COMP> <MAX_POS>

Fig 31: Position definition in PokerLANG

There are 3 predefined regions but the user can create their own and determine which positions belong to each region.

By definition, the position quality range for each region will be calculated has followed:

Min = -(Number of players-2)

Max = (Number of players-2)

TR (Total Range) = (Number of players-2)*2

- Bad Position [Min, Min+(TR/3)[
- Normal Position [Min+(TR/3), Max-(TR/3)]
- Good Position]Max-(TR/3), Max]

In a table of 10 players the predefined regions would be:

- Bad Position [-8, -3[
- Normal Position [-3, 3]
- Good Position]3, 8]

3.2.3 Predictors

The Predictors are concepts of poker that the player may estimate but can not be certain of them. There estimation is based on hidden information. So, the accuracy of the concepts depends on the ability of "reading" the information and what is yet to come:

<PREDICTOR>::= <IMPLIED_ODDS> | <OPPONENT_HAND> | <TYPE_OPPONENT > | <STEAL_BET> | <IMAGE_AT_TABLE>

Fig 32: Predictors definition in PokerLANG

Implied Odds

The Implied Odds are the evolution of the pot odds. Like the pot odds, it is a calculation that gives the player information to call a bet or not. But unlike the pot odds, implied odds are based on predicting the cards that will come in a hand.

<IMPLIED_ODDS>::= <REAL>

Fig 33: Implied Odds definition in PokerLANG

To understand the concept here is a simple example:

Suppose the player has 3000 chips, the opponent has 4250, the pot is currently 1200 chips. The player in analysis has the following hand:

Fig 34: Implied Odds Example – Player Cards

The flop comes with the following cards:

Fig 35: Implied Odds Example – Flop Cards

The opponent bets 200 chips (probably has a King), should the player call or fold?

This is the same example of the pot odds, but instead we are going to calculate the implied odds. If the player hits another heart, he will have the best game (flush), but he may not win any more money because the third flush card will be visible on the board. Suppose the player believes that the opponent will fold against a large bet but will call

against a smaller bet like 200 chips (equal to his bet on the flop). So instead of calling to a pot of 1400 chips, the player is going to call to a bigger pot:

1600 chips = 1200 (the pot) + 200 (his bet on the flop) + 200 (future call by your opponent)

The implied odds are 8-1 (1600/200), bigger than the pot odds (7-1). Of course this is always based on the player prediction that he will hit a flush draw and the opponent will call the bets (not certain information).

Opponent Hand

Another concept that a player may try to predict is the opponent's hand. This implies that he gathers information about the opponent along the way:

- What type of hands does he play?
- In which situations? (Position, size of the stack,…)
- What are his actions with the type of hands?

All this information combined gives the player a perspective of what hand the opponent has at the moment.

```
<OPPONENT_HAND>::=<HAND>
```

Fig 36: Opponent Hand definition in PokerLANG

Type of Opponent

Like the previous concept, the type of opponent that the player is facing can be determined by gathering information:

- How regularly your opponent plays a hand?
- How regularly your opponent bets?

In this definition, there are 4 types of opponent predefined:

```
<TYPE_OF_PLAYER>::= loose_agressive | loose_passive | tight_agressive | tight_passive
```

Fig 37: Types of Player in PokerLANG

Each opponent, depending on the information gathered, is placed in one of these 4 categories, but the amount of information needed to place them can be determined by the user.

Steal Bet

The steal bet is the amount of chips a given player needs to get the pot with no hand at all. It, obviously, depends on the type of opponents that the player is facing and the image he has at the table. This concept is based on gathering information of the drops of the opponents facing bets. Of course the accuracy depends dramatically on if the opponents have any kind of a hand.

```
<STEAL_BET>::= <BET_VALUE>
```

Fig 38: Steal Bet definition in PokerLANG

Image at Table

The player's image at the table is an important factor for the actions taken. The type of player that the opponents see in the player affects the success of the actions taken. If the player is seen as a tight player, the bluffs will have high probability to succeed. In the opposite side, if the player's image is of a loose player the attempts to trap the opponents will be successful most of the times. This concept is defined in the same way that the type of opponents faced but using the information that other players can see of the player.

```
<IMAGE_AT_TABLE>::= <TYPE_OF_PLAYER>
```

Fig 39: Image at Table definition in PokerLANG

3.2.4 Actions

There are several poker playing moves that may be used in a game. This poker playing moves are specific ways of handling a hand to achieve a goal. In this definition, the user can choose well known poker plays or he can create their own:

```
<ACTION>::= {<PREDEFINED_ACTION><PERC> | <DEFINED_ACTION><PERC>}
<PREDEFINED_ACTION>::= <STEAL_THE_POT> | <SEMI_BLUFF> |
                       <CHECK_RAISE_BLUFF> | <SQUEEZE_PLAY> |
                       <CHECK_CALL_TRAP> | <CHECK_RAISE_TRAP> |
                       <POST_OAK_BLUFF>
```

Fig 40: Actions definition in PokerLANG

Defined Action

Despite the several poker plays given by this definition, the user can still define his actions and determine what to do in each phase of the game (pre-flop, flop, turn and river):

```
<DEFINED_ACTION>::= <ACTION_NAME>{<PRE_FLOP_ACTION> | <FLOP_ACTION> |
                                  <TURN_ACTION> | <RIVER_ACTION>}
<PRE_FLOP_ACTION>::= {<BET_VALUE><PERC>}
<FLOP_ACTION>::= {<BET_VALUE><PERC>}
<TURN_ACTION>::={<BET_VALUE><PERC>}
<RIVER_ACTION>::={<BET_VALUE><PERC>}
```

Fig 41: Defined Action definition in PokerLANG

Steal the Pot

Stealing the pot is a poker play regularly used by all players. It's making a large bet with practically nothing to force all the players to fold. It requires a good position at the table and a certain type of players (it's more effective against tight players). In this definition, this action depends directly from the predictor "steal bet":

```
<STEAL_THE_POT>::= <STEAL_BET>
```

Fig 42: Steal the Pot definition in PokerLANG

Semi Bluff

A semi bluff is similar to the "steal the pot" play. The difference is that stealing the pot is a play you use when you don't have any hand at all and you don't have great possibilities of having the best hand. In the semi bluff you still don't have a good hand but your probabilities of having the best hand are big.

```
<SEMI_BLUFF>::= {<BET_VALUE><PERC>}
```

Fig 43: Semi Bluff definition in PokerLANG

For better understanding of this move here is an example:

A given player has 3000 chips, the opponent has 4250, the pot is currently 1200 chips. The player has the following hand:

Fig 44: Semi Bluff Example – Player Cards

The flop comes with the following cards:

Fig 45: Semi Bluff Example – Flop Cards

The opponent checks and the player bets 300 chips. This is a semi bluff. The player probably does not have the best hand (this hand at the moment is only an ace high) but have good chances of hitting a flush which makes the player have the "nuts" hand. So he is betting with two objectives:

- He wants to win the pot right now because in fact he does not have a good hand
- If the opponents call, the player may still not have a hand but may hit a nuts hand which will give him a bigger pot

Check-Raise Bluff

This is an outrageous move made by the top poker players. A Check/Raise bluff is a move used when the player does not have any hand but he feels the opponents are weak. Basically, the player raises an early bet by an opponent to make the opponents wonder if he was trapping them when he checks at his turn and made a powerful move now. It requires courage to do a move like this because the player is putting his money stack at stake in a poor hand. In this definition the player determines a list of size bets to make with the probability of making it.

```
<CHECK_RAISE_BLUFF>::= {<BET_VALUE><PERC>}
```

Fig 46: Check-Raise Bluff definition in PokerLANG

Squeeze Play

The Squeeze Play is an advanced and elegant bluff. The player makes a large re-raise after two players have already entered the pot, the first with a raise and the second with a call. This is what affects both players at their position:

- The first opponent (the raiser) has the pot odds to face the player, but he doesn't know what the second player will do (he maybe trapping and will

make the final move now) so he folds because it is a hand that is now out of his control.

- The second opponent had enough equity to play against one raiser, but not against a re-raiser, so he also folds and the player takes the pot

To do a move like this the player must have a very tight image at the table. If he is seen as a bluffer it will be very difficult for a move like this to succeed.

```
<SQUEEZE_PLAY>::= {<BET_VALUE><PERC>}
```

Fig 47: Squeeze Play definition in PokerLANG

Check-Call Trap

Trapping is one of the key moves in poker. There are different forms of trapping but all of them have one objective: take the most possible money of your opponents. The Check-Call Trap is one of the ways to get it. When a player has the nuts or a very good hand he may give all the action to the adversaries and just call all the way.

```
<CHECK_CALL_TRAP>::= check | call
```

Fig 48: Check-Call Trap definition in PokerLANG

Check-Raise Trap

The Check-Raise Trap is another form of trapping and requires certain type of opponents. The player image at the table is also very important (check/raise trap has great efficiency if you are seen has a regular bluffer). The player waits for action from the opponents and when that happens, he "fires" against them with a raise. It pushes more chips to the pot but it can also end the hand fast.

```
<CHECK_RAISE_TRAP>::= check | <RAISE>
```

Fig 49: Check-Raise Trap definition in PokerLANG

Post Oak Bluff

The "Post Oak Bluff" is Doyle Brunson's term for a particularly sneaky sort of play that a player can make on the turn or the river [Doy78]. This play works in both ways (bluff and trap) and requires a regularly use so that the opponents are expecting it from both ways. The first times a player uses this sort of play should be attempts to trap. Imagine

the player has the nuts in the turn or the river. He knows he is going to win the hand and that the opponents know he has the best hand. Instead of betting large, he makes a small bet in order to give great pot odds to the adversaries to call. If he does this regularly, when he tries to bluff with a play like this, he will also be more successful because the opponent will think he is trying to lure him into the pot and will go away.

```
<POST_OAK_BLUFF>::= {<BET_VALUE><PERC>}
```

Fig 50: Post Oak Bluff definition in PokerLANG

3.2.5 Basic Concepts-Values

To complete the language, some basic concepts are defined for better understanding:

```
<HAND>::= <CARD> <CARD>

<CARD>::= <CARD_VALUE> <SUIT>

<CARD_VALUE>::= ace | king | queen | jack | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2

<SUIT>::= spades | clubs | hearts | diamonds

<BET_VALUE>::= <INTEGER>

<STACK_VALUE>::= <INTEGER>

<COMP>::= < | > | <= | >=

<RAISE>::= <BET_VALUE>
```

Fig 51: Basic Concepts-Values definition in PokerLANG

3.2.6 Example of a Poker Strategy in PokerLANG

After the detailed explanation of each defined concept of the language, figure 3.33 contains an example of a simple strategy with two tactics, each one with two rules, named “Winner”:

```
Winner(green_zone,tactic1(((A,bad_position),check),((good_position,loose_agressiv
e),raise)),tactic2(((E,yellow_zone),fold),((green_zone,79%,tight_passive),raise))))
```

Fig 52: Example of a Strategy in Poker Lang

3.3 Conclusions

From our knowledge, this is the first attempt to create a high-level language of concepts for a poker playing agent. The concepts described by the language are the most common in the poker world. However, they are high-level concepts, difficult to create without domain expertise, but essential in order to fully capture the complexity of the game. Of course, more concepts could be added which would bring a stronger definition to the language, but its present definition seems enough to describe a professional poker playing strategy.

A critic that could be made is that the language is too high-level, but one of the purposes of creating the language is that it can be accessible to the common user, that is a poker player, that is able to describe poker strategies exactly at a high-level of abstraction. This is a language that supports the main application of this work, Poker Builder, an application for building poker strategies. It makes easier to create the application supporting its features with the language. The creation of the strategies will be simplified and the structure to support these strategies will become easy to implement. Even if in the construction of the application the format of the strategies is not the same as in the language, it creates a secure foundation for Poker Builder.

Chapter 4

4. Poker Builder

After defining the high-level language, the next phase of this project is to build a simple interface so the user may effectively use the language to build poker strategies. Poker Builder is a Flex [Fle08] application that allows the user to create rules of concepts using the language previously introduced, and set the behavior of a poker agent. With a smooth interface and simple features, Poker Builder is accessible to any user that understands the main concepts of poker. One of the purposes of this work was to make a very practical application, even usable to users only familiarized with the most basic computer usage.

4.1 Flex

Flex is a free, open source framework created by Adobe [Ado08] for building highly interactive, expressive web applications that deploy consistently on all major browsers, desktops, and operating systems. It provides a modern, standards-based language and programming model that supports common design patterns. MXML [Coe03], a declarative XML-based language, is used to describe User Interface (UI) layout and behaviors, and ActionScript 3, a powerful object-oriented programming language, is used to create client logic. Flex also includes a rich component library with more than 100 proven, extensible UI components for creating Rich Internet Applications (RIAs), as well as an interactive Flex application debugger.

RIAs created with Flex can run in the browser using Adobe Flash® Player software or on the desktop on Adobe AIR™, the cross-operating system runtime. This enables Flex applications to run consistently across all major browsers and on the desktop. And using AIR, Flex applications can now access local data and system resources on the desktop.

It is an easy learning framework, with a pleasant layout for the users.

4.2 Main Classes

For the implementation of the language of concepts, described in Chapter 3, Poker Builder is divided in four major classes: Strategy, Tactic, Rule and Property. The

interface begins with an instance of the Strategy Class that creates instances of all other classes depending on what the user is creating.

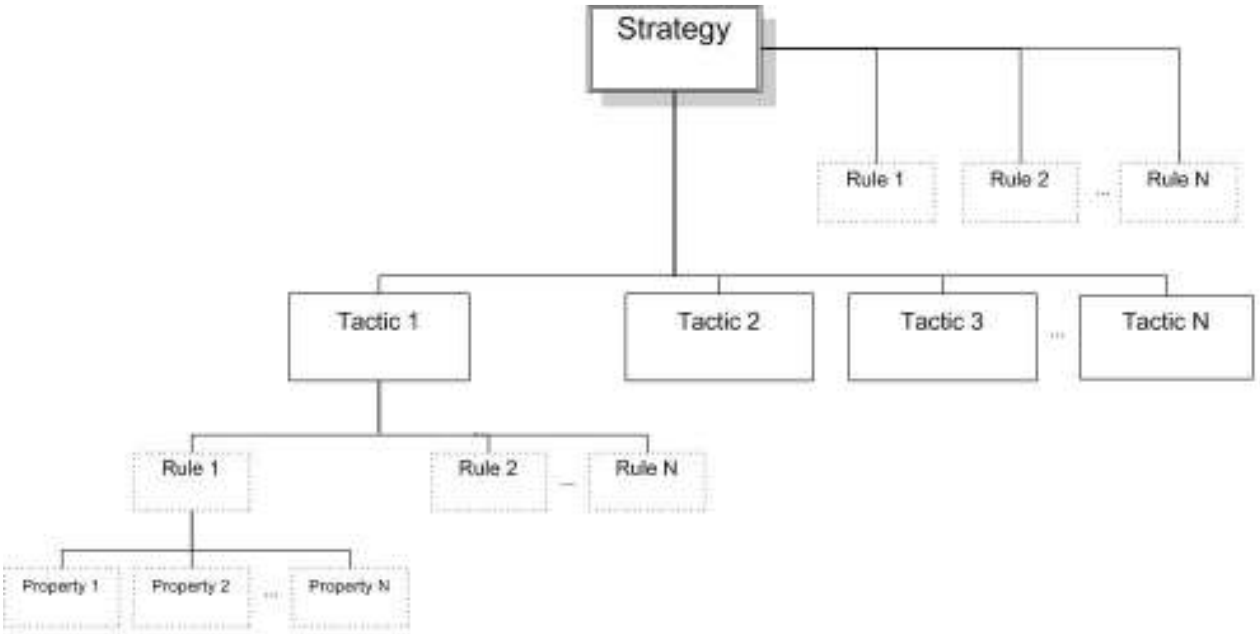

Fig 53: Main Classes Diagram

4.2.1 Strategy Class

A new instance of this class is created each time the application runes. The class is defined by these properties:

Table 4: Strategy Class variables

Variable Name	Type	Description
nameStrategy	String	Name of the Strategy
numTactics	Int	Number of Tactics defined in the Strategy
numRules	Int	Number of Rules defined in the Strategy
tactics	Array	Array of the Objects of type Tactic defined in the Strategy
rules	Array	Array of the Objects of type Rule defined in the Strategy
associatedObject	DisplayObject	Flex Object associated to the Strategy

The Array "rules" contains several instances of the Rule Class. This array represents the rules, created by the user, to define the circumstances of which tactic the poker agent will use. Each of the rules contains has a consequence one of the tactics of the tactics

Array. The associatedObject variable is to define the Flex Object associated to this instance of the Strategy Class, in this case the TabNavigator Object.

The Strategy Class also contains several methods, easy to understand by their names:

Table 5: Strategy Class methods

Method Name	Arguments	Return	Description
addTactic()	Void	Void	To add a new Tactic to Strategy
addRule()	Void	Void	To add a new Rule to Strategy
getNumTactics()	Void	Int	Get the number of tactics that Strategy has
getNumRules()	Void	Int	Get the number of rules that Strategy has
getNameStrategy()	Void	String	Get the Name of the Strategy
setNameStrategy()	String	Void	Set the Name of the Strategy
getAssociatedObject()	Void	DisplayObject	Get the Flex Object associated to the Strategy
setAssociatedObject()	DisplayObject	Void	Set the Flex Object associated to the Strategy
getRule()	Int	Rule	Get a certain Rule by it's index
getTactic()	Int	Tactic	Get a certain Tactic by it's index
getLastTactic()	Void	Tactic	Get Last Tactic of the Tactics Array
getLastRule()	Void	Rule	Get Last Rule of the Rules Array
removeRule()	Int	Void	Remove a certain Rule by it's index
removeAllRules()	Void	Void	Remove all rules from Strategy

As the top class below the interface, Strategy Class contains instances of Rule Class and Tactic Class. The distinction between Strategy and Tactic changes the way that the file saving works, which will be explain later with more detail.

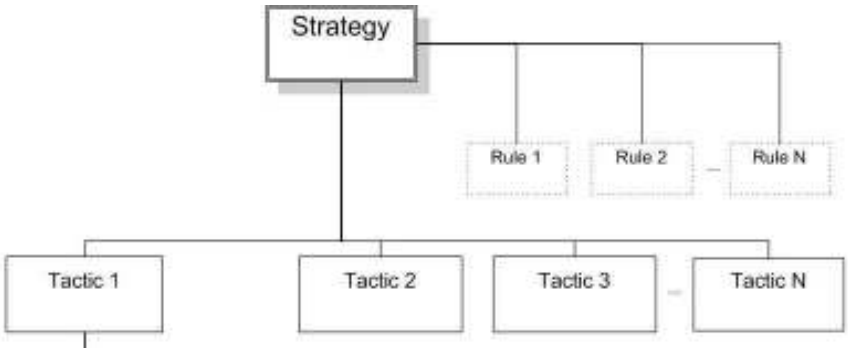

Fig 54: Strategy Class Diagram

4.2.2 Tactic Class

The definition of the Tactic Class is very similar to the Strategy Class. A new instance of the class is created whenever the user wants, by choosing the "New Tactic" option of the top menu. The properties that define this class are:

Table 6: Tactic Class variables

Variable Name	Type	Description
nameTactic	String	Name of the Tactic
numRules	Int	Number of Rules defined in the Tactic
rules	Array	Array of the Objects of type Rule defined in the Tactic
associatedObject	DisplayObject	Flex Object associated to the Tactic

As in the Strategy Class, the "rules" Array contains instances of the Rule Class but the consequences of the rules created by the user are not tactics but the poker moves that are available to the user to choose. The methods that the class provides are:

Table 7: Tactic Class methods

Method Name	Arguments	Return	Description
addRule()	Void	Void	To add a new Rule to Tactic
getNumRules()	Void	Int	Get the number of rules that Tactic has
getNameTactic()	Void	String	Get the Name of the Tactic
setNameTactic()	String	Void	Set the Name of the Tactic
getAssociatedObject()	Void	DisplayObject	Get the Flex Object associated

			to the Tactic
setAssociatedObject()	DisplayObject	Void	Set the Flex Object associated to the Tactic
getRule()	Int	Rule	Get a certain Rule by it's index
getLastRule()	Void	Rule	Get Last Rule of the Rules Array
removeRule()	Int	Void	Remove a certain Rule by it's index
removeAllRules()	Void	Void	Remove all rules from Tactic

With the distinction of Strategy and Tactic, the user can save the Strategy as a hole or he can save the tactics in separate. This allows the user to load already made tactics, to include them in creating a new Strategy.

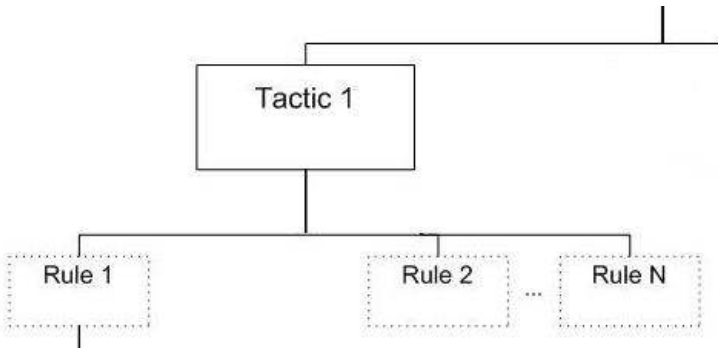

Fig 55: Tactic Class Diagram

4.2.3 Rule Class

The Rule Class is the "translation" of the Rule concept in PokerLANG. When a new Tactic is created, in Poker Builder the "new Rule" option becomes accessible. When it's chose, a new instance of this class is created, with the following properties:

Table 8: Rule Class variables

Variable Name	Type	Description
nameRule	String	Name of the Rule
numProperties	Int	Number of Properties defined in the Rule
properties	Array	Array of the Objects of type Property defined in the Rule
associatedObject	DisplayObject	Flex Object associated to the Rule

hasEqual	Int	Number of consequences of the Rule
validateProperties	Array	Array of validation for the Rule's Porperties

The "propertiesArray" contains several instances of the Property Class, each one represents one of the available poker concepts (evaluators, predictors or actions). This class also has a Array to validate the properties. It exists to avoid the duplication of type of properties in the same rule. For example in a rule it is not logic to have two "Stack" properties defined. For the purpose of validations there is still the "hasEqual" variable that defines the number of actions associated to this rule. For these validations and for the construction of a rule, the class gives several methods to use:

Table 9: Rule Class methods

Method Name	Arguments	Return	Description
addProperty()	Void	Void	To add a new Property to Rule
getNumProperties()	Void	Int	Get the number of properties that Rule has
getNameRule()	Void	String	Get the Name of the Rule
setNameRule()	String	Void	Set the Name of the Rule
getAssociatedObject()	Void	DisplayObject	Get the Flex Object associated to the Rule
setAssociatedObject()	DisplayObject	Void	Set the Flex Object associated to the Rule
getProperty()	Int	Rule	Get a certain Property by it's index
getLastProperty()	Void	Rule	Get Last Property of the properties Array
removeProperty()	int	Void	Remove a certain Property by it's index
removeAllProperties()	Void	Void	Remove all properties from Rule
getHasEqual()	void	Int	Get hasEqual value
setHasEqual()	Int	void	Set hasEqual value
existsProperty()	Property	Boolean	Verify if a Property exists in Rule

validateProperty()	Property	Boolean	Verify if a Property is validated in Rule
removeValProperty()	Property	Void	Remove validation from a Property
addValProperty()	Property	void	Add validation to a Property

The Rule Class is both used by Strategy Class and Tactic Class. There are some validations that this class does in the construction, like:

- Cannot have duplicate evaluators or predictors (it is not logical)
- Can add several actions to the same rule but not duplicated actions
- Defined action is an exception to the previous validation

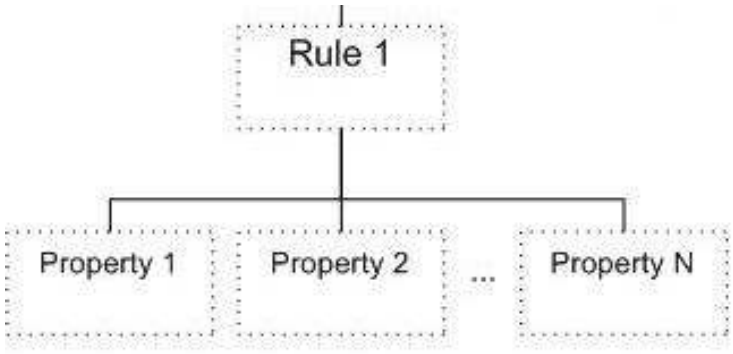

Fig 56: Rule Class Diagram

4.2.4 Property Class

The Property Class is the lowest in the hierarchy. It represents the basic concepts of PokerLANG (evaluators, predictors and actions). The properties that define the class are used depending on which concept the instance is representing:

Table 10: Property Class variables

Variable Name	Type	Description
nameProperty	String	Name of the Property
stackArray	Array	Data Array for concept Stack
positionArray	Array	Data Array for concept Position
handArray	Array	Data Array for concept Hand

typePlayerArray	Array	Data Array for concept Type of Player
minValue	Int	Minimum Value for quantitative concepts
maxValue	Int	Minimum Value for quantitative concepts
nominalValue	Int	Nominal Value for Qualitative concepts
varianceValue	Number	Variance Value for quantitative concepts
betValue	Number	Bet Value for quantitative concepts
nominalValueArray	Array	Data Array for nominal values of Action concepts
varianceValueArray	Array	Data Array for variance values of Action concepts
betValueArray	Array	Data Array for bet values of Action concepts
associatedObject	DisplayObject	Flex Object associated to the Property

The different types that the Property Class can represent are distinguished by their name. Then depending on their type, the respective array or values are used. Which one each concepts uses will be explained further. The class provides several methods:

Table 11: Property Class methods

Method Name	Arguments	Return	Description
getNameProperty()	Void	String	Get the Name of the Property
setNameProperty()	String	Void	Set the Name of the Property
getAssociatedObject()	Void	DisplayObject	Get the Flex Object associated to the Property
setAssociatedObject()	DisplayObject	Void	Set the Flex Object associated to the Property
getStackArray()	Void	Array	Get the Stack concept Array of the Property
setStackArray()	Array	Void	Set the Stack concept Array of the Property
getPositionArray()	Void	Array	Get the Position concept Array of the Property
setPositionArray()	Array	Void	Set the Position concept Array of the Property
getHandArray()	Void	Array	Get the Hand concept Array

			of the Property
setHandArray()	Array	Void	Set the Hand concept Array of the Property
getTypePlayerArray()	Void	Array	Get the Type of Player concept Array of the Property
setTypePlayerArray()	Array	Void	Set the Type of Player concept Array of the Property
getNominalValueArray()	Void	Array	Get the Action concept nominal value Array of the Property
setNominalValueArray()	Array	Void	Set the Action concept nominal value Array of the Property
getVarianceValueArray()	Void	Array	Get the Action concept variance value Array of the Property
setVarianceValueArray()	Array	Void	Set the Action concept variance value Array of the Property
getBetValueArray()	Void	Array	Get the Action concept bet value Array of the Property
setBetValueArray()	Array	Void	Set the Action concept bet value Array of the Property
getMinValue()	Void	int	Get the minimum value for quantitative concepts of the Property
setMinValue()	Int	void	Set the minimum value for quantitative concepts of the Property
getMaxValue()	Void	Int	Get the maximum value for quantitative concepts of the Property
setMaxValue()	Int	Void	Set the maximum value for quantitative concepts of the Property

getBetValue()	Void	Number	Get the bet value for quantitative concepts of the Property
setBetValue()	Number	void	Set the bet value for quantitative concepts of the Property
getNominalValue()	Void	int	Get the nominal value for quantitative concepts of the Property
setNominalValue()	Int	void	Set the nominal value for quantitative concepts of the Property
getVarianceValue()	Void	Number	Get the variance value for quantitative concepts of the Property
setVarianceValue()	Number	void	Set the variance value for quantitative concepts of the Property

The values that each instance of the class has are mostly defined by the user. Some of these properties are pre-defined (most actions) but the user can change their values. These definitions will be explained in more detail next.

4.3 Concepts Pre-Defined

In Poker Builder the user can create rules of poker concepts and define their values. To make it easier to understand, most concepts have pre-defined values. The pre-defined concepts have straight relation with the possible values of the Property Class. Each concept was already explained in detail on Chapter 3. Next, it will be explained the pre-defined values of the concepts, which variables they are related of the Property Class and the interface that supports them.

4.3.1 Evaluators

As explained before, evaluators are the perfect information that can be gathered by just looking at a poker table. To simplify the creation of the poker agent, Poker Builder has pre-defined values for each Evaluator.

Stack/M

The Stack\M concept is divided in five different groups:

- Green Zone (GZ);
- Yellow Zone (YZ);
- Orange Zone (OZ);
- Red Zone (RZ);
- Death Zone (DZ).

The value of this concept is a nominal value and the user may choose one or more groups.

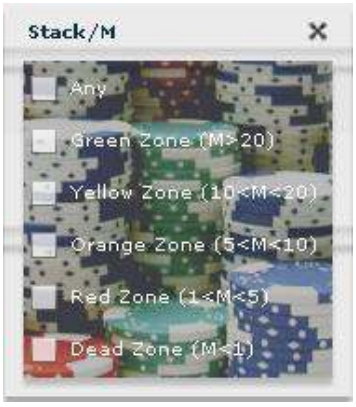

Fig 57: Stack Interface in Poker Builder

In the Property Class, the Stack\M is saved on a five position array, the stackArray variable. Each position gets a value of 0 or 1 that represents which group the user selects. For example, if the user selects the GZ and the YZ the array will be [1,1,0,0,0].

Number of Players

The next concept is the Number of Players that are in game (haven't fold the hand). The user defines a minimum and a maximum number to create an interval.

Fig 58: Number of Players Interface in Poker Builder

The Number of Players is saved in the Property Class, using the minValue and maxValue variables.

Pot Odds

In the Pot Odds concept the user defines a nominal value, given by three choices:

- Equal;
- Greater;
- Greater or Equal.

As explained in Chapter 3, the Pot Odds are related to the Hand Odds and a direct comparison between them is what the user chooses. One problem that occurred in this relation is that Pot Odds and Hand Odds are decimal values and it's very rare to have the same Odds in both ways, so an option given is that the user can define a variance value to approximate the Odds. If the Pot Odds were 2,2 and the Hand Odds 2.12, if the user gives a variance of 0.2, these Odds would be consider equal because the Pot Odds would be now a value between [2.2-0.2,2.2+0.2].

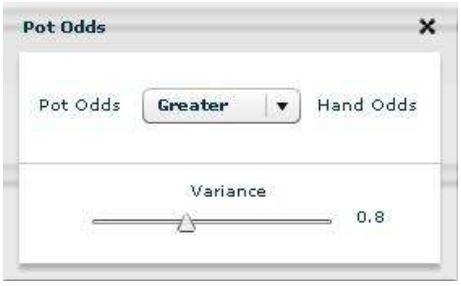

Fig 59: Pot Odds Interface in Poker Builder

In the Property Class, to save this concept it is used the nominalValue and the varianceValue variables.

Hand Region

The Hand concept is similar to the Stack\M concept. The possible hands that a player can hold are divider into five groups, based on Dan Harrington's groups [Dan04].

Fig 60: Hand Region Interface in Poker Builder

To save it, it is used the handArray variable of the Property Class in the same way that Stack\M is used.

Position

Finally, the Position concept is a nominal value. The user can choose between three options:

- Bad Position;
- Normal Position;
- Good Position.

The definition of bad, normal and good position is based on the formula explained in Chapter 3.

Fig 61: Position Interface in Poker Builder

In this case, the nominalValue variable is used, to save the concept in the Property Class.

4.3.2 Predictors

Predictors are the imperfect information available in a poker table. They are based on statistical knowledge and thus are unreliable. The way that predictors are delineated will be explained in Chapter 5. Basically uses the statistical information gathered and with certain formulas it makes a guess of what are the concepts values.

Implied Odds

The Implied Odds concept works in the same way of the Pot Odds concept explained previously. The user defines a nominal value with three options to choose:

- Equal;
- Greater;
- Greater or Equal.

The Implied Odds are also related to the Hand Odds and like in the Pot Odds, Implied Odds are decimal values but the user has at his disposal the chance to define a variance value.

In the Property Class, to save the Implied Odds concept we use the nominalValue and the varianceValue variables just like in Pot Odds.

Opponent Hand

Another predictor that works in the same way as a evaluator, is the Opponent Hand concept. As in the Hand concept, in this predictor the user can also choose from five groups of hands, the Dan Harrington's groups [Dan04].

To save it, like in the Hand evaluator, the Opponent Hand uses the handArray variable of the Property Class.

Type of Opponent

The predictors Type of Opponent and Image at Table work on the same values. Despite the fact that they mean different things, the choices for the user are the same. There are four types of player to choose:

- Loose-Aggressive;
- Tight-Aggressive;
- Loose-Passive;
- Tight-Passive.

For both concepts the typePlayerArray variable is used to label them in the Property Class.

Fig 62: Type of Opponent/Image At Table Interface in Poker Builder

Steal Bet

At last, the predictor Steal Bet is delineated by a decimal value that represents fractions of the player Stack. The user delimits the maximum value to steal. If the steal bet needed at the table is less that the one defined by the user than this predictor is satisfied.

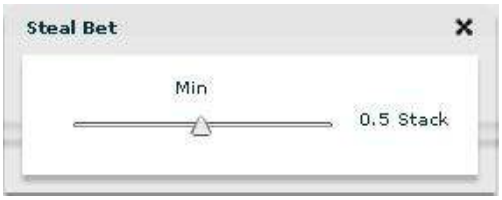

Fig 63: Steal Bet Interface in Poker Builder

Steal Bet is saved with the betValue variable of the Property Class.

4.3.3 Actions

The actions are the concepts of poker moves as it was explained in Chapter 3. Each action is divided in the four stages of a poker hand, Pre-Flop, Flop, Turn and River. In these four stages, the user is presented with three decisions to make: the action type, the bet value and the variance.

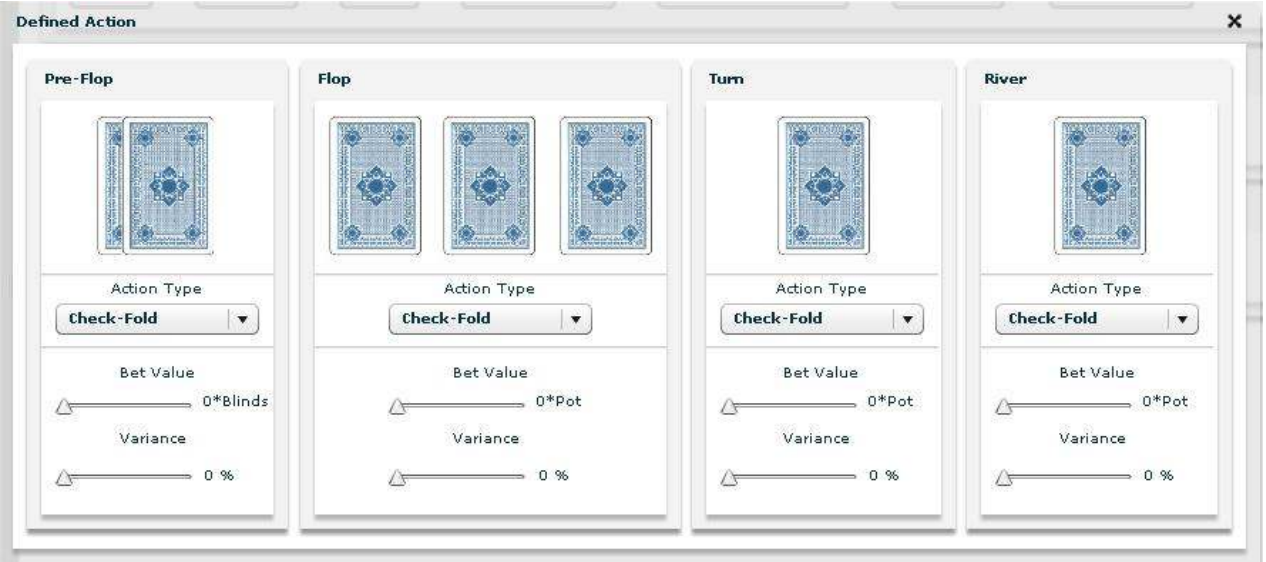

Fig 64: Action Interface in Poker Builder

The action type is a nominal value with several sets of actions:

- Check-Fold;
- Check-Call;
- Check-Raise;
- Raise-Fold;
- Raise-Call;
- Raise-Raise.

These represents the behavior the poker agent will have in terms of betting.

Then the user defines the maximum bet value is willing to call or raise and the respective variance.

In the Defined Action concept, the user can delineate all the values in every stage, but in the pre-defined actions, the values cannot be changed because they represent specific poker moves. The user can access them and see the defined values.

4.4 Application Views

Poker Builder gives the user two different views to create rules (Strategy View and Tactic View). A strategy is a group of tactics that represent different behaviors depending on the circumstances of the game. For the possibility given to use different tactics in a game, it was mandatory to distinguish this two phases.

4.4.1 Strategy View

The Strategy View allows the user to create rules of decision in what tactic the poker agent should use on the defined circumstances. It is the most high-leveled definition that

the user can do of the language of concepts (PokerLANG). The main distinction from the Tactic View is that despite the fact that the evaluators and predictors are the same, the actions are not. Instead of the list of poker moves that are available to the user, the Strategy View presents the list of tactics already defined by the user. Here's an example of the interface of the Strategy View:

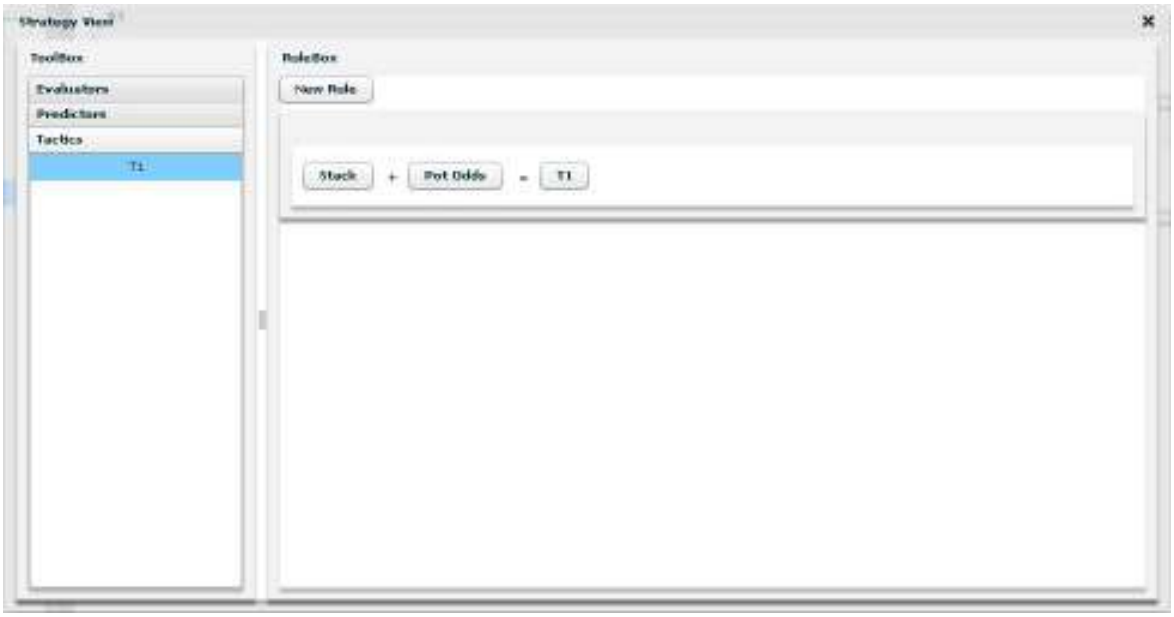

Fig 65: Strategy View Interface in Poker Builder

4.4.2 Tactic View

The Tactic View is the main view of Poker Builder. It is presented when the program starts and is where the user defines the lower specifications of the poker agent. It is presented with a list of the evaluators, the predictors and some of the famous poker moves that professional players use in their game (actions).

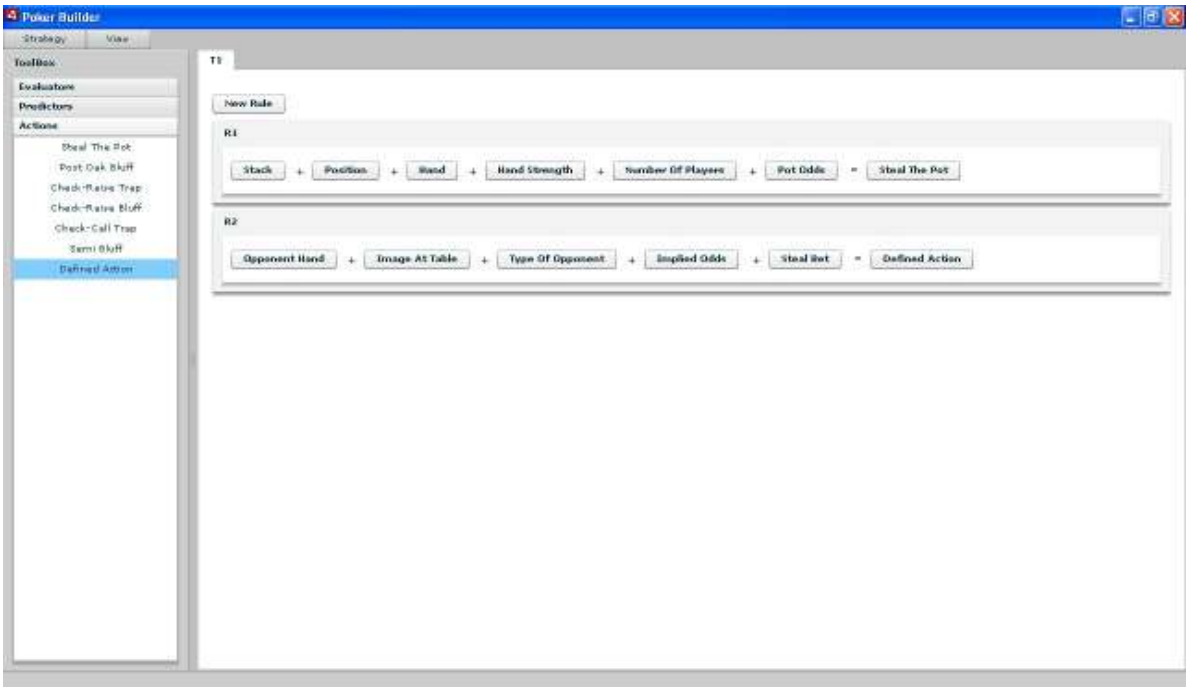

Fig 66: Tactic View Interface in Poker Builder

The menus are only available in this view, which includes the possibility of saving and loading strategies or tactics.

4.5 Files Structure

In Poker Builder, the user has the option of saving his strategies and tactics. When saving, the application creates a text file and saves the strategy or tactic properties in a defined text format based on the language of concepts explained in Chapter 3.

As previously mentioned, it was made the decision of separating Strategy from Tactic, in order to permit the user to load created tactics from different strategies. So, when a strategy is saved, several files are created: one for the strategy top rules and another file for each tactic that the strategy owns.

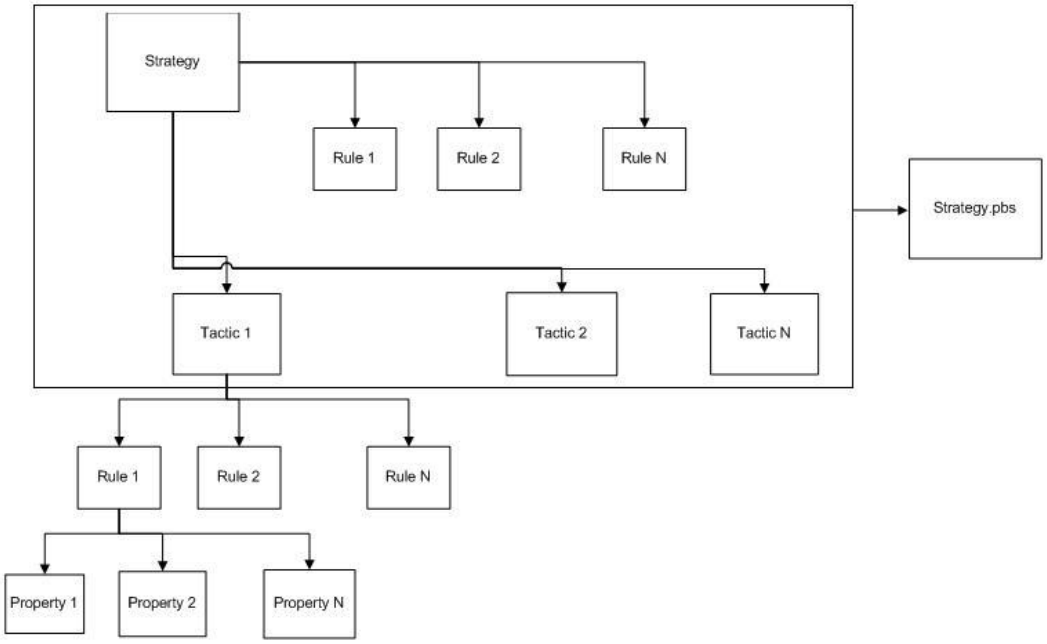

Fig 67: Saving Strategy File

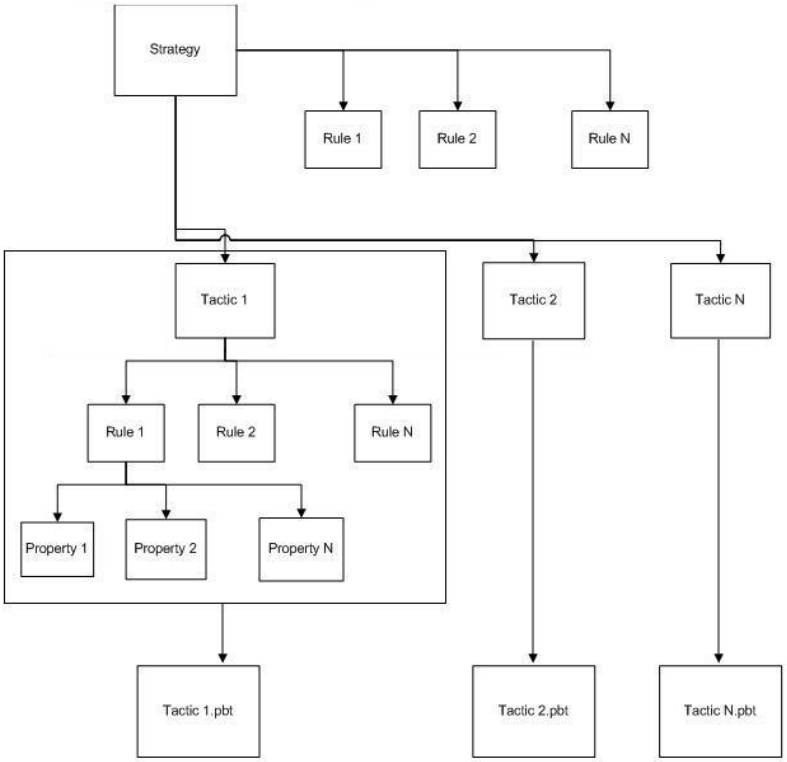

Fig 68: Saving Tactic Files

The decision to make the saving into text files was made thinking on the poker agent that will be explained in Chapter 5. It would make the interpretation for the agent easier and the parser created in Poker Builder would be practically the same in the poker agent.

4.5.1 Strategy File

The text file to save strategies is a file with the extension *.pbs. The text format is made as follow:

- The Strategy Name followed by "[" that symbolizes the beginning of the strategy properties.
- The Names of the Tactics that the Strategy owns separate by "|" (these names are also the names of the tactic files).
- When all the names are placed then "?" separates the tactic names from the strategy rules.
- Each rule starts with "(" followed by the properties that define the rule.
- Each Property is separated by ",".
- The Property values (name and specific values) are separated by ";".
- To separate the rules in the end of the rule a "!" is placed.
- At the end, the strategy is "closed" by a "]".

Here's an example of the final text of a strategy file:

```
Winner[Tactic1{}|Tactic2{}?(Stack;0;0;0;0;1,=,Tactic 1)!(Hand;1;1;1;1;1,=,Tactic 2)]
```

Fig 69: Example of a Strategy File

4.5.2 Tactic File

Tactic files have the *.pbt extension. The text format is very similar to the strategy format, made by these rules:

- The Tactic Name followed by "{" that symbolizes the beginning of the tactic properties.
- Then it is written each Rule that composes the Tactic separated by "|"
  - It begins with the Rule Name followed by "(".
  - Each Property of the rule is divided by a "," and the properties than define it separated by ";".
- Each rule ends with a ")".
- The tactic is "closed" with a "}".

Figure 4.18 shows an example of a tactic file:

```
Tactic1{
        Rule1(Stack;0;0;0;1;1,+,
                Hand;0;1;0;0;0,=,
                Check-Raise Trap;1;1;1;1;0;0;0;0;0;0;0;0)
                !
        Rule2(Hand Strength;79;100,+,
                Image At Table;1;1;0;0,=,
                Semi Bluff;0;4;4;0;0;0.5;0.5;0;0;20;20;0)
}
```

Fig 70: Example of a Tactic File

In the tactic showed in Figure 4.18, only two rules are defined:

- Rule1 is activated when the player's stack is in the lower stack zones (Red Zone and Dead Zone) and he holds a group B hand. The action would be the Check-Raise Trap move.
- Rule2 defines the Semi Bluff action if the player holds a hand in the top 21% hands and if he is seen at the table as a loose player.

4.6 Conclusions

Using the high-level language of concepts made the implementation of the application easier because it saved a lot of time in defining structures and the connections between the main classes. Despite the fact that the specifications used to save the strategies were a little bit different of PokerLANG definitions, it gave a full perspective of what had to be done.

With a simple and intuitive interface, Poker Builder became a very pleasant application for the human eye and a very useful one for creating complete poker strategies.

To enable to fully test the capabilities of PokerLang and Poker Builder, one more step is needed: to create an agent to interpret the strategies defined in Poker Builder and be able to play them in real games against other agents or humans.

Chapter 5

5. Poker Builder Agent

The final step of this work is to build the poker agent that uses the strategies previously created. This agent runs on the simulator developed at LIACC, as referred in Chapter 2. To be able to follow the strategies, the agent need some reading capabilities of the information gathered at a poker table. The evaluators information is easy to get because it is perfect information (data obtained just by looking at the table), but the predictors information is more complicated. It requires a statistical study of the played hands in order to get reliable information.

Another feature, the agent needs, is an algorithm to search all the rules defined in the strategy and which one to use in certain circumstances.

An agent with these features will be an agent capable of strictly following the strategy defined previously.

5.1 Agent Structure

The agent will start by reading the strategy that he will use from the respective file. Then it will basically work like a four states machine: Pre-Flop, Flop, Turn and River.

In each of the states, the agent will follow sequentially three major steps: reading all the information of the table which includes setting the values of the evaluators and try to suit the imperfect information of the predictors, searching the most suitable rules for the table circumstances and choosing the rule to follow.

At the end of each hand, the agent will save all the hand's information: bets from the opponents, each opponent hand (if showed), the position of the opponent and more.

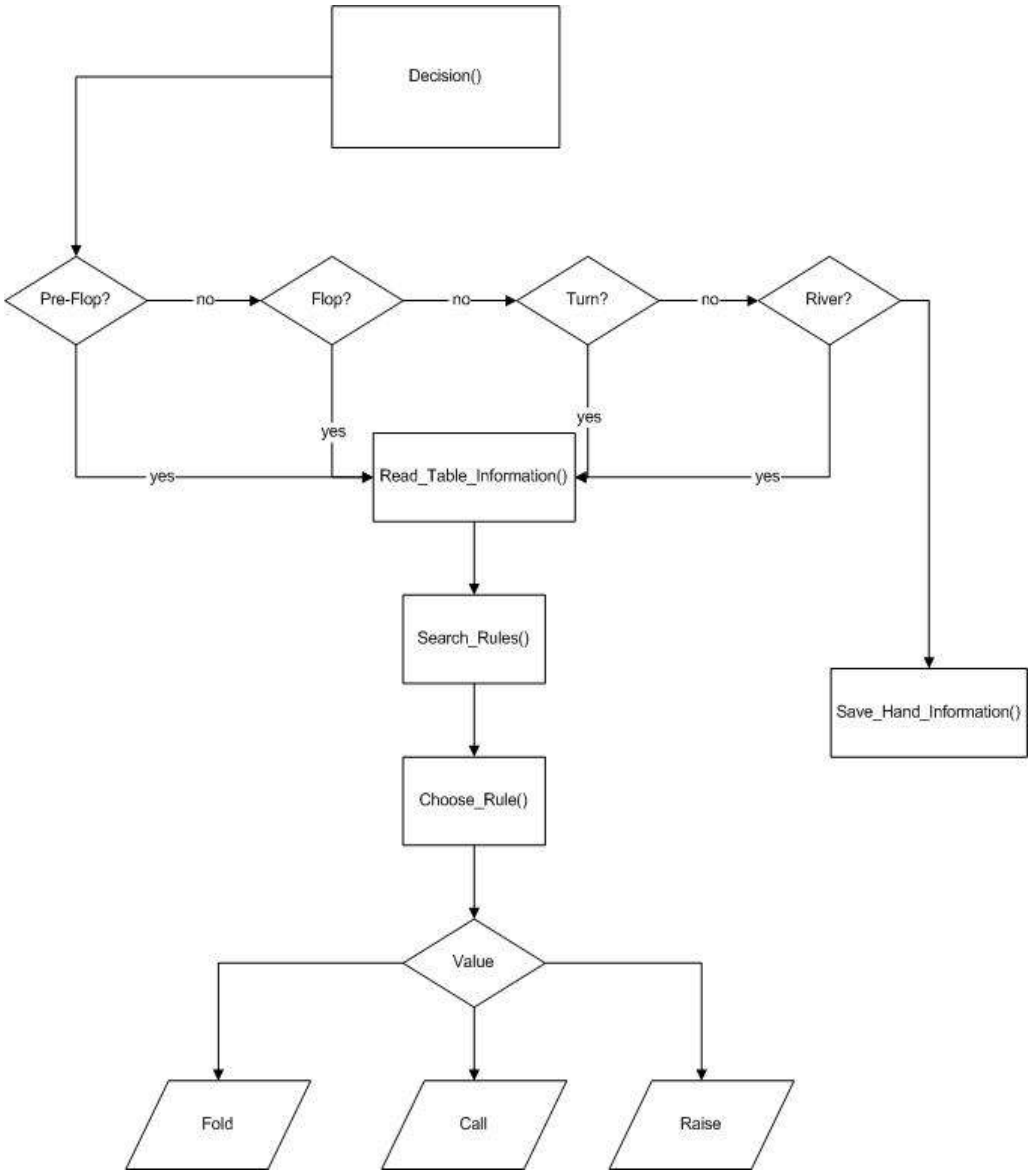

Fig 71: Sequencial Diagram of the Poker Agent

5.2 Communication Protocol

As referred previously, the agent developed is supported by LIACC's Texas Hold'em Simulator. The protocol defined in LIACC's application is based on the University of Alberta protocol that works by exchanging string messages between server and clients.

When a client connects to the server, the client sends an initial message (1) to confirm the connection.

"VERSION:1.0.1\n"

When several clients are connected (with a maximum of ten), the server activates the game by sending a message (2) to every client with the characteristics of the game.

"START:Q:W:E:R:T:Y\n"

When the message (2) is received by all clients, the server starts to send the cards to each player (3).

"MATCHSTATE:0:0::Ah2d|"

This messages are sent in a simple grammar "[1]:[2]:[3]:[4]:[5]:[6]", each field representing information of the game:

- [1] – type of message;
- [2] – number of the player that the message was sent;
- [3] – number of the game;
- [4] – current bets until now, separated by rounds using a "/" (cc/cr – 2 calls at Pre-Flop, a call and a raise at the Flop);
- [5] – player cards, after each stage new cards are added and separated by "/"; (Ah3d|/QsJh9c/3c – Ah3d are the player cards, QsJh9c are the Flop cards already in the table and 3c is the Turn card);
- [6] - movement made by the player (c – call, r – raise and f – fold) only in the messages sent from client to server;

If in (5) a player receives a "|" before his cards, it means that the player is the dealer of the hand, the rest of the players will have a "|" after the cards.

When a hand reaches the final stage (the showdown) in (4) we will have all the cards of the players in game, starting from the player with the lowest number.

5.3 File Parser

The Parser of the agent has a similar way to the Poker Builder parser. To save the Strategy and Tactics, the agent has the same four classes of the application: Strategy, Tactic, Rule and Property. With the same variables used in Poker Builder (except the Object variable that refers to Flex Objects), it becomes easier for the agent to search and follow the Strategy defined.

5.4 Predictors Analysis

To analyze and define the values of each predictor, the agent needs statistical methods to provide reliable information. This is a very important factor because if the data obtained is not majorly certain (it is impossible to be 100% certain but it can be reliable) the agent's behavior will be sustained in wrong circumstances, so the Strategy will lose effectiveness and the agent will not act as defined. For each predictor some methods will be used to define its values.

5.4.1 Implied Odds

The Implied Odds are defined by a prediction of your opponent's action in a hand that at the moment is favorable to him but with some cards dealt it becomes very unfavorable.

A player tries to predict a future bet or call of the opponent after this change. The agent will determine the value of this bet in statistical data of his actions in previous hands. Basically it will save the number of times that the opponent calls to a bet after the major change and the normal value that he calls.

$$BetValue = \frac{\sum Called\_Bets}{Number\ of\ Called\_Bets} \quad \textbf{(Eq 6)}$$

With this value, the calculation of the Implied Odds will become more reliable proportionally to the number of hands played.

5.4.2 Type of Opponent

In a poker table, a player can encounter different types of players relating to their playing style. The players can be defined by two major characteristics: the number of hands that he plays and the nature of player's bets.

The number of hands that a player plays gives us a description if he is a loose player or a tight player. This classification is given by the percentage of played hands, basically the percentage of the times that a player puts money in the pot. This calculation is often referred as the VP$IP factor. It is important to refer that the Big Blind is not considered a voluntary bet, so if a player checks in this position it is not considered in the VP$IP calculation, but if he calls a bet or raise then it is considered.

A player with a VP$IP of about 25% or less is considered a tight player. If the percentage raises above 28% or so the player is probably a loose player. Loose doesn't mean that the player looses more, is just a term that classifies opponents that plays a lot.

With the nature of player's bets, another characteristic can be defined to each player: if he is an aggressive player or a passive player. This calculation is often referred as the Aggression Factor or AF. It is a ratio (not a percentage) calculated by the formula:

$$\boldsymbol{AF = \frac{(bets + raises)}{calls}} \quad \textbf{(Eq 7)}$$

Bets, raises, and calls are all "money bets." Bets and raises, however, are aggressive (increasing the cost of playing) and calls are passive. Checks and folds are not in the equation because they are not bets.

An AF of 1.0 implies that the person makes bets about as often as they call bets.

So with an AF above 1.0 the player is considered to be aggressive and below this value, a passive player.

Knowing an opponent's style is essentially a fusion of the aforementioned aspects.

For better understanding we can look to figure 5.2:

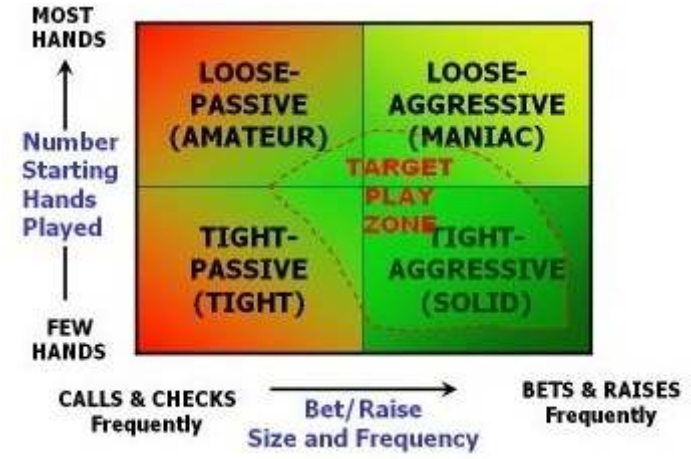

Fig 72: Types of Opponent

5.4.3 Image At Table

The Image At Table concept is calculated in the same way as the Type of Opponent concept, but in this case the calculation of the VP$IP and the Aggression Factor is directed to the hands you played.

Knowing how the opponents see the agent can be very profitable to the agent's play. If the opponents see the agent as a tight-aggressive player, the bluffs will have a higher percentage of success. In the opposite way, if the agent is seen as a loose-aggressive player, its traps will work in a very profitable way.

5.4.4 Opponent Hand

The cards the opponents hold is probably the most difficult concept to establish. The changes in behavior dramatically affect any date that you hold. The poker agent will save all the holdings that the opponent has in every showdown. Then with an analysis of his behavior, with those holdings, an estimated probability will be calculating for the possible holding hands.

This analysis must answer in the most reliable way these two questions:

- With what hands the opponent normally raises?
- With what hands the opponent normally calls?

With these questions answered, the agent can estimate with a good reliability, each opponent's hand.

5.4.5 Steal Bet

The major numbers of poker hands are played in a very quick way, one player makes a bet and everybody folds to that bet. Even when a player's hold is not that strong, this basis maintains. This is a very well-known poker move called "Steal the Pot". Independently of what cards the agent holds, if a player makes a large bet, probably everyone will fold. Every time this situation happens, the agent will save the made bet. After a significant number of hands, the agent will be able to calculate an average bet that causes this situation.

$$\boldsymbol{Steal\_Bet} = \frac{\sum \boldsymbol{Bets\_made}}{\boldsymbol{Number\ of\ times\ every\ player\ folds\ after\ bets}} \quad \textbf{(Eq 8)}$$

This gives us an estimated value of the bet needed to make your opponents fold.

5.5 Search Algorithm

After the calculation of each evaluator and predictor, the agent now needs to choose (based on the values) which rule of the strategy he will use. First, the agent will make a comparison between the Strategy rules, to know which tactic will be used. After the tactic is chosen, the agent will search the tactic rules and test them. To execute a rule, all the values defined in the rule must be satisfied. The agent will define which evaluators and predictors every rule has saving in an array variable. Each position of the array represents an evaluator or a predictor. An example: If the rule contains only the evaluator Stack, the array would be "[1,0,0,0,0,0,0,0,0,0,0]".

The agent will search every rule and compare the calculated values with the values defined in the rule. If a calculated value does not satisfied the property value in a rule, this rule is automatically discarded and the left defined values are not tested. If all property values of the rule are satisfied, the rule is saved to further choosing analysis.

When all the rules are tested, the agent will followed a certain behavior depending on the number of rules satisfied:

- If no rules are satisfied by the calculated values, the agent will check or fold the hand played;
- If one rule is satisfied, the agent will follow the rule behavior;
- If two or more rules are satisfied, the agent will define an equal probability for each rule and it will randomly choose one of them (if three rules are satisfied, each rule as a probability of 33% to be chosen).

Despite the fact that this is an extensive search, with the rules of satisfaction defined it eliminates most of the time spent in processing. Each rule is only compared 100% if none of the properties fail the test.

5.6 Conclusions

The agent had two great tasks to perform: being able to calculate every evaluator and predictor at all stages of the game and read and search the strategy rules to define what action to make.

The search needed to be fast and with the satisfaction rules defined it spares time in computer processing. The structure, of the Poker Builder strategies, in the agent was the easy part because it follows a similar structure to the application's structure.

Chapter 6

6. Results

Poker is a game with elements of chance so obtaining results is a hard task. The purpose of this work is not to build a poker agent capable of winning against every opponent but to permit the user to define poker player behaviors in a very simple way.

In the simulations made in this work, two different agents were used. One was developed by Mauro Gomes and the other by the author.

6.1 Game Simulation

In order to obtain interesting results for the purpose of this work, a statistical agent was made to save different kind of information from the one given by LIAAC's Texas Hold'em Simulator.

It had been made several heads-up games between the two agents and to test the behavior against several players, it was added extra poker agents made by Dinis Felix [Fel08]. These added agents only have a pre-flop strategy defined, based on their type of opponents, but after the pre-flop they work in the same way. If they have a good hand, they bet, if not they fold.

In both types of simulation, a reduced number of games were played in order to avoid long time simulations that take too much time. The number of games played is sufficient to achieve the goals of these simulations.

6.2 Agent PokerTron

This agent is supported by the strategy developed by Mauro Gomes. It is a simple strategy (with only one tactic and five rules), but with the easiness to define behaviors with Poker Builder, Mauro was able to create a tough poker agent, capable of trapping and bluffing their opponents along the way.

The behavior of the agent with all hands, has a good variety of moves (important to refer again, with only five rules) making it very difficult for an opponent to "read" their moves.

6.3 Agent Hansen

This agent was developed by the author with a more complex strategy than "PokerTron". It contains three different tactics, used in specific circumstances, being the choice of what tactic to use based on the current Stack. With a large stack, the agent will play a very loose game, practically never folding any hand pre-flop and trying to get their opponents out of the hand with large bets. With a normal stack it will play more specific hands (mostly group A and B) more carefully, avoiding making bluffs. With a very small stack, the agent will basically wait for a hand in group A or B and going all-in.

6.4 Competition Results

As referred previously, the simulation will be based in several matches between "PokerTron" and "Hansen", in the head's up variant and a tournament between four agents ("PokerTron", "Hansen" and two of Dinis Felix's agents). The first simulation is made to present some interesting results for the Poker Builder study. The second one is intended to see how the agents made by the application behave against agents made in different ways.

6.4.1 Head's Up

In a Head's Up match, the element of chance is very present. The fact is that it has only two hands competing between them and when a very good hand emerges for one of the players and a good hand for the other one, the match can change dramatically in just one hand. But the purpose of this simulation is to present the follow statistic:

Table 12: Rule Activation of "Hansen" and "PokerTron" Agent

	Hansen	PokerTron
Rule Activation	64%	48%

In table 6.1, we can see the percentage of rule activation for each agent, during the 10 games played. It represents the number of times that each agent makes a decision based on their strategy. The fact that a strategy is defined, it doesn't mean that it will be followed every single hand. This happens because the strategy doesn't cover all possible circumstances that can occur in a poker game. Based on the results in table 6.1, we can see that agent "Hansen" has a higher percentage of rule activation. This means that the full area of possible circumstances is more covered in agent "Hansen" then in agent

"PokerTron". It is difficult to cover all possible area but with the help of an Exploration Map it would become easier, but that is a topic to be discussed later in future work.

Other statistic to follow is the tactic activation. In the case of PokerTron, there is only one tactic defined, but in Hansen there are three. The "aggressive" tactic has a higher percentage (the agent won most of the games simulated), which means it had a high stack the most of the times. The lower stack tactic was less used mostly because with the fact that it would go all-in with a hand of group A or B, it probably lost in the times that it occurred. These results can be seen in table 6.2:

Table 13: Tactic Activation of "Hansen" Agent

	HighStack	NormalStack	LowStack
Tactical Activation	56%	39%	5%

6.4.2 Tournament

To see how Poker Builder agents react against other agents, a simulation was made consisting in several games between four agents (two agents from Poker Builder and another two agents). The other two agents used in this simulation were made by Dinis Felix.

In the first set of the simulation, the Poker Builder agents gain advantage, with a higher percentage of wins as we can see in table 6.3:

Table 14: Tournament Results

	PokerTron	Hansen	Agent 1	Agent 2
Win (%)	32%	62%	5%	1%

These were great results for the Poker Builder agents, noticing that defining these agents took a lot less time than implementing of the other two. But the fact is that "Agent 1" and "Agent 2" were in "disadvantage". These agents are only defined with a pre-flop strategy and despite the fact that they have an opponent analysis, makes it easier to lose against agents that have definitions for all four states (Pre-Flop, Flop, Turn and River).

So, to level the competition, a second set of simulated games was made, but this time only playing Pre-Flop. The results changed drastically but the "Hansen" agent was still able to win more times than the other agents:

Table 15: Tournament Pre-Flop Results

	PokerTron	Hansen	Agent 1	Agent 2
Win (%)	14%	37%	34%	15%

It also can be shown a graphical example of the stack evolution in one of the games played. In figure 6.1 we can see one of the "Hansen" victories against the other three agents.

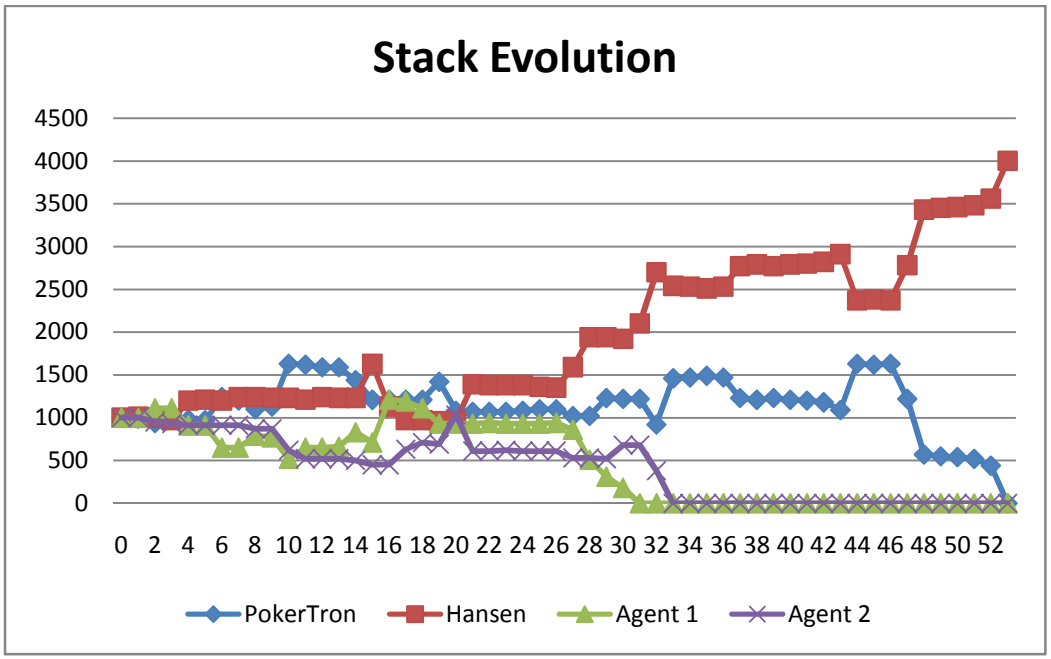

Fig 73: Stack Evolution from one of the simulated games

Both Poker Builder agents gained advantage early in the game, being able to eliminate "Agent 1" and "Agent 2" in the 31st hand and 33rd hand, respectively.

The most important fact to retrieve from these results is that Poker Builder can produce effective agents in short time and in a very simple way.

6.5 Conclusions

The simulations performed could be made with several thousand games played, but the purpose of these tests was essentially to prove the efficiency of the application and the agent that supports it.

The Heads'Up test intended to show the effectiveness of the agent reading and running the strategies defined and it has fulfilled its goals.

In the Tournament simulation, the intention was to show how the Poker Builder agents would handle different agents developed by other researchers. It reached satisfactory results, despite the fact of running only a small number of games.

More tests can be made, running an huge number of games to support these results, but the ones made enable to conclude that Poker Builder and its associated poker playing agent are capable of producing solid poker agents in a very easy way.

Chapter 7

7. Conclusions and Future Work

The purpose of this work was to make Artificial Intelligence in Poker more accessible to the common user. To build poker agents requires several years of study in Artificial Intelligence and most users don't have this knowledge. The variant Texas Hold'em was chosen because it's the most popular variant of poker nowadays.

The thesis presented an introduction to the game of poker, its variants and rules with emphasis on Texas Hold'em poker. It also overviewed some research work on AI applied to poker and presented some information about the existing tools that may be applied to the game. The overview enabled to conclude that although several tools exist that help players to play by giving statistical information and opponent characteristics there is no simple interface enabling normal players to create poker agents capable of playing at a good level.

The first goal of the project was to create a high-level language of poker concepts that could support an application to build poker strategies. The goal was accomplished and it can be created poker strategies easily, following the specifications of PokerLANG. It also allowed a better planning for the application because one of the main questions, how to create a poker agent by just creating strategies of poker concepts, was solved.

The next goal was to build the Poker Builder aplication. This application needed to make the creation of a poker agent, intuitive and accessible to common users, even without any Artificial Intelligence knowledge. The result was an application, capable of creating poker strategies using the language of concepts, with a pleasant interface and a perceptive, and easy to work with, application.

Finally, the final step was to create an agent able to use the strategies created. The agent had to be able to read the strategies created by the application, calculate several poker concepts, defined in the language, and decide the actions to make strictly following the strategy delineated. The result was an agent capable of supporting intrinsically the strategies created in the Poker Builder.

In the simulations made in Chapter 6, some interesting points were described, proving the usability and the usefulness of the application.

With this application, several courses were created for future work. The application itself can be more dominant with the addition of new features. More concepts of poker

can be added to cover up more poker specifications and to turn the agents more effective. Another feature to be created is an Exploration Map. It would be of tremendous help to know which areas of a poker game are not defined in our strategy. This would permit the user to focus on the "blank" areas and turn their agent more powerful. It also could be included, the possibility of the user to define every aspect of the concepts, including the core definition. For example: in the concept of "Type of Opponent", the user being able to characterize how to do the estimation of the opponent's type of game.

Another path that this work could evolve is to support tutorial poker applications by creating agents with specified characteristics for the lessons. Having agents with very specific behaviors creates a great foundation to teach users how to play poker and to the already poker players, to stronger their games. Or even an application that, using the strategies defined in Poker Builder, gives live support to the player, advising him what action he should make based on strategy he defined.

One the main objectives to continue this work is to build a web site that gives access to the application and permits the users to upload their strategies. The site will make several competitions allowing the creation of a huge poker agent's database which would tremendously help in the evolution of the research on this subject.

As explained, the application has a great potential, just by itself, but even more to the development of Artificial Intelligence in Poker. It creates the possibility of generating a new world in poker of Poker playing agents developed by experts.

Bibliography

[Aca08] Poker Academy of BioTools Incorporated, 2008. Available at http://www.poker-academy.com. Access in 27 June 2008

[Ado08] Adobe of Adobe System Incorporated, 2008. Available at http://www.adobe.com. Access in 27 June 2008

[Bil03] D. Billings, N. Burch, A. Davidson, R. Holte, J. Schaeffer, T. Schauenberg, and D. Szafron, *Approximating Game-Theoretic Optimal Strategies for Full-Scale Poker*, in Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-03), 2003.

[Bil06] D. Billings, *Algorithms and Assessment in Computer Poker*, Ph.D. dissertation, University of Alberta, Canada, 2006.

[Bru79] D. Brunson, *Super System - A Course in Power Poker*. Cardoza Publishing, 1979.

[Bur97] M. Buro, *The Othello match of the year: Takeshi Murakami vs. Logistello*. International Computer Chess Association Journal, 20(3):189-193, 1997.

[Che03] M. Chen, K. Dorer, E. Foroughi, F. Heintz, Z. Huang, S. Kapetanakis, K. Kostiadis, J. Kummeneje, J. Murray, I. Noda, O. Obst, P. Riley, T. Steffens, Y. Wang and X. Yin, *RoboCup Soccer Server Users Manual, for Soccer Server Version 7.07 and later*, 2003.

[Coe03] C. Coenraets. (2003) An overview of MXML: The Flex markup language. Access in 2 May 2008

[Dah98] J. Dahl,"Rounders",1998.

[Don05] C. Donninger and U. Lorenz , 2005. Hydra chess webpage. Available at http://hydrachess.com/. Access in 12 April 2008

[Fel08] D. Felix, *Artificial Intelligence Techniques in Games*, Master thesis, 2008.

[Fle08] Adobe Flex of Adobe Systems Incorporated, 2008. Available at http://www.adobe.com/products/flex/. Access in 2 May 2008

[Fuk02] RoboCup-2002 Fukuoka, 2002. Available at http://www.robocup.or.jp/fukuoka/. Access in 27 June 2008

[Ful08] Full Tilt Poker, 2008. Available at http://www.fulltiltpoker.com. Access in 27 June 2008

[Gil06] A. Gilpin and T. Sandholm, *A competitive Texas Hold'em poker player via automated abstraction and real-time equilibrium computation*, in Proceedings of the National Conference on Artificial Intelligence (AAAI-06), 2006.

[Han08] G. Hansen, *Every Hand Revealed*. Lyle Stuart Books, 2008.

[Har04] D. Harrington and B. Robertie, *Harrington on Hold 'em Expert Strategy for No Limit Tournaments, Vol. 1: Strategic Play*. 2004.

[Kor99] K.B. Korb, A.E. Nicholson and N. Jitnah, "Bayesian Poker," in Uncertainty in Artificial Intelligence (UAI-99), 1999.

[Lau01] Nuno Lau e Luís Paulo Reis, *FC Portugal 2001 Team Description:*

Configurable Strategy and Flexible Teamwork, in Andreas Birk, Silvia Coradeschi and Satoshi Tadokoro, editors, RoboCup-2001: Robot Soccer World Cup V, Springer Verlag LNAI, Vol. 2377, pp. 515-518, Berlin, 2002, ISBN 3-540-43912-9.

[Nau60] P. Naur, editor, Revised Report on the Algorithmic Language ALGOL 60, Communications of the ACM, Vol. 3 No.5, pp. 299-314, May 1960.

[New96] M. Newborn, *Kasparov versus deep blue: computer chess comes of age*. New York, U.S.A.: Springer-Verlag, 1996.

[Oel05] M. Oelbermann, *Online Poker – Driving Gambling to New Heights*. MECN Media & Entertainment Consulting Network, 2005. Access in 27 June 2008

[Off08] Poker Office, 2008. Available at http://www.pokeroffice.com Access in 27 June 2008

[Pap98] D. Papp, *Dealing with imperfect information in poker*, Master's thesis, Department of Computing Science, University of Alberta, Canada, 1998. Access in 27 June 2008

[Par08] Party Poker of WPC Productions Limited, 2008. Available at http://pt.partypoker.com Access in 27 June 2008

[Pen99] L. Pena, *Probabilities and simulations in poker*, Master's thesis, Department of Computing Science, University of Alberta, Canada, 1999. Access in 12 April 2008

[Rei01] L. P. Reis and N. Lau, *COACH UNILANG - A Standard Language for Coaching a (Robo) Soccer Team*, 2001.

[Rei01a] Luís Paulo Reis e Nuno Lau, *FC Portugal Team Description: RoboCup 2000 Simulation League Champion*, in Peter Stone, Tucker Balch and Gerhard Kraetzschmar, editors, RoboCup-2000: Robot Soccer World Cup IV, Springer LNAI, Vol. 2019, pp.29-40, Berlin, 2001, ISBN 3-540-42185-8.

[Rei01b] Luis Paulo Reis, Nuno Lau e Eugénio C. Oliveira, *Situation Based Strategic Positioning for Coordinating a Team of Homogeneous Agents* in Markus Hannebauer, Jan Wendler and Enrico Pagello Editors, Balancing Reactivity and Social Deliberation in Multi-Agent System – From RoboCup to Real-World Applications, Springer LNAI, Vol. 2103, pp. 175-197, Berlin, 2001, ISBN 3-540-42327-3.

[Rei02] Luís Paulo Reis e Nuno Lau, *COACH UNILANG – A Standard Language for Coaching a (Robo) Soccer Team*, in Andreas Birk, Silvia Coradeschi and Satoshi Tadokoro, editors, RoboCup-2001: Robot Soccer World Cup V, Springer Verlag LNAI, Vol. 2377, pp. 183-192, Berlin, 2002, ISBN 3-540-43912-9.

[Sch92] J. Schaeffer, J. C. Culberson, N. Treloar, B. Knight, P. Lu, and D. Szafron, A world championship caliber checkers program. Artificial Intelligence, 1992.

[Smi07] C. S. and M. Zinkevich. , 2007. The AAAI'07 poker competition webpage. Available at http://www.cs.ualberta.ca/~pokert/2007 Access in 27 June 2008

[Sta08] PokerStars of Rational Entertainment Enterprises Limited, 2008. Available at

http://www.pokerstars.com Access in 27 June 2008

[Tes95] G. Tesauro, Temporal difference learning and TD-gammon. Communications of the ACM, 1995.

[Una08] University of Alberta, 2008. The University of Alberta GAMES Group. Available at http://www.cs.ualberta.ca/.games/ Access in 27 June 2008

Appendix A – Glossary of Poker Terms

This glossary was taken from Billings Ph. D. dissertation [Bil06], with some additions made in Dinis Felix master thesis [Fel08]:

- **All-in**. To have one's entire stake committed to the current pot. Action continues toward a side pot, with the all-in player being eligible to win only the main pot.
- **All-in Equity**. The expected value income of a hand assuming the game will proceed to the showdown with no further betting (i.e., a fraction of the current pot, based on all possible future outcomes).
- **Bad Beat.** An unlucky loss. In particular, losing a game where the opponent probably should have folded, but instead got extremely lucky to win.
- **Bet**. To make the first wager of a betting round (compare raise).
- **Bet for Value.** To bet with the expectation of winning if called (compare bluff).
- **Big Bet**. The largest bet size in Limit poker (e.g., $20 in $10-$20 Hold'em).
- **Big Blind** (sometimes called the Large Blind). A forced bet made before the deal of the cards (e.g., $10 in $10-$20 Hold'em, posted by the second player to the left of the button).
- **Blind**. A forced bet made before the deal of the cards (see small blind and big blind).
- **Bluff** . To play a weak hand as though it were strong, with the expectation of losing if called (see also semi-bluff and pure bluff , compare bet for value).
- **Board** (or Board Cards). The community cards shared by all players.
- **Board Texture**. Classification of the type of board, such as having lots of high cards, or not having many draws (see dry).
- **Button**. The last player to act in each betting round in Texas Hold'em. Also called the dealer button, representing the person who would be the dealer in a home game.
- **Call**. To match the current level of betting. If the current level of betting is zero, the term check is preferred.

- **Cap**. (a) The maximum number of raises permitted in any single round of betting (typically four in Limit Hold'em, but occasionally unlimited). (b) (vt) To make the last permitted raise in the current betting round (e.g., after a bet, raise, and re-raise, a player caps the betting).
- **Check.** To decline to make the rst wager of a betting round (compare call).
- **Check-Raise.** To check on the rst action, with the intention of raising in the same betting round after an opponent bets.
- **Community Cards.** The public cards shared by all players.
- **Connectors.** Two cards differing by one in rank, such as 7-6. More likely to make a straight than other combinations.
- **Dominated**. A Hold'em hand that has a greatly reduced chance of winning against another because one or both cards cannot make a useful pair (e.g., KQ is dominated by AK, AQ, AA, KK, and QQ, but not by AJ or JJ).
- **Draw**. A holding with high potential to make a strong hand, such as a straight draw or a flush draw (compare made hand).
- **Draw Potential**. The relative likelihood of a hand improving to be the best if it is currently behind.
- **Drawing Dead**. Playing a draw to a hand that will only lose, such as drawing to a flush when the opponent already holds a full house.
- **Drawing Hand**. A hand that has a good draw (compare made hand).
- **Dry**. Lacking possible draws or betting action, as in a dry board or a dry game.
- **Equity (or Pot Equity**). An estimate of the expected value income from a hand that accounts for future chance outcomes, and may or may not account for the effects of future betting (e.g., all-in equity).
- **Expected Value (EV)** (also called mathematical expectation). The average amount one expects to win in a given game situation, based on the payoffs for each possible random outcome.
- **Flop.** The first three community cards dealt in Hold'em, followed by the second betting round (compare board).
- **Fold**. To discard a hand instead of matching the outstanding bet, thereby losing any chance of winning the pot.
- **Fold Equity.** The equity gained by a player when an opponent folds. In particular, the positive equity gained despite the fact that the opponent's fold was entirely correct.
- **Forward Blinds**. The logical extension of blinds for heads-up (two-player)

games, where the first player posts the small blind and the second player (button) posts the big blind (compare reverse blinds). (Both rules are seen in practice, with various casinos and online card rooms having different policies for multi-player games that have only two active players).

- **Free-Card Danger**. The risk associated with allowing an opponent to improve and win the pot without having to call a bet (in particular, when they would have folded).
- **Free-Card Raise**. To raise on the flop intending to check on the turn.
- **Game.** (a) A competitive activity in which players contend with each other according to a set of rules (in poker, a contest with two or more players). (b) A single instance of such an activity (in poker, from the initial dealing of the cards to the showdown, or until one player wins uncontested).
- **Game Theory**. Among serious poker players, game theory normally pertains to the optimal calling frequency (in response to a possible bluff), or the optimal bluffing frequency. Both depend only on the size of the bet in relation to the size of the pot.
- **Hand.** (a) A player's private cards (e.g., two hole cards in Hold'em). (b) One complete game of poker (see game (b)).
- **Heads-up.** A two-player (head-to-head) poker game.
- **Hole Card.** A private card in poker (Texas Hold'em, Omaha, 7-Stud, etc.).
- **Implied Odds**. (a) The pot odds based on the probable future size of the pot instead of the current size of the pot (positive or negative adjustments). (b) The extra money a strong hand stands to win in future betting rounds (compare reverse implied odds).
- **Kicker.** A side card, often deciding the winner when two hands are otherwise tied (e.g., a player holding Q-J when the board is Q-7-4 has top pair with a Jack kicker).
- **Large Blind** (usually called the Big Blind). A forced bet made before the deal of the cards (e.g., $10 in $10-$20 Hold'em, posted by the second player to the left of the button).
- **Loose Game.** A game having several loose players.
- **Loose Player.** A player who does not fold often (e.g., one who plays most hands at least to the ——————————op in Hold'em).
- **Made Hand**. A hand with a good chance of currently being the best, such as top pair on the ——————————op in Hold'em (compare draw).
- **Mixed Strategy.** Handling a particular type of situation in more than one way,

such as to sometimes call, and sometimes raise.

- **Offsuit**. Two cards of different suits (also called unsuited, compare suited).
- **Open-Ended Draw.** A draw to a straight with eight cards to make the straight, such as 6-5 with a board of Q-7-4 in Hold'em.
- **Outs**. Cards that will improve a hand to a probable winner (compare draw).
- **Pocket Pair.** Two cards of the same rank, such as 6-6. More likely to make three of a kind than other combinations (see set).
- **Post-flop**. The actions after the flop in Texas Hold'em, including the turn and river cards interleaved with the three betting rounds, and ending with the showdown.
- **Pot**. The common pool of all collected wagers during a game.
- **Pot Equity** (or simply Equity). An estimate of the expected value income from a hand that accounts for future chance outcomes, and may or may not account for the effects of future betting (e.g., all-in equity).
- **Pot Odds**. The ratio of the size of the pot to the size of the outstanding bet, used to determine if a draw will have a positive expected value.
- **Pre-fop**. The first round of betting in Texas Hold'em before the flop, beginning with the posting of the blinds and the dealing of the private hole cards.
- **Pure bluff** . A bluff with a hand that can only win if the opponent folds (compare semi-bluff).
- **Pure Drawing Hand**. A weak hand that can only win by completing a draw, or by a successful bluff .
- **Raise**. To increase the current level of betting. If the current level of betting is zero, the term bet is preferred.
- **Raising for a Free-card.** To raise on the ——————————op intending to check on the turn.
- **Rake.** A portion of the pot withheld by the casino or host of a poker game, typically a percentage of the pot up to some maximum, such as 5% up to $3.
- **Re-raise.** To increase to the third level of betting after a bet and a raise.
- **Reverse Blinds.** A special rule sometimes used for heads-up (two-player) games, where the second player (button) posts the small blind and the first player posts the big blind (compare forward blinds). (Both rules are seen in practice, with various casinos and online card rooms having different policies for multi-player games that have only two active players).
- **Reverse Implied Odds.** The unaccounted (negative) money a mediocre hand

stands to lose in future betting rounds (compare implied odds (b)).

- **River.** The fifth community card dealt in Hold'em, followed by the fourth (and final) betting round.
- **Semi-bluff** . A bluff when there are still cards to be dealt, with a hand that might be the best, or that has a reasonable chance of improving to the best if it is called (compare pure bluff).
- **Second pair.** Matching the second highest community card in Hold'em, such as having 7-6 with a board of Q-7-4.
- **Session.** A series of games, typically lasting several hours in length.
- **Set.** Three of a kind, formed with a pocket pair and one card of matching rank on the board. A very powerful and well-disguised hand (compare trips).
- **Short-handed Game.** A game with less than the full complement of players, such as a Texas Hold'em game with ve or fewer players.
- **Showdown.** The revealing of cards at the end of a game to determine the winner.
- **Side pot**. A second pot for the remaining active players after another player is all-in.
- **Slow-play.** To check or call a strong hand as though it were weak, with the intention of raising in a later betting round (compare smooth-call and checkraise).
- **Small Bet**. The smallest bet size in Limit poker (e.g., $10 in $10-$20 Hold'em).
- **Small Blind.** A forced bet made before the deal of the cards (e.g., $5 in $10-$20 Hold'em, posted by the first player to the left of the button).
- **Smooth-call**. To only call a bet instead of raising with a strong hand, for purposes of deception (as in a slow-play).
- **Suited.** Two cards of the same suit, such as both Hearts. More likely to make a flush than other combinations (compare offsuit or unsuited).
- **Table Image**. The general perception other players have of one's play.
- **Table Stakes.** A poker rule allowing a player who cannot match the outstanding bet to go all-in with his remaining money, and proceed to the showdown (also see side pot).
- **Texture of the Board.** Classification of the type of board, such as having lots of high cards, or not having many draws (see dry).
- **Tight Player**. A player who usually folds unless the situation is clearly

profitable (e.g., one who folds most hands before the flop in Hold'em).

- **Time Charge.** A fee charged to the players in a poker game by a casino or other host of the game, typically collected once every 30 minutes.
- **Top Pair**. Matching the highest community card in Hold'em, such as having Q-J with a board of Q-7-4.
- **Trap**. To play a strong hand as though it were weak, hoping to lure a weaker hand into betting. Usually a check-raise, or a slow-play.
- **Trips.** Three of a kind, formed with one hole card and two cards of matching rank on the board. A strong hand, but not well-disguised (compare set).
- **Turn**. The fourth community card dealt in Hold'em, followed by the third betting round.
- **Unsuited.** Two cards of different suits (also called offsuit, compare suited).
- **Value Bet**. To bet with the expectation of winning if called (compare bluff).
- **Wild Game.** A game with a lot of raising and re-raising. Also called an action game.