High-Level Language to Build Poker Agents

Luís Paulo Reis[1,3], Pedro Mendes[2], Luís Filipe Teófilo[1,2], and Henrique Lopes Cardoso[1,2]

[1] LIACC – Artificial Intelligence and Computer Science Lab., University of Porto, Portugal
[2] FEUP – Faculty of Engineering, University of Porto – DEI, Portugal
[3] EEUM – School of Engineering, University of Minho – DSI, Portugal
lpreis@dsi.uminho.pt, {ei01108,luis.teofilo,hlc}@fe.up.pt

Abstract. On the last decade Poker has been one of the most interesting subjects for artificial intelligence, because it is a game that requires game playing agents to deal with an incomplete information and stochastic scenario. The development of Poker agents has seen significant advances but it is still hard to evaluate agents' performance against human players. This is either because it is illicit to use agents in online games, or because human players cannot create agents that play like themselves due to lack of knowledge on computer science and/or AI. The purpose of this work is to fill the gap between poker players and AI in Poker by allowing players without programming skills to build their own agents. To meet this goal, a high-level language of poker concepts – PokerLang – was created, whose structure is easy to read and interpret for domain experts. This language allows for the quick definition of an agent strategy. A graphical application was also created to support the writing of PokerLang strategies. To validate this approach, some Poker players created their agents using the graphical application. Results validated the usability of the application and the language that supports it. Moreover, the created agents showed very good results against agents developed by other experts.

Keywords: Knowledge Representation, Decision Support Systems, Artificial Intelligence, Computer Games, Poker.

1 Introduction

Poker is the most popular betting game in the world. Played by millions around the world, poker has become a very profitable business. Given its growing popularity and the amounts of money involved (millions of dollars), Poker became a research subject in very different areas such as Mathematics, Artificial Intelligence and Sociology, among others. Key features such as incomplete knowledge, risk management, need for opponent modeling and dealing with unreliable information, have turned Poker into an important topic in Computer Science, especially for artificial intelligence.

Since the number of online Poker players still continues to grow, a large number of tools have been created to assist players in their game. Most tools are statistics-based programs that save information about the played games, creating statistical knowledge about opponents in order to help the user to make the right decision in future games.

The main goal of this work is to provide a tool capable of creating Poker Agents, through the definition of high level rules. This way, anyone with interest and knowledge about Poker can easily create a Poker agent, even without any computer programming skills. This work has been divided into the following sub goals:

- Create a language of concepts, which includes the main ideas behind poker moves and agent behavior.
- Build a graphical user interface for this language, which allows the user to create rules in a more simple way.
- Develop a Poker agent that follows the language specification.
- Evaluate the interface usability and the performance of the developed agent.

The rest of the paper is as follows. Section 2 presents recent methods to create Poker agents and information representation in Poker. Section 3 presents the specification for PokerLang. Section 4 presents a graphical application that was built to aid the creation of PokerLang files. Section 5 describes the agent that was built to follow PokerLang strategies. Section 6 presents some experiments and results. Finally, section 7 concludes and points directions for future research.

2 Related Work

First approaches to build Poker agents were rule-based, which involves specifying the action that should be taken for a given information set [1]. The next approaches were based on simulation techniques like [2], i.e. generating game random instances in order to obtain a statistical average and decide the action. These approaches led to the creation of agents that were able to defeat weak human opponents.

The great breakthrough in Computer Poker research was the discovery of the Counter Factual Regret Minimization Algorithm (CFR) in [3]. The CFR algorithm allows for the computation of a Nash Equilibrium strategy in large games such as Poker through self-play[1]. This could be done before through linear programming methods (e.g. Simplex) but CFR is much faster because the processing time is proportional to the number of information sets instead of to the number of game states (about 6 orders of magnitude less). Several approaches based on CFR, like Restricted Nash Response [4] and Data-biased response [5] backed up the first victories against Poker experts.

Other recent methodologies were based on pattern matching [6, 7], Monte Carlo Search Tree algorithm [8], reinforcement learning [9] and case based reasoning [10]. More recent works are described in the reviews [11, 12].

Another possible approach consists on the defining of the agent's strategy through a high level specification language. One example is the Poker Programming Language (PPL) [13], which is the most similar work to the one described in this paper. The main issue about PPL is that it only considers low level features of Poker which means that it takes a long time to specify a complete strategy. Moreover, the

[1] Self-play – an agent playing against itself or against an agent with an equal strategy.

absence of advanced game concepts (such as pot odds, implied odds and others) makes it only possible to create very basic and static strategies which are easily beaten by a medium level opponent.

3 PokerLang

Due to its stochastic nature, Poker players use rather different tactics in each game situation. A tactic is used under certain conditions that are represented by specific game features such as current stack, number of opponents, position at the table and others. A set of tactics compose the player's strategy. In order to specify these concepts and determine when to use each tactic, a high-level language was created – PokerLang – whose syntax and grammar was based on COACH UNILANG [14, 15] and similar languages [15–17]. COACH UNILANG was successfully used in the robotics soccer domain [18–21]. The generic approach of this language allows for its easy adaption to other domains.

The language root starts by defining the concept of strategy: a strategy is a set of tuples which one composed by a tactic and an activation condition for that tactic. Each activation condition corresponds to a set of verifications of the visible game features (through evaluators) or predictions about uncertain information (through predictors). The activation condition consists of comparing those features with parameterized values. The tactic is the procedure followed by the player when the activation condition is met. The tactic could be either user-defined or language predefined (based on common expert tactics). A top level specification of the language can be found below. In the following subsections, each language concept will be presented in depth.

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

3.1 Evaluators

Evaluators compare the game's visible features with given values. Since Poker is an incomplete information game, evaluators make use of only certain measures to assess how the player is standing in the game.

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

Number of Players. This evaluates how many players one is competing against. The number of players is an important measure because the higher it is, the lower is the probability of success of a given hand[2].

Stack. The stack is the relative amount of chips that a player currently owns given by formula M (Equation 1). The value has to be relative since there is a plethora of possibilities of a player's amount of chips. $M = \frac{Stack}{SB+BB+Antes}$. The stack evaluator is defined by levels. They can be predefined (see Table 1) or customized as follows.

```
<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>
```

Table 1. User defined Stack Regions

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

Pot Odds. Pot Odds is the ratio between the size of the pot and the cost calling[3] the opponent's bet. The pot odds are usually compared with the hand odds. When the pot odds are higher than the hand odds, the player should call the hand.

Hand Region. The probability of winning a game in Poker depends on the player's starting cards. There are $\frac{52!}{(52-2)!\times 2!} = 1326$ possible combinations of starting hands. This poses a problem because if the user were to define a tactic for every starting hand, the number of possible combinations would be enormous. To solve this problem, the language uses *bucketing*. Bucketing is an abstraction technique that consists of grouping different hands that should be played in a similar way [5]. PokerLang allows the users either to define their own groups or to use Dan Harrington's groups (see Table 2) [22].

[2] Hand – set of a player's cards that determine his/her score in the game.
[3] Call – match the current highest bet.

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

Table 2. Dan Harrington's Groups

Group	*Hands*
A	AA, KK, AKs
B	QQ, AK, JJ, TT
C	AQs, 99, AQ, 88, AJs
D	77, KQs, 66, ATs, 55, AJ
E	KQ, 44, KJs, 33, 22, AT, QJs

Hand Strength. This defines the minimum hand strength to activate the evaluator. The hand strength is given by the ratio between the number of hands that has lower score than the player's hand and the total number of possible hands [23].

Position at Table. The position at table is the player's relative position to the current Big Blind position[4]. The later the position is, the better chance the player has to observe his or her opponents' moves. Since games have a variable number of players, in order to better abstract the strategies, the position value is defined through the position quality PQ (Equation 2), which also depends on the type of the opponents.

PQ = Position – (Number of aggressive players + Number of tight players) (1)

The range of position quality depends on the number of players in the following proportion: Range = [-(Number of players-2), (Number of players-2)]. For instance, in a 10 player table, the range would be [-8, 8].

```
<POSITION_AT_TABLE>::= <PREDEF_POSITION_REGION>|<POSITION_REGION_DEF>
<PREDEF_POSITION_REGION>::= bad_pos | normal_pos | good_pos
<POSITION_REGION_DEF>::= <POSITION_REGION_NAME>{POSITION}
<POSITION_REGION_NAME>::= [string]
<POSITION>::= <MIN_POS> <COMP> <POS_VALUE> <COMP> <MAX_POS>
<POS_VALUE>::= <INTEGER>
```

There are 3 predefined regions but the user is allowed to defined custom regions. Being n the number of players, the regions are calculated as depicted in Equation 3.

$$Min = -n + 2, Max = n - 2, TR = (n - 2) \times 2 \quad (2)$$

$$Bad = [Min, Min + {}^{TR}/_{3} \, [Normal =] Min + {}^{TR}/_{3}, Max - {}^{TR}/_{3}$$
$$[Good =] \, Max - {}^{TR}/_{3}, Max]$$

[4] Big blind position – the position of the last player to act.

3.2 Predictors

The predictors are game features that are estimated. Since the hidden information in Poker (opponents' cards) is crucial to the game's outcome, to be competitive a player must make predictions about what is the actual game state. Predictions are based on the opponents' moves on previous games.

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

Implied Odds. Implied Odds corresponds to the pot odds but taking into account the evolution of the player's hand.

```
<IMPLIED_ODDS>::= <REAL>
```

Opponent Hand. A possible opponent hand taking into account the player's cards and the community cards[5]. For instance, if the opponent hand predictor is "Flush", this should be read as "If the opponent is able to reach a flush".

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

Type of Player. The type of the last opponent in the table taking into account his/her past behavior in the game. There are 4 predefined types of opponents based on [24].

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

Steal Bet. The steal bet is the amount of chips you need to get the pot with no hand at all. It depends on the type of opponents that one is facing.

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

Image at Table. The type of player that one's opponents see in him / her. This is rather important because if, for instance, the player is seen as a tight player, his/her bluffs will have higher probability of succeeding.

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

3.3 Actions

There are several poker plays that one can use in a game. These moves are specific ways of handling a hand to achieve a goal. In this definition, the user can choose predefined moves (based on common expert moves) or custom moves.

[5] Community card – table card that every player can score with.

```
<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>
```

Moves can be customized by defining the bet amount on each round.

```
<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>}
```

4 Poker Builder

After defining the high-level language, the next phase of this work was concerned with building a simple graphical application to allow users to easily create PokerLang files based on the group previous work on the area [25, 26]. Poker Builder is a Flex 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.

For the implementation of the language of concepts, Poker Builder is divided in four major classes: Strategy, Tactic, Rule and Property (Fig. 1). The interface begins with an instance of the Strategy Class that creates instances of all other classes depending on what the user is creating. Poker Builder gives the user two different views to create rules: Strategy View and Tactic View.

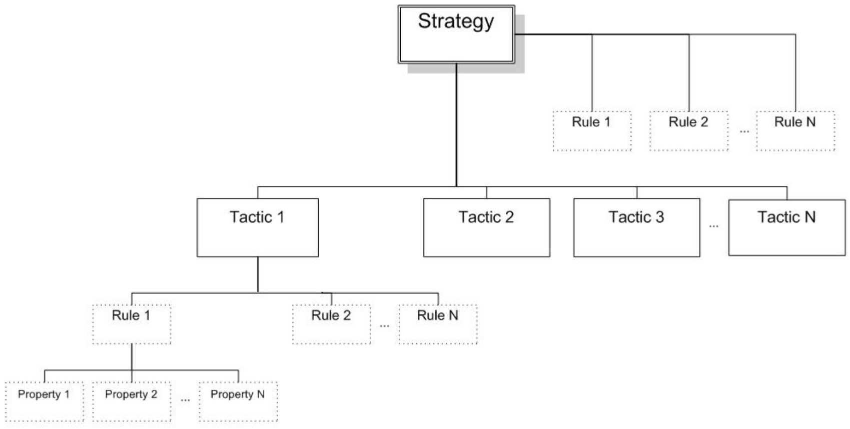
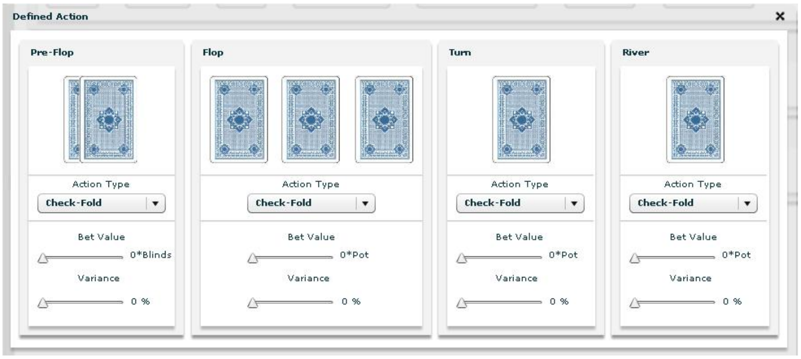


Fig. 1. Main Classes Diagram

The software includes a Strategy View to allow the user to create decision rules in what tactic the poker agent should use on the defined circumstances. It is the most high-leveled definition that the user can use from the language of concepts. 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. The software also includes a Tactic View as the main view of Poker Builder. It is presented when the program starts and is where the user defines the lower level specifications of the agent. It is presented with a list of the evaluators, the predictors and some common poker moves that professional players use in their game (actions). The menus are only available in this view, which includes the possibility of saving and loading strategies or tactics.


Defined Action

Pre-Flop
Action Type: Check-Fold
Bet Value: 0*Blinds
Variance: 0 %

Flop
Action Type: Check-Fold
Bet Value: 0*Pot
Variance: 0 %

Turn
Action Type: Check-Fold
Bet Value: 0*Pot
Variance: 0 %

River
Action Type: Check-Fold
Bet Value: 0*Pot
Variance: 0 %

Fig. 2. Action Interface in Poker Builder

5 Poker Builder Agent

The final step of this work was to build the poker agent that uses the strategies previously created. To be able to follow the strategies, the agent needs some reading features of the information gathered at a poker table. Obtaining the evaluators' features is trivial because they comprise perfect information (data obtained just by looking at the table), as opposed to predictors. Predictors require a statistical study of the played hands in order to get reliable information. Another feature required by the agent is an algorithm to select which rule to apply. An agent with these features will be an agent capable of strictly following the strategy defined previously.

The agent's action sequence starts by reading the strategy to use from the respective file. 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 trying 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 shown), the position of the opponent and more.

The agent was built to work on the LIACC Simulator described in [27]. This simulator has features that ease the construction, test and validation of the agent. Moreover, due to compatibility with AAAI simulator, it also allows the developed agent to directly participate in the annual computer poker competition [28].

6 Tests and Results

Poker is a game with elements of chance thus complicating player rating. The purpose of this work is not to build a poker agent to win against every opponent but to enable the user to define behaviors in a simple way.

All the tests were conducted in the Pre-Flop version of No Limit Texas Hold'em in head's up games. Two distinct agents were built:

- Agent PokerTron - This agent has a simple strategy (with only one tactic and five rules) but yet capable of trapping and bluffing opponents along the game. The behavior of this agent with all hands has a good variety of moves making it very difficult to read.
- Agent Hansen - This agent has a much 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 game with large bets. With a normal stack it will play more specific hands (group A and B, see Table 2) more carefully, avoiding making bluffs. With a very small stack, the agent will wait for a hand A or B and goes all-in[6].

Two simulations were run: one to test the PokerLang agents' behavior and another to test their performance against two previously developed agents.

6.1 Behavior Test

In Table 3 we can see the percentage of rule activation for each agent, during the 10 games played. This represents the number of times each agent makes a decision based on its strategy. The fact that a strategy is defined does not imply that it will be followed every single hand. This happens because the strategy does not cover all possible circumstances that can occur in a poker game. In Table 3, 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 than it is in agent PokerTron.

Table 3. Rule Activation of Hansen and PokerTron Agent

	Hansen	*PokerTron*
Rule Activation	64%	48%

[6] All-in – betting the total amount of chips.

Another important statistic is the tactic activation (Table 4). 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 simulated games), which means it had a high stack most of the times. The low stack tactic was less used because this tactic is only activated for low stack and for hands of group A/B, which did not happen often since Hansen was almost always leading the tournament.

Table 4. Tactic Activation of Hansen Agent

	HighStack	*NormalStack*	*LowStack*
Tactical Activation	56%	39%	5%

6.2 Performance Test

Hansen and PokerTron were put up against the two observing agents created by Dinis Ferreira [12] in a tournament (limited resources). Figure 3 shows that the PokerLang agents ended up competing against themselves with a final victory for Hansen (the agent with a more complex strategy).

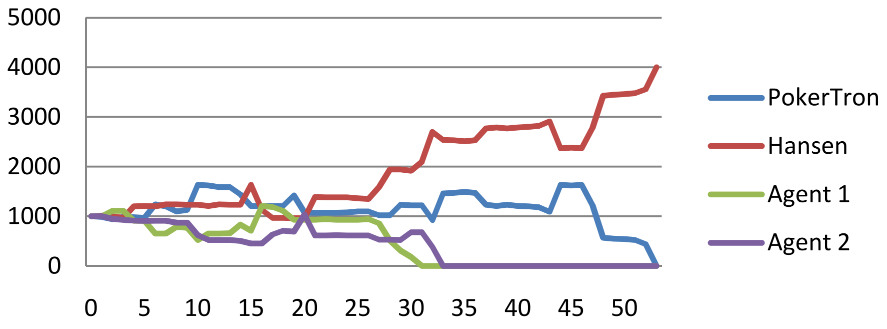


Fig. 3. Stack Evolution from one of the simulated games. Horizontal axis shows the number of hands and the vertical axis displays the agent’s stacks.

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 be used to produce effective agents in a short time and in a very simple way.

These simulations could be made with several thousand games played, but the purpose of these tests was to prove the efficiency of the application and the agent that supports it. The first test showed the effectiveness of the agent reading and running the strategies defined. In the Tournament simulation, the intention was to show how Poker Builder agents would handle different agents from another. Satisfactory results were obtained, despite the fact of running a small number of games.

7 Conclusions

The purpose of this work was to create Poker playing agents more accessible to the common user and, thus, a comprehensible high level language that represents Poker strategies was created. PokerLang filled the gaps of previous approaches like the Poker Programming Language because it allows the definition of much more complex and complete strategies. An intuitive and pleasant graphical application to support the creation of PokerLang files was also created, thus making it easier the creation of Poker playing agents.

Tests and simulations showed that the created agents correctly followed several PokerLang strategies. Moreover, agents made by Poker players were able to beat previously developed agents. However, experiments with PokerLang agents developed using professional Poker players and playing against the best poker playing agents and the best human poker players, are still required to further validate this approach.

In future research, more game concepts can be added to cover up more poker specifications and to make the agents even more effective, such as the customization of abstraction techniques. Another important feature would be the inclusion of an exploration map to allow the agent to assume how to play with information sets that were not defined, instead of just folding. The work will also be concerned with gathering professional poker player models using this language and comparing the models with the real players' behavior in order to fully and further test the expressiveness of the PokerLang language.

References

1. Billings, D., Papp, D., Schaeffer, J., Szafron, D.: Opponent modeling in poker. In: National Conf. on AI, pp. 493–499. John Wiley & Sons (1998)
2. Billings, D., Papp, D., Peña, L., Schaeffer, J., Szafron, D.: Using selective-sampling simulations in poker. In: AAAI Syring Symp. Search Tec. for Problem Solving under Uncertainty and Incomplete Information, pp. 1–6 (1999)
3. Zinkevich, M., Bowling, M., Burch, N.: A new algorithm for generating equilibria in massive zero-sum games. In: AAAI 2007, vol. 1, pp. 788–793 (2007)
4. Johanson, M.: Robust Strategies and Counter-Strategies: Building a Champion Level Computer Poker Player (2007)
5. Johanson, M., Bowling, M.: Data biased robust counter strategies. In: AISTATS 2009, pp. 264–271 (2009)
6. Teofilo, L.F., Reis, L.P.: HoldemML: A framework to generate No Limit Hold'em Poker agents from human player strategies. In: CISTI 2011, pp. 755–760 (2011)
7. Teófilo, L.F., Reis, L.P.: Building a No Limit Texas Hold'em Poker Agent Based on Game Logs Using Supervised Learning. In: Kamel, M., Karray, F., Gueaieb, W., Khamis, A. (eds.) AIS 2011. LNCS, vol. 6752, pp. 73–82. Springer, Heidelberg (2011)
8. Van den Broeck, G., Driessens, K., Ramon, J.: Monte-Carlo Tree Search in Poker Using Expected Reward Distributions. In: Zhou, Z.-H., Washio, T. (eds.) ACML 2009. LNCS, vol. 5828, pp. 367–381. Springer, Heidelberg (2009)

9. Teófilo, L.F., Passos, N., Reis, L.P., Cardoso, H.L.: Adapting Strategies to Opponent Models in Incomplete Information Games: A Reinforcement Learning Approach for Poker. In: Kamel, M., Karray, F., Hagras, H. (eds.) AIS 2012. LNCS, vol. 7326, pp. 220–227. Springer, Heidelberg (2012)
10. Rubin, J., Watson, I.: Case-based strategies in computer poker. AI Communications 25, 19–48 (2012)
11. Rubin, J., Watson, I.: Computer poker: A review. Artificial Intelligence 175, 958–987 (2011)
12. Teofilo, L.F., Reis, L.P., Cardoso, H.L.: Computer Poker Research at LIACC. In: Computer Poker Symposium, AAAI 2012 (2012)
13. Technologies, S.: Poker Programming Language User Guide (2009)
14. Reis, L.P., Lau, N.: COACH UNILANG - A Standard Language for Coaching a (Robo)Soccer Team. In: Birk, A., Coradeschi, S., Tadokoro, S. (eds.) RoboCup 2001. LNCS (LNAI), vol. 2377, pp. 183–192. Springer, Heidelberg (2002)
15. Reis, L.P., Oliveira, E.C.: A Language for Specifying Complete Timetabling Problems. In: Burke, E., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, pp. 322–341. Springer, Heidelberg (2001)
16. Pereira, A., Duarte, P., Reis, L.P.: ECOLANG - A Communication Language for Simulations of Complex Ecological Systems. In: Merkuriev, Y., Zobel, R., Kerckhoffs, E. (eds.) ECMS 2005, Riga Latvia, pp. 493–500 (2005)
17. Neves, R., Reis, L.P., Abreu, P., Faria, B.M.: A multi-agent system to help Farmville players on game management tasks. In: CISTI 2012, pp. 409–414 (2012)
18. Reis, L.P., Lau, N.: FC Portugal Team Description: RoboCup 2000 Simulation League Champion. In: Stone, P., Balch, T., Kraetzschmar, G.K. (eds.) RoboCup 2000. LNCS (LNAI), vol. 2019, pp. 29–40. Springer, Heidelberg (2001)
19. Reis, L.P., Lau, N., Oliveira, E.C.: Situation Based Strategic Positioning for Coordinating a Team of Homogeneous Agents. In: Hannebauer, M., Wendler, J., Pagello, E. (eds.) ECAI-WS 2000. LNCS (LNAI), vol. 2103, pp. 175–197. Springer, Heidelberg (2001)
20. Abreu, P.H., Moura, J., Silva, D.C., Reis, L.P., Garganta, J.: Performance analysis in soccer: A Cartesian coordinates based approach using RoboCup data. Soft Computing 16, 47–61 (2012)
21. Mota, L., Reis, L.P., Lau, N.: Multi-robot coordination using Setplays in the middle-size and simulation leagues. Mechatronics 21, 434–444 (2011)
22. Harrington, D., Robertie, B.: Harrington on Hold 'em Expert Strategy for No Limit Tournaments. Strategic Play, vol. 1. Two Plus Two Pub. (2004)
23. Teófilo, L.F.: Estimating the Probability of Winning for Texas Hold'em Poker Agents. In: 6th Doctoral Symposium on Inf. Eng., pp. 129–140 (2011)
24. Sklansky, D.: The Theory of Poker: A Professional Poker Player Teaches You How to Think Like One. Two Plus Two (2007)
25. Felix, D., Reis, L.P.: An Experimental Approach to Online Opponent Modeling in Texas Hold'em Poker. In: Zaverucha, G., da Costa, A.L. (eds.) SBIA 2008. LNCS (LNAI), vol. 5249, pp. 83–92. Springer, Heidelberg (2008)
26. Felix, D., Reis, L.P.: Opponent Modelling in Texas Hold'em Poker as the Key for Success. In: Ghalib, M., Spyropoulos, C.D., Fakotakis, N., Avouris, N. (eds.) ECAI 2008 – 18th European Conference on Artificial Intelligence, Patras, Greece, vol. 178, pp. 893–894. IOS Press (2008)
27. Teófilo, L.F., Rossetti, R., Reis, L.P., Cardoso, H.L.: A Simulation System to Support Computer Poker Research. In: 13th MABS 2012, Valência (2012)
28. Zinkevich, M., Littman, M.L.: The 2006 AAAI Computer Poker Competition. Journal of International Computer Games Association, 166–167 (2006)