Adaptive opponent modelling for the iterated prisoner's dilemma

DOI: 10.1109/CEC.2011.5949705

Adaptive Opponent Modelling for the Iterated Prisoner's Dilemma

Elio Piccolo
elio.piccolo@polito.it

Giovanni Squillero
giovanni.squillero@polito.it

***Abstract*—This paper describes the design of *Laran*, an intelligent player for the iterated prisoner's dilemma. *Laran* is based on an evolutionary algorithm, but instead of using evolution as a mean to define a suitable strategy, it uses evolution to model the behavior of its adversary. In some sense, it *understands* its opponent, and then exploits such knowledge to devise the best possible conduct. The internal model of the opponent is continuously adapted during the game to match the actual outcome of the game, taking into consideration all played actions. Whether the model is correct, *Laran* is likely to gain constant advantages and eventually win. A prototype of the proposed approach was matched against twenty players implementing state-of-the arte strategies. Results clearly demonstrated the claims.**

Keywords-component; games; iterated prisoner's dilemma; evolutionary algorithm; FSM

Introduction

The *prisoner's dilemma* is a simple puzzle of paramount importance. The simplicity of its rules is paired by the enormous amount of research performed on it. The *iterated prisoner's dilemma*, a slightly modified version of the puzzle, has been studied in many different fields, such as game theory, economics, and biology. In the past two decades, evolutionary computation scholars showed a remarkable interest in it.

This paper proposes the design of an intelligent player for the iterated prisoner's dilemma exploiting evolutionary computation. Differently from traditional approaches, it does not focus on the evolution of a fixed strategy, able to win against a generic opponent. On the contrary, it aims at devising a player that adapts its strategy considering the opponent. Such a player internally builds a reliable model of its opponent, and then determines the optimal strategy to defeat it. Borrowing the terminology from evolvable hardware, the strategy of the player is not evolved *off line*, but rather *on line*.

In the pioneering days of evolutionary computation, Lawrence Fogel defined intelligence as the composite ability to make predictions in an environment coupled with the translation of each prediction into a suitable response in light of a given goal [1]. Adopting his view, this paper proposes a methodology to build an *intelligent player*, able to foresee the behavior of its opponent and to take advantage of it.

The approach has been tested on the iterated prisoner's dilemma, but it may be used on different games as well. Moreover, instead of maximizing the strength, the ability to foresee opponent moves may be used to increase the gaming experience of a human player.

The paper is organized as follows: Section II and III introduce the prisoner's dilemma and the idea of adaptive opponent modeling. Section IV describes *Laran*, the proposed player. Section V describes the experimental evaluation. Section VI concludes the paper.

The Prisoner's Dilemma

The prisoner's dilemma is a nonzero-sum game for two players. The term "nonzero-sum" indicates that whatever benefit accrues to one player does not necessarily imply a similar penalty imposed on the other player. The prisoner's dilemma can also be defined as non-cooperative, to indicate that no communication is permitted between the players apart from game actions. However, this term is quite misleading, because the analysis of "cooperation" and emerging cooperative behaviors are usually one of the objectives of the game.

In its essential form, the game describes a situation faced by two players where there are only two possible behaviors: the first can be labeled as *cooperative* and the other *selfish*. If both players choose to cooperate, they can earn a high reward; on the other side, if both act selfishly, they will get only a low reward. But, if one is selfish and the other cooperative, the former obtains a very high reward and the latter endures a very low reward or no reward at all.

Let *R* be the payoff for mutual cooperation (reward) and *P* the payoff if both act selfishly (punishment). When only one player acts selfishly its payoff is denoted with *T* (temptation), and the payoff of its opponent with *S* (sucker). The requirement of the puzzle can be easily formalized (Equation 1). Additionally, the reward for mutual cooperation should be greater than the average of the payoff for the temptation and the sucker (Equation 2).

$$T > R > P > S \qquad (1)$$

$$2 \cdot R > S + T \qquad (2)$$

Puzzles with the above structure were devised and discussed by Merrill Flood and Melvin Dresher in 1950 as part of the *RAND Corporation*[1] investigations into game theory and its possible applications to global nuclear strategy [2]. The name "prisoner's dilemma" was chosen by Albert Tucker, who wanted to make Flood and Dresher's ideas more accessible to psychologists. He imagined two criminals arrested for an offence and placed in separate isolation cells. Each villain could opt to *cooperate* with his accomplice by negating any involvement in the crime, or *defect* by confessing to the police and betraying the (ex-)partner [3]. Since the final goal was to minimize the prison sentence instead of maximize the reward, equations (1) and (2) are reversed. But the underlying structure of the puzzle is the same.

Disregarding moral considerations, the more convenient behavior is always the selfish one. If one player cooperates, the most profitable action for the other one is to exploit the good faith being selfish. On the other hand, if one player is selfish, the best action for the other is, again, being selfish too, minimizing the damage. However, if the puzzle is indefinitely iterated the situation becomes far more interesting. On the long run, the most convenient strategy is to come to an agreement with the other player and adopt a cooperative behavior. Such version, called *iterated prisoner's dilemma*, was discussed from the time the game was first devised, but interest accelerated after the publications of Robert Axelrod [4] [5].

In 1979, Axelrod organized the first important iterated prisoner's dilemma tournament, soliciting strategies from game theorists who had published in the field [6]. The 14 entries were competed along with a fifteenth one which

[1] The *Research ANd Development* Corporation is a nonprofit global policy think tank first formed in 1948 to offer research and analysis to the United States armed forces.

cooperates or defects with equal probability. Each strategy was played against all others over a sequence of 200 moves. The winner of the tournament was submitted by Anatol Rapoport, a Russian-born American mathematical psychologist. His strategy was named “Tit for Tat”: cooperate on the first move, and then mimic whatever the other player did on the previous move.

Subsequent analysis indicated that this *Tit for tat* strategy is robust because it never defects first and is never taken advantage of for more than one iteration at a time. *Tit for tat* is not an evolutionarily stable strategy. Nevertheless, in a second tournament, Axelrod collected 62 entries and, again, the winner was *Tit for tat*.

The interest in the game blown and never faded away in subsequent years. On the one hand, the desire to find a winning strategy, on the other hand, the attempt to study why and how cooperative behaviors emerge or does not emerge kept scholars aroused.

The iterated prisoner’s dilemma has also deeply studied in the evolutionary computation community. Axelrod himself tried to evolve a population of strategies in the early 1980s using a genetic algorithm and representing strategies as bit vectors. Later, finite state machines (FSMs) became common to represent strategies [7], since they can behave like complex Markov models. Subsequent attempts also considered different paradigms, such as evolutionary programming and neural networks [8].

The interested reader may find many volumes devoted to game theory and the prisoner’s dilemma, as well as academic papers on the subject. A book by Axelrod [9] reproduces some of his papers published from 1986 to 1997, together with a new commentary. The works also consider the use a genetic algorithm, but more recent results taking advantage of evolutionary computation can be found in [10].

Adaptive Opponent Modelling

In game playing, it is often assumed that the opponent is perfectly rational and uses a strategy similar to our one. This led John von Neumann to develop the famous *Minimax* procedure in 1928 [11]. Several improvements and optimizations have been proposed in subsequent years, and some variants also tried to take into account deception or impulsiveness to improve the gaming experience. However, the fundamental goal is to maximize the reward against an opponent seeking the very same objective.

In zero-sum games, where the payoff to one player is taken away from the other, the ideas of maximizing the rewards and minimizing the damage sustained can be assumed equivalent. In more complex nonzero-sum game, however, they may be conflicting objectives. Additionally, in these cases not all players seek to minimize the maximum damage that the opponent can do. Some players are risk takers, rather than being risk averse. For instance, a player that always acts selfishly cannot be beaten, but it will probably earn a very low payoff in all its games.

Moreover, it must be considered that some players may have irrational motives to act in a specific way. This is particularly true in a game with direct moral implications, such as the prisoner’s dilemma.

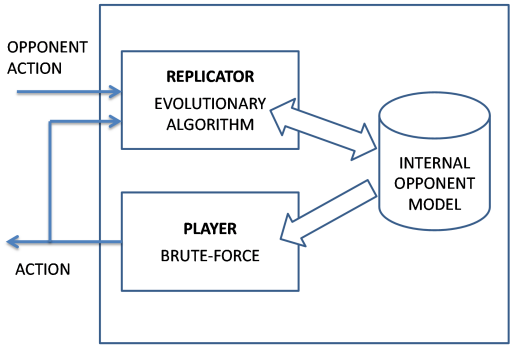

Figure 1. Adaptive Opponent Modeling

The immediate consequence is that in complex scenarios, it is advisable to consider that the opponent may be using a strategy different from our one. Thus, the Minimax algorithm is not immediately usable. The knowledge of the intent and strategy of the opponent is essential to play effectively. Indeed, the idea of modeling the opponent has been proposed from the beginning of game-playing by computers in the 1950s. Several playing algorithms have been proposed that incorporate such a knowledge, but none were ever applied truly successfully [12].

The proposed approach is based on the concept of *adaptive opponent modeling*. That is, the player is split in two loosely coupled blocks: a *replicator* sub-system able to model the current opponent as an FSM; a *player* sub-system that exploits the model to find the best next move.

The goal of the replicator sub-system is to devise a model able to *explain* past actions, and reliably *predict* future ones. It observes the game, recording the sequence of actions performed by the player itself and the counteractions taken by the opponent. Such information is eventually used to create an internal model of the strategy. The model is updated whenever new data is available, i.e., after each turn during the game.

The sub-system is based on an evolutionary algorithm. Thus, a *population* of different candidate models is cultivated. Evolutionary algorithms have been demonstrated able to cope with difficult problems, exhibiting the ability to adapt themselves to changing scenarios and are perfectly suited for the task.

The internal model is continuously modified and it is likely to change at every turn. The diversity of internal population must be preserved in order to avoid premature convergence. This sort of *over-fitting* problem would inhibit future adaptation. Thus, in each step the number *G* of generations performed by the evolutionary algorithm should be quite low. Indeed, running a limited number of generation also increase the global responsiveness of the player.

The player sub-system exploits the opponent model to find the optimal action. It considers the possible continuations of the game, selects the optimum one and plays the first action of it. The calculation needs to be repeated at every turn.

Since the opponent model is fully deterministic, the sub-system uses a very simple *brute force* approach: it generates all possible 2^n sequences of *N* moves, and picks the one leading to the best result. Obviously, since the opponent model is updated after recording the opponent action, the foreseen continuation may be proven inexact.

The depth of the brute force analysis *N* is the only parameter of the player sub-system. It directly influences the amount of resources required to select an action. Since the opponent model is an approximation and may change, calculating long sequences of moves is not useful.

Laran: Adaptive Opponent Modelling for the Iterated Prisoner's Dilemma

This paper describes a player for the iterated prisoner's dilemma taking advantage of the adaptive opponent modeling. The proposed approach was named was named *Laran*, after the god of war in Etruscan mythology.

Laran represents a strategy as a *Moore machine*, that is, an FSM whose output value is determined only by its current state and not by the value of its input. Other works on the prisoner's dilemma uses FSMs where the output is a function of the current state and of the input, called *Mealy machines*. In our opinion, Moore machines better convey the idea of the game: a player first decides its action, and after that gets the opponent's move. Anyway, Moore and Mealy machines are equivalent: it is trivial to translate the former in the latter, and it is always possible to translate the latter in the former by adding new states.

Figure 2 shows a Moore machine describing the *Tit for tat* strategy. There are two states numbered with "0" and "1". The output of the state is revealed by its border color: in light states the player cooperates, in dark ones defects. State transitions are indicated with arrows labeled with the opponent move: "c" for cooperation and "d" for defection. The starting state is marked with an arrow.

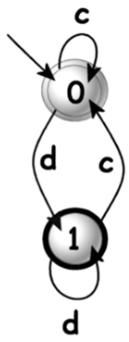

Figure 3. The Moore machine describing *Tit for tat*

Not surprisingly, the evolutionary algorithm embedded in the replicator closely resembles the pioneering work of Fogel in the choice of operators and overall structure. Some details, however, have been mutated from different evolutionary algorithms.

Evolutionary Algorithm

The evolutionary algorithm uses a steady-state approach, with a population of *I* individuals. The initial population is randomly generated. Then, in each generation *O* parents are randomly selected and a genetic operator is applied to each of them. The genetic operator is chosen randomly in a pool of four: *change initial state; add node*; *delete node*; and *modify transition*. No recombination operators have been implemented.

Change initial state clones the parent, and then modifies the initial state of the FSM selecting randomly a new node. *Add node* clones the parent, then adds a new node and connects it randomly to existing ones. *Delete node* selects a random node and creates two new individuals without it: in the first, the predecessor is connected to the successor in case of cooperation; in the second, the predecessor is connected to the successor in case of defection. *Modify transition* clones the parent, and then adds a new node and connects it randomly. All genetic operators are equally probable.

Parents are selected through a tournament of size two: two individuals are randomly picked out, compared and eventually one is chosen as the parent. To avoid bloating, a *fitness hole* is used [13]. With probability *p*, the selection criterion for tournaments is not the fitness, but the size of the individuals: the smallest individual is chosen for reproduction, not the fittest one. The name fitness hole derives from the fact that this is a hole in the probability distribution that rules selection.

Survival selection is a purely deterministic process. The worst individuals are removed until the population size comes back to *I*.

Fitness Function

As mentioned above, the goal of the replicator sub-system is to devise a model able to explain past actions and predict future ones. In order to fulfill both requirements, the model of the opponent must be able to play reasonably well. A simple FSM able to reproduce all past opponent actions may have no predicting ability. Thus, it is necessary to take into account the *strength* of the models, and not only their ability to reproduce the past records.

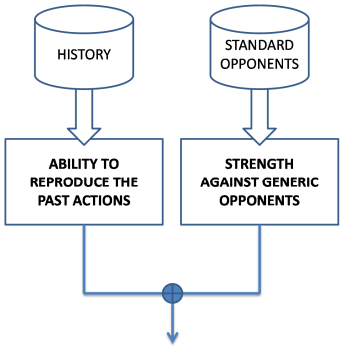

Figure 2. Fitness calculation

In order to achieve this objective, the evolutionary algorithm splits the evaluation of each candidate model in two steps (Figure 2): first, the ability to replay the past actions of the game is evaluated. The candidate gets a score measuring how many times the strategy specified correctly the actions that have been played in the game. Then, the candidate is confronted against a set of *standard opponents*. The goal of the second evaluation is to assess the strength of the strategy.

The set of standard opponents is never modified, nor does it take part of the evolutionary process. It is merely used to give a quantitative evaluation of the candidate strength. In theory, it could be replaced by a static analysis or other suitable tool.

TABLE 1: LARAN PARAMETERS

Parameter	Meaning	Value
G	Generations in the replicator	10
N	Brute-force depth	6
I	Population size	50
O	Offspring size	10
p	Fitness hole probability	0.05

In the proposed implementation, the importance of the first contribution overwhelms the importance of the second one, thus strategies are de facto required to be fully coherent with the history.

Experimental Evaluation

The proposed player was implemented in about 1,500 lines of C++. Internal parameters are summarized in Table 1.

Table 2 summarizes the payoff of the prisoner's dilemma. According to Tucker, they are expressed as years of jail sentence. Thus, the goal is to minimize the total outcome.

TABLE 2: PRISONER'S DILEMMA PAYOFF (YEARS OF JAIL SENTENCE)

Parameter	Meaning	Value
T	Temptation	0
R	Reward	1
P	Penalty	6
S	Sucker	7

For the sake of comparison, *Laran* was matched against 16 players implementing well-known strategies, and 4 dummy players:

- *AC (Always Coop)*: a simplistic player that always cooperates.
- *AD (Always Defect)*: a simplistic player that always defects.
- *BA (Brainless Alternation)*: a simple player that blindly alternates "cooperate" and "defect" actions, in this order.
- *E-BA (Evil Brainless Alternation)*: a simple player that blindly alternates "defect" and "cooperate" actions, in this order.
- *G (Grudger)*: a player that cooperates until the opponent first defects. Then it relentlessly defects until the end of the game.
- *SG (Soft Grudger)*: a player that cooperates unless the opponent defects. In this case, it punishes the behavior with a sequence of four defections. Then, it offers a *peace agreement* with two consecutive moves of cooperation.
- *TFT (Tit For Tat)*: probably the most famous and successful strategy for the iterated prisoner's dilemma. The player starts cooperating, and then replicates the opponent previous action.
- *E-TFT (Evil tit for tat)*: a slightly more malicious version of *Tit for* tat. The player starts defecting, and then replicates the opponent previous action. The strategy is also known as *Suspicious Tit For Tat.*
- *A-TFT (Anti Tit For Tat)*: an apparently illogical variant of *Tit for tat.* The player starts cooperating, and then chooses the opposite of the opponent previous action.
- *TF2T (Tit For Two Tats)*: yet another variant of *Tit for tat.* The player cooperates until the opponent plays two consecutive defections. At this point, it defects and keeps on defecting until the opponent cooperates two times in a row. Then it starts cooperating again, and so on.
- *5TM (Five is Too Much)*: similarly to *Tit for tat*, if the opponent defects the player revenges itself in the next turn. However, when the opponent cooperates five times in a row, it deefects twice.
- *GC (General Cooperator)*: if the opponent defects while the strategy cooperates the player seeks revenge in the next turn as *Tit for tat*. However, whether both defect simultaneously, it forgives and cooperates on the next turn.
- *P (Pavlov)*: it starts with a cooperative move. Then, the player repeats the last action if it was profitable, i.e., if it brought an advantage over the opponent. Otherwise, it changes actions.
- *2TT (Two to Trust)*: the player cooperates unless the opponent defects, but then it starts defecting. It requires two consecutive moves of cooperation to forgive the rudeness and start cooperating again. It is also known as *two to forgive.*
- *E-2TT (Evil Two to Trust)*: the player defects unless the opponent plays two consecutive moves of cooperation, then it starts cooperating. But if the opponent defects once, it starts defecting as in the beginning. It is equivalent to *two to trust*, but with a different starting state.
- *2TB (Two To Betray)*: the player cooperates unless the opponent defects twice in a row, then it starts defecting. However, it is eager to forgive and it starts cooperating again as soon as the opponent cooperates once.

- *E-2TB (Evil Two To Betray)*: the player defects until the opponent cooperates once. Subsequently it acts as *two to betray*: it cooperates unless the opponent defects twice in a row. Then it defects unless the opponent cooperates once.
- *TTP (Three Then Punish)*: the player cooperates unless the opponent defects three times, then it punishes the misbehavior with a sequence of three defections. After that, if the opponent cooperates, it forgives and starts cooperating again.
- *T/D (Trust/Distrust)*: the player internally ranks the trustfulness of the opponent on a scale of 1 to 4. When the opponent cooperates, the trustfulness is increased; when defects, it is decreased. Such internal value determines the action of the player: defection on 1 or 2, cooperation on 3 or 4.
- *EGET (Evil Good Evil Trust)*: the player starts with a sequence of cooperation-defection-cooperation. If the opponent replies by cooperating all three times, it thinks that it can be trusted and keeps on cooperating.

Table 3 reports the results of the evaluations. *Laran* played 100 turns of the iterated prisoner's dilemma against each opponent. The Table 3 reports, for each opponent, the cumulative jail sentence sustained by *Laran*, and the cumulative jail sentence imposed to the opponent. The minimum, maximum and average values over 10 runs are shown. The minimum average value is marked in **boldface**.

In a 100-turn match, *Laran* defeats its opponent in 12 cases out of 20 and draw once. It must be noted that a short game, or the first portion of a long one, is the most revealing test. Since the proposed approach builds an internal model of the opponent, on the long run it is always likely to win.

In fact, *Laran* is able to correctly model the *Soft grudger* strategy and after the first 100 turns its performance becomes systematically better than its opponent's. However, filling the initial disadvantage takes about 400 turns. *Tit for two tats* is correctly modeled in about 200 turns; also *Two to trust* and *Evil two to trust* are correctly modeled in almost the same time. In all these cases, the initial handicap is fully recovered in less than 1,000 turns.

TABLE 3: CUMULATIVE JAIL SENTENCES (100 TURNS)

Opponent	Sustained			Imposed		
	Min	Max	Avg	Min	Max	Avg
AC	0	1	**0.3**	694	700	698.0
AD	600	605	602.0	570	600	**588.6**
BA	303	348	**323.3**	362	632	510.0
E-BA	300	348	**318.0**	362	650	542.0
G	697	600	598.3	565	583	**575.0**
SG	198	226	208.3	184	212	**196.5**
TFT	134	344	**209.0**	141	351	216.0
E-TFT	190	285	243.3	190	285	243.3
A-TFT	0	14	**4.7**	686	700	695.3
TF2T	591	595	592.7	560	584	**574.4**
5TM	201	298	**265.3**	368	633	543.0
GC	212	276	**242.7**	408	584	492.3
P	139	194	**165.7**	146	201	172.7
2TT	223	598	472.3	202	589	**456.4**
E-2TT	559	606	588.7	496	594	**551.3**
2TB	402	588	**486.7**	458	602	553.3
E-2TB	89	126	**108.7**	189	349	293.0
TTP	160	268	**204.3**	265	492	360.7
T/D	310	590	**404.3**	292	595	444.0
EGET	168	531	310	147	524	**284.7**

Laran is able to quickly model the strategy of *Always defect*, and consequently starts defecting. But the initial loss will be never smoothed out. Similarly, when the *Grudger* starts defecting it is not possible to recover. Interestingly, *Laran* always draws when playing against *Evil tit for tat.*

The opponent playing the *Evil good evil trust* strategy is the most problematic. After 1,000 turns some experiments show a slight prevalence of it, while in some others *Laran* is winning. But results are not definite.

To conclude, when the prisoner's dilemma is iterated for 1,000 or more turns the final outcome is: 16 wins, 1 draw (*Evil tit for tat*), 2 losses (*Always defect* and *Grudger*) and 1 uncertain result (*Evil good evil trust*). A quite good result, especially considering that against *Always defect* and *Grudger* is not possible to win.

Player Analysis

To exemplify the efficacy of the proposed algorithm, Figure 3 details a game against *Trust/Distrust*. The graph plot the difference between the cumulative jail sentences imposed to the opponent and the cumulative jail sentences sustained by *Laran*. Thus, a positive value is a good result for the proposed methodology.

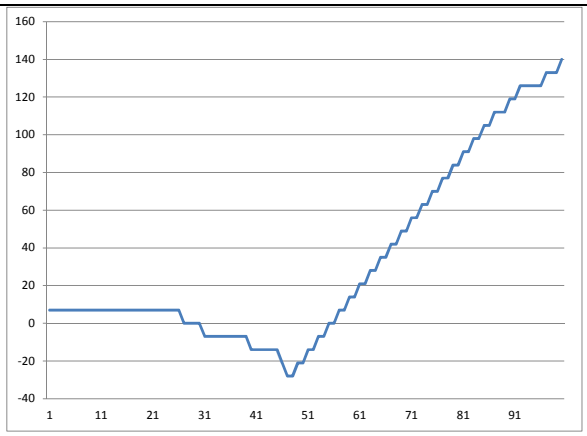

Figure 3. Progress of a game against *Trust/distrust*

The graph shows clearly three different phases: in the first turns, *Laran* is playing almost randomly. Then it starts building an incorrect model and the performances quickly drops. At about the 50th turns, it finds an (almost) correct model of the *Trust/distrust* strategy. From this point, it systematically increases its advantage.

Figure 4 shows an FSM implementing the *Trust/distrust* strategy (above), and the model built by *Laran* internally after the 100th turn. Disregarding some spurious nodes, it perfectly matches the opponent.

Conclusions and Future Works

As predicted, *Laran* was demonstrated able to correctly model and, on the long run, outperform standard strategies. The objectives of the first phase of the research are fulfilled; now it could be interesting to extend the approach.

Authors are actually working on using the methodology in different scenarios, in particular when the strategy is not fully deterministic or changes during the game. It could be interesting to apply the adaptive opponent modeling whenever the opponent is not, and cannot be, modeled as an FSM.

Furthermore, the idea to adapt a model of the opponent using an evolutionary algorithm could be applied to different domain, and with different objectives. Significantly, it could be used to increase the gaming experience in commercial game.

Finally, not relying on the Minimax procedure, authors are trying to test the approach with games that stresses cooperation rather than competition, such as Antoine Bauza's *Hanabi* [14].

Acknowledgemnts

Authors need to acknowledge Andrea Mussano for implementing *Laran* and performing the experiments.

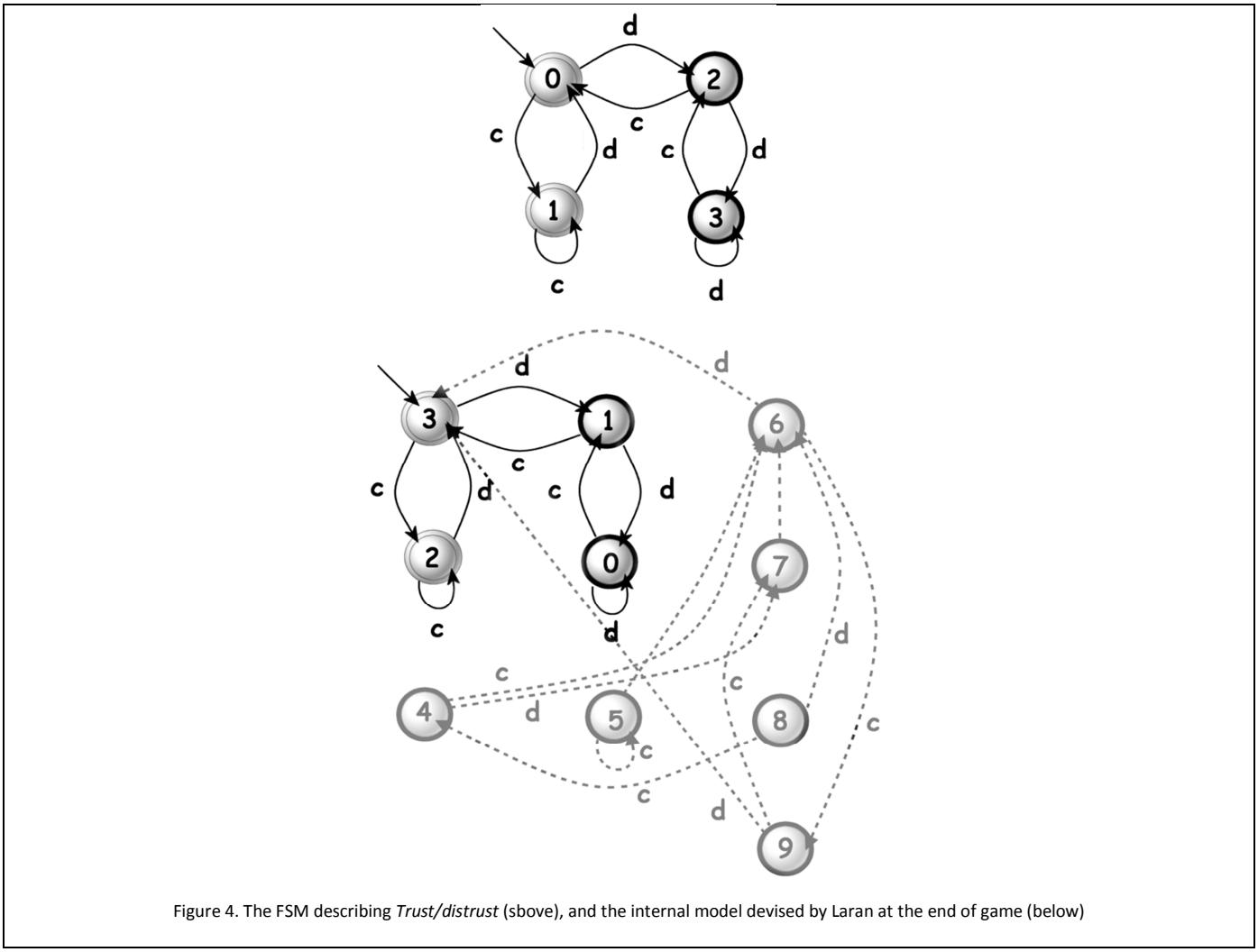

Figure 4. The FSM describing *Trust/distrust* (sbove), and the internal model devised by Laran at the end of game (below)

References

[1] A.J. Owens, M.J. Walsh L.J. Fogel, *Artificial Intelligence through Simulated Evolution*. New York: John Wiley, 1966.

[2] The Stanford Encyclopedia of Philosophy.

[3] William Poundstone, *Prisoner's Dilemma*.: Anchor , 1993.

[4] W. D. Hamilton R. Axelrod, "The Evolution of Cooperation," *Science*, vol. 211, no. 4489, pp. 1390–96, March 1981.

[5] Robert Axelrod, *The Evolution of Cooperation*.: Basic Books, 1984.

[6] R. Axelrod, "Effective choice in the iterated prisoner's," *Journal Conflict Resolution*, vol. 24, pp. 3-25, 1980.

[7] A. Rubinstein, "Finite automata play by repeated prisoner's dilemma," *ST/ICERD Discussion Paper 85/109*, 1985.

[8] K. Chellapilla and D.B. Fogel, "Evolution, neural networks, games, and intelligence," in *Proceedings of the IEEE*, 1999 , p. 1471.

[9] Axelrod R., *The Compleity of Cooperation*.: Princeton University Press, 1997.

[10] Xin Yao, Siang Yew Chong Graham Kendall, *The iterated prisoners' dilemma: 20 years on.*: World Scientific Publishing Co., 2007.

[11] J. Von Neumann, "Zur Theorie der Gesellschaftsspiele," *Mathematische Annalen*, vol. 100, no. 1, pp. 295-320, 1928.

[12] Jeroen Donkers, Pieter Spronck Jaap van den Herik, "Opponent modelling and commercial games," in *Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games*, 2005, pp. 15-25.

[13] R. Poli, "A Simple but Theoretically-Motivated Method to Control Bloat in Genetic Programming," in *Genetic Programming*, 2003, pp. 43-76.

[14] Les XII singes. Hanabi & Ikebana. [Online]. http://www.les12singes.com/sousrubrique.php3?id_rubrique=34