Learn Your Opponent's Strategy(in Polynomial Time)!

Yishay Mor and **Claudia V. Goldman** and **Jeffrey S. Rosenschein**
Computer Science Department
Hebrew University
Givat Ram, Jerusalem, Israel
ph: 011-972-2-658-5353 fax: 011-972-2-658-5439
email: yish@cs.huji.ac.il, clag@cs.huji.ac.il, jeff@cs.huji.ac.il

Abstract

Agents that interact in a distributed environment might increase their utility by behaving optimally given the strategies of the other agents. To do so, agents need to learn about those with whom they share the same world.

This paper examines interactions among agents from a game theoretic perspective. In this context, learning has been assumed as a means to reach equilibrium. We analyze the complexity of this learning process. We start with a restricted two–agent model, in which agents are represented by finite automata, and one of the agents plays a fixed strategy. We show that even with this restrictions, the learning process may be exponential in time.

We then suggest a criterion of simplicity, that induces a class of automata that are learnable in polynomial time.

Keywords: Distributed Artificial Intelligence, Learning, repeated games, automata

1 Introduction

Standard notions of equilibria in game theory involve a set of players holding strategies, such that no player can gain by deviating from his current strategy while the others' strategies stay fixed. This idea implicitly assumes some degree of knowledge about the players' strategies [Aumann and Brandenburger, 1991]. An obvious question is how this knowledge came to be. One possible answer is to have the players negotiate over their strategies. This solution cannot hold in the absence of communication.

We are interested in the case in which the players don't communicate with each other, apart from observing each other's move. Thus the problem we pose in this work is the problem of learning: how the players model their opponent and compute their best response at the same time? We are also interested in the complexity issue of the learning process.

When a player is engaged in a repeated interaction, he is in fact doing three things at the same time: first, he is playing the game defined by the payoff structure of the interaction, according to some strategy. If we attribute the players some degree of rationality, this strategy should be what the player believes is a best response to his opponent's strategy. Secondly, he is trying to learn what his opponent's strategy is. Note that the player's incentive to learn is limited to information that is relevant for his own choice of strategy. The third behavior the player might be involved in is what can be called "training". If the player assumes his opponent is also trying to learn his strategy, he might try to influence the opponent's beliefs, so as to push him towards a preferable strategy. For instance, in the repeated *Battle of the Sexes* game (Figure 1) player I might consistently play D, even at the cost of receiving a payoff of 0 for a period, in order to "teach" player II to play R.

		II	
		L	R
I	T	2 1	0 0
	D	0 0	1 2

Figure 1: The Battle of the Sexes game matrix

Much is yet to be done before a model allowing for these three simultaneous behaviors is available. We examine a restricted setting: player A chooses a strategy and plays by it. Player B tries to learn A's strategy and design her strategy as a best response to it. We require that B learn A's strategy in polynomial time. We assume A restricts herself to strategies realizable by *Deterministic Finite State Automata* (DFS). We do so for two reasons: on one hand, DFS strategies have been accepted widely as a model of *bounded rationality*. On the other hand, learning the structure of an automaton has been shown to be a very hard problem [Kearns and Vazirani, 1994].

We focused, as an example, on the repeated game of *The Prisoner's Dilemma* (Fig. 2). However, most, if not all, of our results can easily be generalized to a wider class of two-person non-zero-sum games.

		B	
		$\mathcal{D}$	$\mathcal{C}$
A	$\mathcal{D}$	P, P	T, S
	$\mathcal{C}$	S, T	R, R

Figure 2: The Prisoner's dilemma game

1.1 Related Work

Finite automata players were suggested as a model of bounded rationality, and as a means of resolving the prisoner's dilemma paradox, by Rubinstein [Rubinstein, 1985] and by Neyman [Neyman, 1985]. An extensive survey of the relevant literature appears in [Kalai, 1990]. The basic concept underlying this trend is that the players are rational, but are constrained to submit automata of limited size as their agents in the game. The number of states in the automata is accepted as a measure of their complexity. A series of "folk theorems" have shown that if the players are restricted to automata of size sub-exponential in the game length (i.e. the number of rounds) then cooperative behavior can be achieved at equilibrium.

This line of work is, in a sense, contrapositional to other common measures of complexity. Papadimitriou [Papadimitriou, 1992] has shown that as the bound on the number of states of the automaton becomes more restrictive, the problem of designing the optimal automaton becomes harder. Fortnow and Whang [Fortnow and Whang, 1994] were the first to assume total ignorance of the opponent's automaton. They show that in zero-sum games, a rational player can discover an optimal strategy w.r.t. the opponent's automaton in polynomial time, but in non-zero-sum games this is not the general case.

The apparent dissent is perhaps made clear by the following observation: Let K_A be the limit on the number of states in player A's automaton. If player B is allowed to use an automaton of size super-exponential in K_A, she can construct an automaton that will be optimal against any strategy of A. All B has to do, is to construct a 2^{K_A} deep tree, that will enable her to identify A's automaton, then compute the best-reply automaton to every K_A size automaton, and attach it to the relevant branch of the tree. This idea has two pitfalls from the point of view of traditional complexity theory. First, it is obvious that the time needed to construct such an automaton is unacceptable. Second, allowing automata of such size undermines the essence of computational learning theory: this automaton is an "instant learning machine". In fact, it serves as a table of all possible states of the world, replacing the desired decision process.

1.2 Outline of this paper

Section 2 unfolds the theoretical framework used in this paper. A central concept introduced in this section is that of automata *supporting* certain payoffs. The idea is to restrict the automata to those displaying some level of rational behavior, ensuring they cannot be exploited.

Section 3 addresses the issue of designing an automaton tuned towards a specific equilibrium payoff. The novelty of the work presented is not in the existence of the equilibrium, but in the constructive proof, presenting a polynomial time algorithm. The reason we bring this proof, is that we do not see any point in polynomial time learning of strategies that cannot be designed in polynomial time.

Section 4 is the focal point of this paper. In this section we show that even restricting the set of automata to those supporting rational payoffs is not sufficient to make them learnable. A further criterion of simplicity is needed. This criterion goes beyond the standard number-of-states criterion.

2 Preliminaries

This paper examines the role of learning in two-person, non-zero-sum repeated games. In this section we define the concepts of games and game equilibrium.

Def. 1 [Games]
A Game is a 3-tuple $\mathcal{G} = \{N, \alpha, \Pi\}$ *Where:*

- N *is the number of players,*
- $\alpha = \{\alpha^i\}_{i=1...N}$, $\alpha^i = \{\alpha^i_1 \ldots \alpha^i_{M_i}\}$ *is the set of* actions *available to player* i*, and*
- $\Pi : \times_i \alpha_i \rightarrow \mathcal{R}^N$ *is the* Payoff *function, i.e.* Π *assigns each player a real number payoff for any combination of all players actions.*

We will use π_i *to denote the payoff to player* i*.*

Def. 2 [Equilibrium]
A **(Nash) equilibrium** *in an* n*-player game is a set of strategies,* $\Sigma = \{\sigma^1, ..\sigma^n\}$*, such that, given that for all* i *player* i *plays* σ^i*, no player* j *can get a higher payoff by playing a strategy other then* σ^j*.*

Consider two players, A, and B, playing this game. Each player's strategy, σ_i, $i \in \{A, B\}$, is a sequence of actions taken by player i. Strategy i can be represented by a deterministic finite (*DFS*) automaton where i's actions are given in every state of the automaton and the transitions are determined by the actions taken by i's opponent. For example if the automaton in Fig. 3 represents A's strategy then, both players will stay in the initial state if both perform *cooperate*. A will move to the other state if he performs *cooperate* and B performs *defect* (TIT-FOR-TAT).

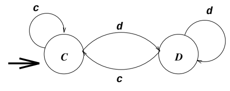

Figure 3: A's strategy - example

When one or more players are restricted to playing strategies realized by DFSs, the set of equilibria change. We will always interpret the notion of equilibrium with respect to the set of strategies available to each player. For instance, in the repeated PD game, if all players are rational, the only equilibrium is mutual defection throughout the game. However Neyman [Neyman, 1985], Rubinstein [Rubinstein, 1985] and others have shown that even if only one player is restricted to an Automaton with a limited number of states, any payoff pair in the *Individually Rational Region* (Fig. 4) can be accomplished as an equilibrium payoff.

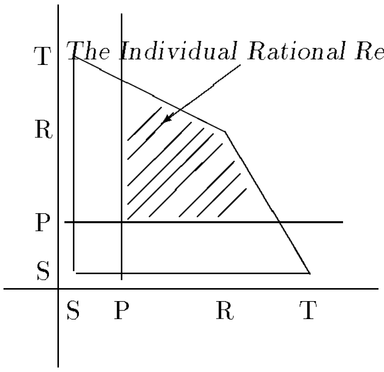

Figure 4: Payoffs of the PD game

We will denote the automaton that represents i's strategy ($i \in \{A, B\}$), and have Q states by A_i^Q. In this work, we are concerned with connected automata. [1]

When playing against an automaton, the game history is eventually cyclic. If player A is an automaton, and B is indeed trying to maximize her payoff, it is enough for her to consider only simple cycles in A (i.e. cycles in which every state is passed only once). Thus, when considering the possible payoffs induced by an automaton, it is sufficient to examine it's simple cycles.

Def. 3 *A cycle C in A_i^Q* implements $< \alpha, \beta >$ *if* $\overline{\Pi}_A(C) = \alpha$ *and* $\overline{\Pi}_B(C) = \beta$

Def. 4 *A cycle C in A_i^Q* supports $< \alpha, \beta >$[2] *if*

1. *C implements $< \alpha, \beta >$*
2. *$\forall$ C' s.t. C' implements $< \alpha', \beta' >$, if $\alpha' < \alpha$ then $\beta' < \beta$*

Def. 5 *A cycle C in A_i^Q* ϵ-supports $< \alpha, \beta >$ *if it supports $< \alpha', \beta' >$ s.t. $\| < \alpha', \beta' > - < \alpha, \beta > \| \le \epsilon$*

Def. 6 *An automaton A_i^Q* ϵ-supports $< \alpha, \beta >$ *if $\exists C \in A_i^Q$ s.t. C ϵ-supports $< \alpha, \beta >$.*

3 Polynomial Time Design of an Automaton Strategy

Theorem 1 *$\exists A_i^Q$, A_i^Q ϵ-supports $< \alpha, \beta >$.(where $\epsilon = c \times Q^{-1}$)*
Furthermore, there exists an algorithm that constructs the appropiate automaton for any $< \alpha, \beta >$ in polynomial time in Q.

Proof. For any given $< \alpha, \beta >$, we have the equalities:

$$\alpha = K_T \cdot T + K_S \cdot S + K_R \cdot R + K_P \cdot P \qquad (1)$$

$$\beta = K_S \cdot T + K_T \cdot S + K_R \cdot R + K_P \cdot P \qquad (2)$$

where K_π denotes the number of states in A_i^Q in which player A gets the payoff π[3]. Without loss of generality, we will assume P=0. In general we have, $K_T + K_S + K_R + K_P = K$. We will first normalize these values, getting $K_T + K_S + K_R = 1 - K_P$. Putting the above equations in a matrix form we have the equation:

$$\begin{pmatrix} T & S & R \\ S & T & R \\ 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} K_T \\ K_S \\ K_R \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \\ 1 - K_P \end{pmatrix}$$

Denote by M the above matrix, and by δ its determinant.

$$\delta = T^2 - S^2 - 2 \cdot R \cdot (T - S)$$

$$M^{-1} = \frac{1}{\delta} \cdot \begin{pmatrix} (T-R) & -(S-R) & R(S-T) \\ -(S-R) & (T-R) & -R(T-S) \\ S-T & -(T-S) & T^2 - S^2 \end{pmatrix}$$

Now, we can deduce the coefficients of Equation 1 and Equation 2 by fixing a value for K_P.

$$K_T = \frac{1}{\delta}(\alpha(T-R) - \beta(S-R) + (1-K_P)R(S-T)) \quad (3)$$

$$K_S = \frac{1}{\delta}(-\alpha(S-R) + \beta(T-R) - (1-K_P)R(T-S)) \quad (4)$$

$$K_R = \frac{1}{\delta}(\alpha(S-T) - \beta(T-S) + (1-K_P)(T^2-S^2)) \quad (5)$$

We have normalized the coefficients values to 1, but the sum of the coefficients should be the number of states of the automaton A_i^Q, that is Q. Hence, the final values of the coefficients are given by :

$$K_T' = K_T \cdot Q,\ K_S' = K_S \cdot Q,\ K_R' = K_R \cdot Q,\ K_P' = K_P \cdot Q \qquad (6)$$

We can now construct the desired automaton. The construction consists of three stages:

- Construct a cycle C_{imp} with $\frac{Q}{2}$ states that ϵ-implements $< \alpha, \beta >$, using the coefficients computed above to determine the number of states of each type.
- Construct a "punishment" chain C_{pun}: $\frac{Q}{2}$ states in which A plays D. The chain is linked so that B can only escape from it by playing C for $\frac{Q}{2}$ successive rounds.
- link C_{imp} to C_{pun} so that any deviation from the cycle will lead to the first state in C_{pun}, and the last punishing state is linked back to the first state of the cycle.

[1] In this we follow other researchers [Gilboa and Samet, 1989; Fortnow and Whang, 1994] who used this restriction to avoid "infinitly vengeful" strategies. Connected automata are such that have no disjoint states, i.e. for any states i, j there is an input sequence that leads from i to j. Another way to avoid unrevertable actions is to allow players to *opt out* (see [Mor and Rosenschein, 1995; Mor, 1995]).

[2] Notice that if $< \alpha, \beta >$ is not a Nash equilibrium point then it cannot be supported

[3] To be precise, this is the payoff the player gets in this state when playing his optimal strategy at that point. We will refer to the payoff that B gets as the *type* of this state. The payoff is defined by the actions of both players, while the state defines only the action of the automaton player. However, the optimal or best response strategy is unique for any state, even if the state under question would not have been reached by an optimal strategy.

This automaton ϵ-supports $< \alpha, \beta >$ because this is the way we built it.

Notice that after we have computed the coefficients from equations 4 to 6 (in $O(1)$ time), we can build the automaton by determining the action and the transitions among the states in one pass over all states. $\square$

4 Learnable and Unlearnable Strategies

4.1 An Unlearnable Automaton Strategy

Denote by $\overline{TL_B}(A_A^Q)$ the expected time that will take player B to learn A's automaton whose number of states is bounded by Q.

Theorem 2 $\exists A_A^Q$ *s.t.* A_A^Q *ϵ-supports* $< \alpha, \beta > \wedge \overline{TL_B}(A_A^Q) = \Omega(2^Q)$ [4]

Proof.
For any $< \alpha, \beta >$, we construct an automaton that ϵ-supports $< \alpha, \beta >$ as follows:

- Build a cycle, C_{imp} that implements $< \alpha, \beta >$ as in Theorem 1. Denote it the consensus cycle.
- Build the punishing chain, C_{pun} that is based on the automaton presented in [Fortnow and Whang, 1994]. The idea is to choose a random binary string of Cs and Ds and to construct the punishing chain so that B can escape from it only by following this string.

It was shown in [Fortnow and Whang, 1994] that C_{pun} cannot be learnt in polynomial time. Therefore if B enters C_{pun} then $\overline{TL_B}(A_A^Q) = \Omega(2^Q)$.

Lemma 2.1 *B enters C_{pun} with probability* $(1 - (\frac{1}{2})^{\frac{Q}{2}})$.

Proof. When B visits a state in the consensus cycle for the first time, he has no information regarding which action he should choose in order to stay in this cycle. Hence with probability $\frac{1}{2}$, B stays in the consensus cycle and with probability $\frac{1}{2}$, he enters C_{pun}. There are $\frac{Q}{2}$ states in the consensus cycle, therefore the probability of B following the whole consensus cycle is given by $(\frac{1}{2})^{\frac{Q}{2}}$ and the probability of B entering C_{pun} is $(1 - (\frac{1}{2})^{\frac{Q}{2}})$. $\square$

Figure 5 is an example of such an automaton for

$$\alpha = \beta = \frac{3P + 2R}{5}$$

[4] Exponential time might not disturb some researchers. Many of the strategies discussed in the context of PD are two or three state automata strategies. However, note that the number of states defines the granularity of the grid of possible payoff vectors. Thus, for instance, the class of two-state automata allows for 10 distinct payoff vectors, only 3 or 5 of which are in the rational region (depending or the relation between S, P, and R).

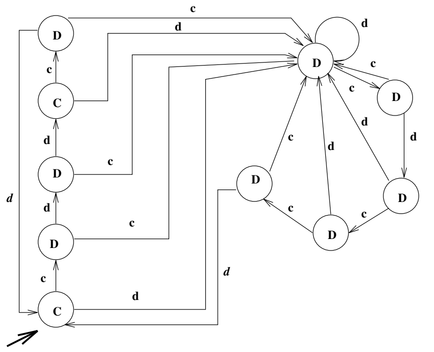

Figure 5: An Example of an Unlearnable Automaton strategy

4.2 Learnable Automata Strategies

Our objective is to categorize a class of game-playing automata that are learnable in polynomial time. The categorization we propose is based on a criterion of simplicity. Indeed, one motivation for studying automata-based strategies was their relative simplicity. Although the standard measure of complexity used for automata is the number of states, it obviously does not capture the intuitive notion of complexity. Consider the automaton we used as an example for unlearnability. Clearly, it is more complex than an automaton with the same number of states, which are all identical.

Def. 7 [C_{simp}] *In the course of theorem 1 we defined four types of states (see Footnote 3) in an automaton for the PD game. We will group the states of the automaton into chunks of connected (in the automaton graph) states of the same type. Let $\#(A)$ be the number of states in an automaton A, and $\#C(A)$ be the number of chunks of equi-type states. We denote by $<_c$ the complexity relation between two automata:*
$A <_c \tilde{A}$ if:

1. $\#(A) < \#(\tilde{A}) \vee$
2. $\#(A) = \#(\tilde{A}) \wedge \#C(A) < \#C(\tilde{A})$

The class of simple automata is:

$$C_{simp} \equiv A : \exists < \alpha, \beta > \text{ s.t. } A \text{ supports } < \alpha, \beta > \wedge$$
$$\forall \tilde{A} : \tilde{A} \text{ supports } < \alpha, \beta > \Rightarrow A <_c \tilde{A}$$

Theorem 3 *The class C_{simp} is learnable in polynomial time, i.e.* $\overline{TL_B}(A_A^Q) = O(Q)$

Proof. The proof consists of the following stages:
First, we show a canonical structure of the automata in C_{simp}, then we compute the size of this class using the canonical structure. This size is polynomial in the number of states of the automata, therefore a polynomial number of examples is sufficient to distinguish among the different automata in the class.

Lemma 3.1 *Any automaton in C_{simp} consists of a consensus cycle and a punishment chain.*

Proof. By definition, an automaton A in C_{simp} is a minimal-complexity automaton that supports a certain $< \alpha, \beta >$. Since it supports $< \alpha, \beta >$ it must have at least one cycle that implements this payoff vector. Among all these cycles, choose the one that is minimal in complexity and call it the consensus cycle (i.e. C_{imp}). The transition table of the automaton holds two entries for each state in the consensus cycle. One entry is part of this cycle and the other is not, leading to another chain which eventually will have a connection back to the consensus cycle. We are left to prove that there is at most one such chain.

Assume the contrary. Choose the chain, $\tilde{C}$, s.t. $\overline{\pi}_B(\tilde{C})$ is minimal. Denote by $s_{\tilde{C}}$ a state in $\tilde{C}$ that is accessed from the consensus cycle. Redirect all the transitions out of the consensus cycle to $s_{\tilde{C}}$. Since all the other chains different from $\tilde{C}$ and the consensus cycle are no longer accessible, remove them. We have constructed an automaton that still supports $< \alpha, \beta >$ but has less states than A in contradiction to A being in C_{simp}. Denote $\tilde{C}$ the punishment chain. □

Lemma 3.2 *All the states in the punishment chain are of type S.*

Proof. From Lemma 3.1, we know that there exists at most one punishment chain in automaton A. Replace this chain, C_{pun}, by another one, C'_{pun}, with the same number of states which are all of type S. Denote by A' the modified automaton.

$\overline{\pi}_B(C'_{pun}) \leq \overline{\pi}_B(C_{pun})$ so if A supported $< \alpha, \beta >$ so does A'. However, $A' \leq_c A$ ($A' =_c A$ when the states in C_{pun} are equi-type). therefore $A \in C_{simp}$ iff all the states in C_{pun} were equi-type.

By contradiction, assume all the states in C_{pun} are of type different from S. W.lo.g. assume this type is P. Every chain $\tilde{C} \neq C_{imp}$ in A can be decomposed into two parts: a prefix of C_{imp} that will be denoted by h, and C_{pun}. The average payoff of B in $\tilde{C}$ is given by

$$\overline{\pi}_B(\tilde{C}) = \frac{K_h \cdot \overline{\pi}_h + K_{pun} \cdot P}{K_h + K_{pun}}$$

where K_h is the number of states in h, K_{pun} is the number of states in C_{pun} and $\overline{\pi}_h$ is the average payoff that B receives in h.

Let K'_{pun} be the number of states in C_{pun} when they are all of type S and $\overline{\pi}_B(\tilde{C})$ remains unchanged.

$$\frac{K_h \cdot \overline{\pi}_h + K_{pun} \cdot P}{K_h + K_{pun}} = \frac{K_h \cdot \overline{\pi}_h + K'_{pun} \cdot S}{K_h + K'_{pun}}$$

$$K'_{pun} \leq K_{pun} \cdot \frac{K_h \cdot (\overline{\pi}_h - P)}{K_h \cdot (\overline{\pi}_h - P) + \epsilon}, \epsilon > 0$$

Therefore $K'_{pun} < K_{pun}$ in contradiction to the assumption that A is in C_{simp}. □

Lemma 3.3 *For each automaton $A \in C_{simp}$, and for each type of state t, there is at most one chunk of states of type t in A.*

Proof. By contradiction: assume that $\exists A \in C_{simp} \exists t$ s.t. there is more than one chunk of states of type t in A. We construct an automaton $\tilde{A}$ that supports the same payoff vector, and $\tilde{A} <_c A$. This contradicts A being in C_{simp}: all we have to do is to group all the states of type t together. □

Lemma 3.4 $|C_{simp}| < 4!Q^3$

Proof. There are four possible types of states, therefore by Lemma 3.3 there could be at most four chunks. Hence there are 4! possible ways to arrange them. Each chunk has at most Q states, but the number of states in one of the chunks is determined by the size of the others. Therefore there are less than Q^3 possible combinations of chunks' sizes for each of the 4! arrangements. □

We have shown that for every Q, the number of automata with $O(Q)$ states in C_{simp} is polynomial in Q. Therefore, player B could enumerate all the possible automata and he could learn which automaton is A's in time polynomial in A's automaton's size. This completes the proof of Theorem 3. □

4.3 An example of a learning algorithm

So far, we dealt with all payoff vectors in the individually rational region. However, the range of possible payoffs requires a more detailed inspection in our context.

In the setting we studied, player A designs an automaton that is "tuned" towards a certain payoff vector, and player B tries to learn that automaton and play accordingly. It is reasonable that A will choose an automaton that gives B a payoff of P (or $P + \epsilon$), so as to maximize his own payoff. However, we might want to allow more complex situations, emerging from various possible beliefs of the players. Consider, for instance a setting in which B can opt out of the game, and be matched with a different partner. If both players believe B can receive an expected payoff of θ if he opts out, then A will construct his automaton so as to award B at least θ in equilibrium [5]. Let us assume that A restricts himself to strategies that grant B a payoff of at least β at equilibrium. Still, among all these strategies, A will choose that which maximizes his own payoff. Consider again the graph in Figure 4. Given that A maximizes his payoff for a certain minimal payoff he attributes to B, the only possible payoffs to be received by both players can be represented in the upper and rightmost boundaries. The first line is defined by $K_R + K_T = 1$ and the second is defined by $K_R + K_S = 1$.

Assuming player B knows that A's automaton is simple, i.e., A's automaton is in C_{simp}, we construct the following learning algorithm for B (see Figure 6). Notice that B doesn't know how many states there are in A's automaton (Q). In the algorithm LearnSR(Q), B could have played C all the time in order to play according to A's automaton (a chunk of S type states connected to a chunk of T type states). But, if B would have played so, A could have taken advantage of that and play D forever. Hence we have added in B's learning algorithm, a step where B will play D to prevent A from abusing him. B will discover the size of Q in polynomial time since he will know it after log_2 steps.

[5] this observation coincides with empirical data on human behavior [Roth *et al.*, 1991]

```
Learn(Q):
Play C
If payoff = S then
LearnSR(Q)
else
LearnRT(Q)

LearnSR(Q):
For i=1 to ∞ {
Play C for 2^i times
Play D for 4^i times
}
```

```
LearnRT(Q):
K_R = 0
Repeat {
Play C for K_R times
Play D
Play D
If payoff = P then
K_R ← K_R + 1
else break
}
K_S = 0
While (payoff = T) {
Play D
K_S ← K_S + 1
}
Repeat {
Play C for K_R times
Play D for K_S times
}
```

Figure 6: The Learning algorithm

5 Conclusions and Future Work

When players do not communicate about their actions, it might take each of them exponential time to find the best response to his opponent's strategy. We have shown that this holds even if one of the players is playing a fixed, Nash–equilibrium strategy.

We have first defined the notion of an automaton that supports a payoff vector $< \alpha, \beta >$. We have also presented an algorithm to design an automaton that supports a certain payoff vector to be received by the players if they play according to it. We have shown that the complexity of this algorithm is polynomial in the number of states of the automaton. The reason for deriving this proof is that we do not see any point in polynomial time learning of strategies that cannot be designed in polynomial time.

We have defined the class of automata C_{simp} that can be learned in polynomial time and we have also given an example sub-class for which we have specified the learning algorithm.

Other issues that need to be further investigated regard extensions of the results presented in this paper: In the Prisoner's dilemma, a player can punish his opponent without harming himself. An interesting question is which payoffs can be supported when this doesn't hold.

In some games, equilibria in pure strategies do not exist, players must randomize their actions. It might be possible to use unlearnable automata in order to create pseudo-random strategies.

In this work we have confined ourselves to a scenario in which one player remains static and the other is adaptive. A more general model will need to account for mutual learning. In such a model, players have to learn non-fixed strategies. Furthermore, players may attempt to manipulate each other's learning process.

We have shown the existence of a learning algorithm for the class of simple automata, but have not constructed an algorithm. The automata learning literature [Rivest and Schapire, 1993; Kearns and Vazirani, 1994] shows how to construct such algorithms, when "homing sequences" are available - input sequences that guarantee a certain state is reached. A side-effect of Lemma 3.2 is to identify such a sequence: Once the learning player is thrown out of the consensus cycle, she can return to its first state by playing C for a known number of rounds.

References

[Aumann and Brandenburger, 1991] R. Aumann and A. Brandenburger. Epistemic conditions for Nash equilibrium. Working Paper 91-042, Harvard Business School, 1991.

[Fortnow and Whang, 1994] L. Fortnow and D. Whang. Optimality and domination in repeated games with bounded players. Technical report, Department of Computer Science University of Chicago, Chicago, 1994.

[Gilboa and Samet, 1989] I. Gilboa and D. Samet. Bounded vs. unbounded rationality: The tyranny of the weak. *Games and Economic Behavior*, 1:213–221, 1989.

[Kalai, 1990] Ehud Kalai. Bounded rationality and strategic complexity in repeated games. In T. Ichiishi, A. Neyman, and Y. Tauman, editors, *Game Theory and Aplications*, pages 131–157. Academic Press, San Diego, 1990.

[Kearns and Vazirani, 1994] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory.* MIT press, Cambridge, Massachusetts, 1994.

[Mor and Rosenschein, 1995] Yishay Mor and Jeffrey S. Rosenschein. Time and the prisoner's dilemma, 1995. International Conference on Multiagent Systems.(to appear).

[Mor, 1995] Yishay Mor. Computational approaches to rational choice. Master's thesis, Hebrew University, 1995. In preparation.

[Neyman, 1985] A. Neyman. Bounded complexity justifies cooperation in finitely repeated prisoner's dilemma. *Economic Letters*, pages 227–229, 1985.

[Papadimitriou, 1992] Christos H. Papadimitriou. On players with a bounded ,number of states. *Games and Economic Behavior*, 4:122–131, 1992.

[Rivest and Schapire, 1993] R. Rivest and R. Schapire. Inference of finite automata using homing sequences. *Information and Computation*, 103:299–347, 1993.

[Roth *et al.*, 1991] Alvin E. Roth, Vesna Prasnikar, Mashiro Okuno-Fujiwara, and Shmuel Zamir. Bargining and market behavior in jerusalem, ljubljana, pittsburg, and tokyo: an experimantal study. *American Economic Review*, 81(5):1068–1095, 1991.

[Rubinstein, 1985] A. Rubinstein. Finite automata play the repeated prisoner's dilemma. ST/ICERD Discussion Paper 85/109, London School of Economics, 1985.