ON PLAY BY MEANS OF COMPUTING MACHINES

(preliminary version)

Nimrod Megiddo[1] and Avi Wigderson

Mathematical Sciences Research Institute
Berkeley, California 94720

Abstract. This paper examines the "bounded rationality" inherent in play by means of computing machines. The main example is the finitely repeated prisoners' dilemma game which is discussed under different models. The game is played by Turing machines with restricted number of internal states using unlimited time and space. The observations in this paper strongly suggest that the cooperative outcome of the game can be approximated in equilibrium. Thus, the cooperative play can be approximated even if the machines memorize the entire history of the game and are capable of counting the number of stages.

---

[1] The IBM Almaden Research Center, 650 Harry Road San Jose, California 95120-6099, and Tel Aviv University, Tel Aviv, Israel.

1. Introduction

We consider here the famous prisoners' dilemma game, a version of which is stated below. The framework however applies to games in general. The prisoners' dilemma game was chosen since it focuses on the central issue raised in this paper.

Definition 1.1. By the (one-shot) *prisoners' dilemma game* G we refer to a game as follows. The game is played by two players with symmetric roles. Each player has to choose (independently of the other) between the action C ("cooperate") and the action D ("defect"). The payoffs to the two players, corresponding to the four possible combinations of choices of actions, are as follows. The players are each paid 3 units of utility if both play C. Both are paid 1 if both play D. If one plays C and the other one plays D then the one who plays C gets 0 and the other one gets 4.

Definition 1.2. The finitely repeated prisoners' dilemma consists of N rounds, where during each round the one-shot prisoners' dilemma, game is played, and both players are then informed of each other's actions. The number N is common knowledge between the players, that is each knows N, each knows that each knows N, each knows that each knows that each knows N, and so on.

Definition 1.3. A Nash-equilibrium point in a 2-person game is a pair $\sigma=(\sigma^1,\sigma^2)$ of strategies (one for each player) such that, given that player i $(i=1,2)$ is playing σ^i, the other player, $3-i$, cannot get a higher payoff for himself by playing any strategy other than σ^{3-i}.

The perplexing fact about the finitely repeated prisoners' dilemma is the following:

Proposition 1.4. *The only equilibrium payoffs in the finitely repeated prisoners' dilemma correspond to "defection" by both players throughout. Moreover, this equilibrium is obtained by iterated elimination of dominated strategies.*

Proof. It can easily be shown by induction on k that in any equilibrium the players must play (D,D) in round $N+1-k$, $(k=1,\ldots,N)$.

It has been said that the prisoners' dilemma may be "resolved" by "bounded rationality". One possibility of bounding rationality is to have the players play the game through computing machines. In this paper we discuss issues that can be dealt with under different models of play by means of computing machines. We do not attempt to present a formal theory of rationality and bounded rationality. Our work was inspired by the works of Neyman [N] and Rubinstein [Ru] where the prisoners' dilemma game was considered as being played by means of finite automata. Neyman's results are cited in Section 2. We consider two approaches: (i) Given the number of rounds, choose a machine for playing the game. (ii) Choose a machine that will be play any number of rounds, given as input at the start of the game. The only restriction is on the number of internal states of the machine. It should be emphasized that most of the results are nonconstructive and we do not believe that the approach taken in this paper will lead to a practical resolution of the prisoners' dilemma

2. Play by means of finite automata

In this section we cite results recently obtained by A. Neyman [N]. He has pointed out that finite automata [HU] of appropriate sizes may cooperate in the finitely repeated prisoners' dilemma game.

Definition 2.1. Let $G^N(s_1, s_2)$ denote a 2-person game as follows. Two ("human") players, 1 and 2, independently choose finite automata of sizes s_1 and s_2, respectively, which then play the prisoners' dilemma game for N rounds. The action of an automaton amounts to the combination of its play and the state it switches to. The action depends on its own state and on the play of the other automaton during the preceding round (except, of course, for the very first round).

A fairly simple result stated in [N] is the following:

Proposition 2.2. *(i) If* $2 \le s_1, s_2 \le N-1$ *then in the game* $G^N(s_1, s_2)$ *there exist equilibria that result in the play of* (C, C) *in each round. (ii) If either* s_1 *or* s_2 *is at least* N *then there does not exist an equilibrium point that results in the play of* (C, C) *during every stage of the game.*

Intuitively, Proposition 2.2 can be traced to the fact that an automaton with less than N states cannot recognize that it is playing the last round of an N-round game, when the last round is being played. In this context, we can state another result that follows easily by a similar argument:

Proposition 2.3. *If both* $s_1 \ge N$ *and* $s_2 \ge N$ *then every pure strategy equilibrium (that is, a pair of deterministic choices of deterministic automata, which is in equilibrium) results in the play of* (D, D) *throughout.*

Interestingly, even when $s_1 \ge N$, if $s_2 < N$ then there exist other deterministic equilibria that approximate the cooperative play. For example, if s_1 is unbounded and $s_2 = 2$, the following automata are in equilibrium. Player 2 plays C in his START state, and stays in this state as long as player 1 plays C. In this state, if 1 played D then 2 switches to his second state END, at which he plays D and stays in END, regardless of what player 1 does. Player 1 uses N of his states to count to N. In his state q_i $(i = 1, \ldots, N-1)$ he plays C and switches to state q_{i+1}, if 2 played C; otherwise, if 2 played D then 1 plays D and switches to state q_N. In his state q_N he plays D and stays in q_N.

A much stronger result, with respect to equilibria in mixed strategies in $G^N(s_1, s_2)$, is stated in [N]. A mixed strategy amounts to a probability distribution over automata. Thus, the players in this case are allowed to randomize their choices of automata, and their probabilistic choices are considered to be in equilibrium if the *expected* payoffs satisfy the equilibrium conditions. The stronger result of [N] is as follows.

Proposition 2.4. *For any integer k, there is an* N_o *with the following property: for every* $N \ge N_o$ *and every pair* (s_1, s_2) *such that* $N^{1/k} < \min(s_1, s_2) < \max(s_1, s_2) < N^k$, *there exists a mixed strategy equilibrium, where the payoffs to each player are at least* $3 - 1/k$.

3. Play by means of unrestricted Turing machines

Our purpose is to study Neyman-type models with more general computing machines. For the sake of completeness we include in the Appendix a description of how the game is actually played through Turing machines. We also include a brief definition of one type of a Turing machine. The unfamiliar reader is however advised to consult, for example, [HU] for more detail. Intuitively, one can think of computer programs instead of Turing machines. Thus, our model corresponds to a situation where real players write computer programs to play the game for them.

Our approach differs from Neyman's in several ways: (i) We consider machines with unlimited memory, whereas automata have no memory besides their states. (ii) We work with *uniform* machines, that is, our players write programs that can play *any* number of rounds, where the number of rounds is given to the program at the start of the game. (iii) We concentrate on deterministic choices of machines, whereas Neyman considers choices of probability distributions over the space of all possible deterministic machines.

Rationality in our model is bounded in the sense that, regardless of the number of rounds the game, choices of actions in the game are determined by a *finite* control, even though the entire history of the game may be held in memory. The implications of this restriction are discussed below. As we show later, finiteness in itself does not suffice for obtaining high payoffs in equilibrium. For high payoffs to be feasible in equilibrium, there has to be a specific bound on the size of the machine, that is, the number of internal states (or the length of the computer program, in the intuitive interpretation).

One may consider various definitions of the game, corresponding to different restrictions on the computational resources. In this paper we consider only restrictions on the number of internal states of the machine. In real computers this corresponds to restricting the length of the computer program, that is the space for storing the instructions, excluding the space for storing program variables. We start with the case where the number of internal states of the machine is not restricted. A more precise definition follows.

Definition 3.1. Let Γ denote a game as follows. Two players independently choose Turing machines for repeatedly playing the prisoners' dilemma game for some unknown *finite* number of rounds. The number of rounds, N, is chosen by "nature" and given to the *machines* as input before round 1.

A natural question here is whether the players are at all restricted (in the game Γ) when they have to play through Turing machines. The answer is *yes*:

Proposition 3.2. *There exist strategies for playing the game* Γ *that cannot be played through any Turing machine.*

Proof. The proof follows by a classical cardinality argument. Since a strategy for Γ amounts to choosing a strategy for each N-round game, it follows that there are $2^{\aleph_0}$ strategies for playing the game Γ. On the other, there are only $\aleph_0$ Turing machines.

Those strategies of Γ, which can be played by Turing machines, may be called *recursive strategies.* The "advantage" of the definition of Γ is that it allows for "bounded rationality" without imposing a strict limit. Rationality is bounded in the sense that players are restricted in their decision rules, even though they may have unlimited time and space for computation.

So far, we have not defined a payoff function for the game Γ. In other words, we have not specified how the number N is chosen by nature after the players have selected their machines. However, we show below that, regardless of the rule by which payoffs for different values of N are aggregated, there does not exist a "cooperative equilibrium". Given any pair of strategies (σ_1, σ_2), we say that player i has a profitable deviation if there is a strategy σ_i^* that yields him payoff higher than σ_i, given that the other player plays σ_{3-i}.

Proposition 3.3. *For any pair of machines* (T_1, T_2) *playing* Γ *and for each player* i $(i = 1, 2)$, *there exists a machine* T_i^* *with the following property: for every* N, *if in the* N*-round game there exists a profitable deviation for* i *(relative to the play of* (T_1, T_2)*), then* T_i^* *discovers such a possibility for a profitable deviation and executes it.*

Proof. The proof is based on a "simulation" argument. One Turing machine can simulate the execution of another, that is, compute the entire run of the other machine. Since the size of the machine is not restricted, player i can choose a machine that, given N, simulates the entire play of T_1 against T_2. This machine can thus find a profitable deviation (whenever there exists one) and then play the game, carrying out that deviation.

We thus conclude the case of unrestricted machines with the following theorem, which follows from Propositions 1.4 and 3.3:

Theorem 3.4. *If the players are restricted to recursive strategies, but the size of machine is not restricted, then every equilibrium results in the play of* (D, D) *throughout any* N*-round game.*

Remark 3.5. It can be shown that even when players are allowed to choose probabilistic machines, still every equilibrium pair of machines results in the steady play of (D, D).

4. On machines with limited numbers of internal states

It is important to note that in all our models we do *not* require the machines to halt after N rounds. Such a requirement would probably make most of the questions trivial. Of course, only the first N rounds are taken into the account of payoffs.

Henceforth, we consider only play through machines with restricted numbers of states, that is, each player must choose a machine whose size does not exceed some given upper bound. The simulation argument used in the preceding section does not hold anymore. A simulation may require a number of additional states which exceeds the limit. Thus, machines may be in a highly paying equilibrium if they use a number of states close to the limit. In the present section we study this aspect. We first consider (for a didactical purpose) the case where the number of rounds is *fixed* in advance, that is, "given N, choose a machine." We return later to the uniform case of "choose a machine, get N".

Definition 4.1. Let $\Gamma^N(s_1, s_2)$ denote the N-round game where players, knowing N, choose machines of no more than s_1 and s_2 states, respectively.

Consider any sequence $S = ((S_1^1; S_1^2), \ldots, (S_N^1, S_N^2))$, of pairs of choices in the N-round game, that is, $S_j^i \in \{C, D\}$, $(i = 1, 2,\ j = 1, \ldots, N)$. Obviously, there are many pairs of machines that together produce the same sequence S. Denote by $\mathcal{T}(S)$ the set of all such pairs of machines.

Proposition 4.2. *The set* $\mathcal{T}(S)$ *is a cartesian product,* $\mathcal{T}(S) = \mathcal{T}^1(S) \times \mathcal{T}^2(S)$, *of sets of machines for the individual players.*

Proof. It is easy to verify that if (T_1, T_2) and (T'_1, T'_2) produce S then so do (T_1, T'_2) and (T'_1, T_2).

Let us define more restricted sets $\mathcal{R}^1$ and $\mathcal{R}^2$ as follows.

Definition 4.3. For any S, let $\mathcal{R}^i = \mathcal{R}^i(S)$ be the set of all the machines for player i that do the following:

(i) Play S_1^i in round 1.

(ii) Keep playing according to S if the opponent does.

(iii) If during any round the opponent deviates from S then switch to playing D in all following rounds.

Let us examine the possibility of achieving (in equilibrium) payoffs higher than those of the steady play of (D, D), when the number of rounds N is fixed, while the number of states is restricted. The question is which sequences S can be realized in equilibrium, given the limits on the numbers of states. Player i is mostly limited with respect to S if he must use all of his states for playing according to S. Intuitively, sequences S, with respect to which the players are mostly limited, are good candidates to be realizable in equilibrium, since the machines have minimum freedom for deviating from such strategies.

Definition 4.4. For any sequence S, other than the steady play of (D, D), let the *critical round*, $j^* = j^*(S)$, be the last round where at least one of the players plays C.

Proposition 4.5. *Let* S *be any symmetric sequence of actions (that is,* $S_j^1 = S_j^2$ *for every* j*) other than the steady* (D, D). *Suppose player* 2 *chooses a machine* $T_2 \in \mathcal{R}^2(S)$. *Under these conditions, if* T_1 *is a machine for player* 1, *that yields him a payoff higher than the machines in* $\mathcal{R}^1(S)$, *then* T_1 *plays according to* S *in rounds* $1, \ldots, j^*(S) - 1$ *and then switches to playing* D.

Proof. The proof follows from the fact that if 1 deviates from S during some round $j<j^*$ then his payoff can increase only in round j. However, the local gain (getting 4 instead of 3) is washed off by the loss in round j^* (a drop from 3 to 1) which is caused by this deviation.

We are now led to the interesting problem of designing highly paying sequences S, which are feasible subject to the limit on the number of internal states, such that none of the players can deviate *precisely* in round $j^*(S)$. We first develop some concepts that suggest solutions to the problem. Recall that we do not require the machines to halt at any time after round N.

Definition 4.6. Consider a Turing machine T that "plays" either C or D (see the Appendix for a detailed description of how the game is played). We say that T can *count* if for *any* natural number N, starting with the number N written on the work tape (in binary), T plays C precisely N times in a row, and then plays D.

Proposition 4.7. *There exists a Turing machine with two states that can count.*

Proof. The construction is simple and is left as an exercise to the reader.

Denote by $f(s)$ the number of different Turing machines (as specified in the Appendix) with s internal states that work with three tape symbols: $\{0,1,\#\}$. It is easy to see that there are integers β and γ such that $f(s)=(\beta s)^{\gamma s}$. Obviously, the number of different strings (of C's and D's) that such machines can generate is bounded by $f(s)$. This observation gives rise to the following definition, which is in the spirit of Kolmogorov complexity [K]:

Definition 4.8. For any $x \in \{C,D\}^k$ (that is, x is a string of of length k consisting of C's and D's) the *state-complexity of* x, denoted $s(x)$, is defined to be the minimum number of internal states in a Turing machine (as specified in the Appendix) that outputs the string x and then halts.

The configuration of two machines playing together may be thought of as a single "distributed" machine producing some output. However, if the machines send each other the same signals then this configuration is duplicating the work of each of them. Thus, $s(x)$ is also the minimum number of states in each member of a pair of Turing machines that (when playing together some finitely repeated prisoners' dilemma) produce the symmetric sequence corresponding to x and then *halt*.

We are interested in strings x that are hard to construct. The idea is that each machine would have to "waste" most of its power on the generation of the string x.

Remark 4.9. Suppose the machines also have to watch each other, that is, to switch to D as soon as the opponent has deviated from x. Here more states may be required. However, for symmetric play, only one more state suffices. We assume each machine can read in round $i>1$ the moves of both machines during round $i-1$ before it has to choose its move for round i. Alternately, it suffices to assume that each machine can read whether the moves in the preceding round were identical.

Thus, what the machine has to do is verify that in the preceding round both players made the same move; if not, it enters a special state NOMORE. Once in NOMORE, it always plays D. In fact, often there no additional state is required. Suppose the machine is not taking full advantage of the power of s states. For example, the machine never finds itself in a situation that it reads a blank on its work tape while its internal state is some q_i. This suggests a slot that can be used for watching the opponent. We can modify the machine as follows. Whenever the moves of the preceding round are not identical, the machine erases the contents of the current cell of its work tape, leaves the work-head in its place, and switches to internal state q_i. If the machine reads a blank while in state q_i then it plays D, and does not change anything else. Thus, in symmetric play at most one additional state may be required for watching the opponent and this may happen only when the full power of s states is used.

Definition 4.10. A string x is called *efficient* with respect to s if it exploits all the states, that is, $s(x) = s$.

Proposition 4.11. *For every s, there exists an string x, efficient with respect to s, whose length is not greater than* $\log_2 f(s)$.

Proof. There are precisely 2^k different strings of length k. On the other hand, there are at most $f(s)$ different strings that can be produced by s-state machines. Let $k = \lfloor \log_2 f(s) \rfloor$. It follows that there is a string y of length $k+1$ that cannot be produced by any s-state machine. Let $y = y_1 y_2 .. y_{l+1}$ be such an infeasible string of *minimal* length (hence $l \leq k$). Consider the prefix $y' = y_1 .. y_l$. This latter string *is* feasible. Suppose, per absurdum, it is not efficient. Thus it can be produced by an $(s-1)$-machine T which halts after y' is produced. It is this halting property that allows us to modify T into an s-state machine T' that produces the string y. Specifically, instead of entering the halting state of T, the modified machine T' enters the additional state, prints y_{l+1}, and then halts. Thus, we have reached a contradiction and it follows that y' is efficient.

5. Fixed N, deterministic machines, unlimited space

In this section we consider the case of a fixed N, that is, the players may choose different machines for different numbers of rounds. For simplicity of the discussion, let us assume that these machines receive no input in this case.

We propose a machine that works as follows. It plays according to a string of the form xC^∞, where x is a certain string of length k, and C^∞ is an infinite string of C's. If during any round the opponent machine has not played the same way, our machine switches to playing D throughout. Recall that $f(s) = (\beta s)^{\gamma s}$ denotes the number of different Turing machines (as specified in the Appendix) with no more than s states. The following proposition is analogous to Proposition 4.11:

Proposition 5.1. *There exists a string x of length $O(s \log s)$ such that xC^∞ is played by some s-state machine (when the opponent is doing the same) but not by any $(s-1)$-state machine.*

Proof. There are 2^k distinct infinite strings of the form xC^∞ where x is of length k. Obviously, if $2^k > f(s)$ then there exists a string x of length k, such that xC^∞ cannot be produced by any s-state machine. As in Proposition 4.11, let y' denote the *suffix* of length l of a string y of length $l+1$, where yC^∞ cannot be produced by any s-state machine and, furthermore, y is of minimal length with respect to this property. It follows that $l \le \log_2 f(s) = O(s \log s)$. Moreover, y' cannot be played by any $(s-1)$-state machine.

Obviously, we can also show that there exists a string xC^∞, where x is of length $O(s \log s)$, which can be played by an s-state machine that also *watches the opponent*, such that no $(s-1)$-state machine can do the same.

At this point it seems that we are close to proving the existence of a high-payoff equilibrium. Suppose both players use the same machine T^* that plays xC^∞ and also watches the opponent, that is, switches to playing D as soon as the opponent deviates from the string xC^∞. Moreover, suppose x is so complex that no $(s-1)$-state machine can do the same. It is conceivable that there is an $(s-1)$-state machine that plays xC^∞ without watching the opponent. But since one additional state always suffices for watching the opponent, it follows that there is no $(s-2)$-state machine that plays xC^∞. The resulting play is described by the string xC^{N-k}, where $k < N$ is the length of x, even though the machines keep on playing C ad infinitum. By Proposition 4.5, the only profitable deviation for either player is to play according to xC^{N-k-1} during the first $N-1$ rounds, and then play D. The key question is whether there exists an s-state machine that does it. In other words, the only way to deviate profitably is to use a machine that not only plays according to the string xC^{N-k-1} in the first $N-1$ rounds but also "counts" so as to play D in round N. Intuitively, this counting task seems to require (at least for *some* strings x obtained in this way) computational resources that are not available (for example, two more internal states). Of course, there may exist special strings x such that although xC^∞ requires $s-1$ states, there is a machine that produces enough information on its work tape so that it can play D in round N with just one more state. However, this seems to imply some connection between the string x and the number N. It seems reasonable to conjecture that there is at least one string x of length $O(\log f(s))$ for which this does not happen.

Motivated by the previous discussion, we now state a specific conjecture and prove results implied by it. Note that we do not have any specific reason to insist on play of strings of the form xC^{N-k}. The only property we need from a string is that it be complex and contain sufficiently many C's. Recall that $f(s)$ is the number of Turing machines (as defined in this paper) with s states. We say that a Turing machine T *produces* a string x if the output of T starts with x. In particular, the machine is not required to halt or even to stop outputing.

The Hypothesis. *If N and s are given so that* $\log f(s) = o(N)$ *then there exists an N-string x with following properties: (i) x is produced by some s-state Turing machine T. (ii) x contains only $o(N)$ D's. (iii) If j is the largest index such that $x_j = C$ then the N-string x', where $x'_i = x_i$ for $i \neq j$ and $x'_j = D$, is not produced by any s-state machine.*

The hypothesis reflects our intuition that at least for some strings y, a string of the form yC^{∞} contains less information than a string of the form $yC^{l}D$. If Kolmogorov complexity is a good measure of information contents then this difference should be reflected in the number of states in the machine. We are not aware of any result with respect to Kolmogorov complexity which does not follow by a counting argument. However, counting arguments do not seem to work here so we believe new techniques have to be developed for proving this hypothesis. We conjecture that a stronger assertion is true, namely, that in (iii) the string x' cannot be produced by any machine with $s + g(s)$ states for some slowly increasing function $g(s)$. However, for our purpose, it suffices to consider only one additional state for the difference between watching and not watching the opponent. Stated more directly, we rely on an extended hypothesis as follows.

The Extended Hypothesis. *Under the conditions of the Hypothesis, the string x (but not x') is produced by an s-state machine T^* that also watches the opponent.*

Recall that the payoff to both players when they play (C, C) in the one-shot game is 3 and the goal is to achieve (in equilibrium) an average payoff of approximately 3 per round.

Theorem 5.2. *Assuming the extended hypothesis, if $s = o(N/\log N)$ then for every $\varepsilon > 0$, there is an N_0 such that for all $N \geq N_0$, an average payoff of at least $3 - \varepsilon$ can be obtained in equilibrium in the N-round game, played by s-state Turing machines (with no limit on space).*

Proof. Obviously, $s = o(N/\log N)$ implies $\log f(s) = o(N)$. Let T^* be an s-state machine the produces a string x as stated in the extended hypothesis. Suppose both players use T^* for playing the game. Thus, in every stage they play either (C, C) or (D, D). But the number of D's is $o(N)$, hence the average payoff per stage tends to 3 when N tends to infinity. This pair of choices is in equilibrium since the only profitable deviation is to play according to x, except that the last C is replaced by a D (see Proposition 4.5). The extended hypothesis states that such a deviation cannot be implemented by an s-state machine.

Remark 5.3. It is interesting to compare Theorem 5.2 to Neyman's results (Propositions 2.2, 2.4). For the comparison, recall that automata have no memory, so in a certain sense they have to use some of their states as memory. Thus, they need at least N states just for counting to N. It is therefore not surprising that even $(N-1)$-state automata can play steady (C, C) in equilibrium. On the other hand, a Turing machine with only two internal states can count to any N, when given the number N in binary as input (Proposition 4.7). We note that the results of the present section can be easily adapted to a model in which the number N is also given to the machine as input. It is obvious that with N states, both automata and Turing machines can play only the steady (D, D) in equilibrium relative to deterministic choices of machines. The proof of this claim is as follows. Given any pair of machines (M_1, M_2), each player i can design an N-state machine M_i' that plays precisely what M_i would play against M_{3-i} (regardless of what $3-i$ actually plays) and yet deviate in the critical round if there is one. Consider a game where players have to choose specific machines rather than probability distributions over machines. If the machines are automata then a restriction to $N-1$ states allows steady (C, C) in equilibrium, but N states leave only the steady (D, D) in equilibrium. Thus, in the context of deterministic choice of automata, the issue is precisely the ability to count to N; this amounts to sufficient "memory" in the form of states. In the case of Turing machines, memory is not the issue. The equilibrium is based on complexity of computation. The machines have unlimited space but they play strategies that waste their computational power so that with the remaining power they cannot count. We have not yet pursued the case of probabilistic choices of (deterministic) machines.

6. Uniform deterministic machines with limited number of states

In the tradition of theoretical computer science a Turing machine is perceived as a "uniform" tool. Here this corresponds to machines that can play a game of *any* number of rounds N which is given to the machine as input. Thus we are now interested in a game as follows.

Definition 6.1. We denote by $\Gamma(s_1, s_2)$ a game where the players have to choose Turing machines T_1, T_2 with no more than s_1 and s_2 states, respectively. The machines then play an N-round prisoners' dilemma game, where the number N is written in binary on the input tapes of the machines.

We discuss here the symmetric case, that is, $s_1 = s_2 = s$. The asymmetric case is a trivial extension of the symmetric one (see the conclusion of this paper). Notice that neither have we defined a payoff function for the game $\Gamma(s_1, s_2)$ nor have we specified the mechanism by which the number of rounds N is selected. One can think of a strong notion of equilibrium as follows.

Definition 6.2. A pair of machines (T_1, T_2) for $\Gamma(s_1, s_2)$ is in *strong equilibrium* if for any other allowable machine T_i' for player i $(i = 1, 2)$ there does *not* exist a number N (or, more weakly, there is an N_o such that there does *not* exist a number $N > N_o$) such that in the N-round game i strictly prefers T_i' over T_i, given that the opponent plays T_{3-i}.

Unfortunately, as we see in the following proposition, the notion of strong equilibrium is "too strong". This is true because machines may interpret their inputs in various ways.

Proposition 6.3. *Suppose* T_1 *and* T_2 *are deterministic machines for playing the game* $\Gamma(s,s)$ *and suppose that, except possibly for finitely many values of* N*, the play of these machines in the* N*-round game is different from the steady* (D,D)*. Under these conditions, for* s *sufficiently large, for at least one player* i*, there exists an* s*-state machine* T_i' *such that for infinitely many values of* N*,* i *prefers* T_i' *over* T_i *when his opponent plays* T_{3-i}*.*

Proof. Consider a "universal" machine T_i', for player i, that acts as follows. For input N sufficiently large, T_i' reads the $2k$-bit prefix of the input string (where k is some sufficiently large constant whose construction by T_i' will be discussed later) and decodes it as a description of a pair of Turing machines. In most of the cases this would be meaningless and we allow T_i' to play arbitrarily. However, when the $2k$-bit prefix describes a valid pair of machines, (T_1^*, T_2^*), T_i' then simulates the play of (T_1^*, T_2^*) on the input N and discovers a round (if one exists) in which player i can deviate profitably in the sense of the N-round game. When T_i' actually plays the game, it plays like T_i^* would have played against T_{3-i}^* but deviates in the critical round. We now discuss the choice of the number k. First, there are obvious ways to encode the description of a Turing machine in binary. Let us fix such an encoding. Obviously, the size of the code of a machine with s states is $O(s\log s)$. Given the number s, we would like to construct a number k such the length of code of any machine with s states does not exceed k. Obviously, we can construct a number k of the form $k=2^j$ where $j=O(\log s)$ is an integer, and moreover, this effort does not require more than $O(\log s)$ states. Now, if $N \geq 2k$, all the pairs of s-states machines are encoded by some strings with less than $2k$ bits, so in some fixed fraction of the cases (at least 2^{-2k}), T_i' simulates precisely the pair (T_1, T_2). Obviously, for at least one of the players such a machine T_i' is preferred in infinitely many cases. Note that the universal machine needs a only constant number of states (that is, independent of s) for most of its work except for the construction of the number k.

Because of Proposition 6.3, we prefer to work with a weaker notion of equilibrium. We consider a *weighted average* of the payoffs of the N-round games. That is, if the play of the machines results in payoffs $u_i^N(T_1, T_2)$ (for player i) in the N-round game, then in the "grand game" we define payoffs $U_i(T_1, T_2) = \sum_N P_N u_i^N(T_1, T_2)$, where $P_N \geq 0$ and $\sum_N P_N = 1$[2]. We argue that under a certain assumption on the weighting sequence P_N and the extended hypothesis, there exists a pair of deterministic machines in equilibrium yielding high payoffs.

---

[2] It should be noted that this weighting function does *not* turn our game into a kind of an infinitely repeated one. It is well-known that if a game is repeated indefinitely, but after each round there is a probability $\alpha > 0$ of stopping, then despite ending (with probability 1) after a finite number of rounds, this game is equivalent to an infinitely repeated game with discounted payoffs. In our case, the interpretation may be that the players play the N-round game with probability P_N but they always *know* the number of rounds *in advance*.

As argued before, the idea behind the equilibria in this paper is that the machines playing them are so busy with the implementation of the strategy, that they cannot do almost anything else. We now extend the ideas developed in the previous sections to the case of selecting uniform machines, given a *distribution* on the number of rounds. The strategy to be used by the machines is similar to the one described for the case of a fixed N.

In Section 5 we stated an hypothesis about machines with no input. Here we discuss machines with input and we believe an analogous assertion is also true.

The Extended Hypothesis (for machines with input). *For every s, there exist numbers $L = L(s)$ and $N_o = N_o(s)$, and a monotone increasing function $\mu_s(N)$, such that for every $N \geq N_o$, there exists an N-string x with following properties: (i) x is produced by some s-state Turing machine T when given the number in N in binary as input, and T also watches the opponent. (ii) x has the form yC^{N-k} where the length of y less than $L(s)$. (iii) For every $N' \geq N - \mu_s(N)$, a string x' that begins with $yC^{N'-k-1}D$ is not produced by any s-state machine when given the number N as input.*

The intuition here is that there has to be a string y that can be produced only by machines with at least s states and there is one which produces yC^{∞}. Since N is considerably larger than k, it seems there is a y with the following property: if y is produced as a prefix of a string $z \neq yC^{\infty}$ then z and yC^{∞} must disagree on some index less than $N - \mu_s(N)$, for some slowly increasing function $\mu_s(N)$.

Let $H(s)$ denote the set of s-state machines T such that for any input N, while watching the opponent, the machine T plays according to a string of the form yC^{N-k}, where the length of y is less than $L(s)$. Denote the number of such machines by $h(s)$.

Proposition 6.4. *Suppose the input N is drawn from a probability distribution $P = \{P_N : N = 1, 2, \ldots\}$ and denote $P'(s) = \sum_{N \geq N_o(s)} P_N$. Then, under the extended hypothesis for machines with input, for any s, there exists an s-state machine $T \in H(s)$ that satisfies the following: (i) T plays a string of the form yC^{N-k}, watching the opponent. (ii) The probability (induced by the distribution P), that any string $yC^{N'-k-1}D$, where $N' \geq N - \mu_s(N)$ can be played by some s-state machine (given N, not necessarily watching the opponent) is less than $1 - P'(s)/h(s)$.*

Proof. Let us say that machine T is "defeated" by machine T' on the number N if, given N, T produces a string yC^{N-k} (watching the opponent) while T' produces the string $yC^{N'-k-1}D$ with $N' \geq N - \mu_s(N)$. Notice that only the case $N' = N$ corresponds to a profitable deviation. For any $T \in H(s)$ and any input N, let $a(T, N) = 1$ if T is not defeated on N by any s-state machine; otherwise, let $a(T, N) = 0$. By the extended hypothesis for machines with input, for every $N \geq N_o(s)$,

$$\sum_{T \in H(s)} a(T, N) \geq 1.$$

It follows that

$$\sum_N P_N \sum_{T \in H(s)} a(T,N) \geq P'(s),$$

so that

$$\frac{1}{h(s)} \sum_{T \in H(s)} \sum_N P_N\, a(T,N) \geq \frac{P'(s)}{h(s)}.$$

The left-hand side of the latter inequality is the average, taken over all $T \in H(s)$, of the probability that the output of T is not defeated on N by any s-state machine. It follows that there is at least one such machine for which this probability is at least $P'(s)/h(s)$.

Proposition 6.4 suggests conditions on the distribution P and the number of states s, under which good equilibria exist. The idea is as follows. Suppose both players "agree" to use the machine T described in Proposition 6.4 but one of them is unfaithful and uses another s-state machine T' instead. Consider a distribution P that does not converge too fast to zero. For a deviation in a certain N-round game (against a faithful player) to be profitable, it must occur in the *last* round. The profit then is only 1. However, if the deviation occurs before round $N-\mu_s(N)$ then the unfaithful player incurs a *loss* of at least $2\mu_s(N)-1$, since for at least $\mu_s(N)$ rounds the faithful player would then play D instead of C. The assumption is that $\mu_s(N)$ is increasing. If the probability for large values of N is sufficiently large, the expected profit is negative. For simplicity, we state the result as follows. We first determine the scope of our probability distributions. Given a natural number n and a positive ε, let $\mathcal{P}_{n,\varepsilon}$ denote the set of probability distributions $P=(P_1,P_2,\ldots)$ ($P_i \geq 0$, $\sum P_i = 1$), over the naturals, for which $P(\{N \leq n\}) \leq \varepsilon$.

Theorem 6.5. *Under the extended hypothesis for machines with input, for any number of states s, there exist a natural number $n=n(s)$ and an $\varepsilon=\varepsilon(s)>0$, such that for any probability distribution $P \in \mathcal{P}_{n,\varepsilon}$, there exists a pair of s-state deterministic machines that are in equilibrium in $\Gamma(s,s)$ relative to the distribution P, whose average payoffs per round in the N-round game tend to that of (C,C), as N tends to infinity.*

7. Conclusion

We have discussed mainly the symmetric case of the the prisoners' dilemma game, that is, players have to use machines under the same restrictions. Similar ideas can be used in a more general situation, that is, asymmetric restrictions (not necessarily on numbers of states) and more general games.

Consider first the asymmetric prisoners' dilemma game. More precisely, suppose the players are restricted to s_1 and s_2 states in their machines, respectively, where s_1 and s_2 are not necessarily equal.

The extension is easy. Suppose $s_1 < s_2$. Player 1 uses the same strategy as in the symmetric case, that is, he plays xC^∞ where x is chosen appropriately and also watches player 2. Player 2's strategy depends on the value of s_2. Let q denote the minimum number of states, with which player 2 could play according to xC^∞, except that in the last round he would play D. The high-payoff equilibrium depends on the the value of q. The extended hypothesis essentially states that $q > s_1$. If $s_2 \geq q+1$ then player 2 plays this way and also watches player 1; this is possible since only one additional state is required for watching. If $s_2 \leq q-1$ then player 2 plays exactly as in the symmetric case. The difficult case is $s_2 = q$. Here player 2 can profitably deviate from the strategy of the symmetric case. However, he may have to do that at the expense of not watching player 1. On the other hand, if player 1 is not being watched then he can deviate profitably.

In a general repeated game situation the long sequence of (C, C) of the prisoners' dilemma game should be replaced by a sequence of payoff pairs that yield high average payoff. For sufficiently large s and N (but s must be small relative to N), any individually rational payoff pair can be approximated by the average payoff in some equilibrium. The players choose a sequence of pairs of moves which is, on the one hand, complex (so that it requires all the states of the machine), and on the other hand yields approximately the desirable payoff. The equilibrium is achieved by prescribing "punishments" to a deviator. These ideas seem fairly standard in the context of repeated games.

We have discussed here restrictions only on the numbers of states, or alternately, lengths of programs. It seems that similar ideas should work when other computational resources are limited. For example, one might consider a game where players can use any number of machines of a fixed type where the total number of machine operations between rounds is limited. Then an equilibrium is likely to exist, based on a string whose computation requires the maximum number allowed. However, even if such equilibria can be shown to exist, it would probably be very hard to construct one and prove that it is an equilibrium.

References

[HU] J. E. Hopcroft and J. D. Ullman, *Introduction to automata theory, languages, and computation*, Addison-Wesley, Reading, Mass., 1979

[K] A. N. Kolmogorov, "Three approaches for defining the information quantity", *Prob. Inform. Trans.*, 1 (1965) 1-7.

[N] A. Neyman, "Bounded complexity justifies cooperation in the finitely repeated prisoners' dilemma", manuscript, February 1985.

[Ru] A. Rubinstein, "Finite automata play -- The repeated prisoner's dilemma", Research Report No. 149, Department of Economics, The Hebrew University, Jerusalem, Israel, 1985.

Appendix: The mechanism

In this appendix we specify the mechanism by which our machines play the game. Our machines are standard Turing machines with the following features: (i) a read/write *work tape*, (ii) a read-only *input tape*, (iii) a *play* device which can be set to either C or D and also can be read, and (iv) an *eye* which can read either C, D or #. Before round 1 the eye sees #. The game is administered by a "referee". The referee first writes into the input tapes of the participating machines the number N of rounds to be played (in binary encoding). After round i the "referee" shows each "eye" what was "played" by the other machine. The machine then has to specify its choice of C or D for the following round. The machine can be in one of a *finite* number of internal states including two special states: PAUSE and GO. The work of each machine is determined by a "next move" function δ. The arguments of the function δ are the following: (i) the current state (except for PAUSE), (ii) the symbols currently being scanned on the work and input tapes, the symbol currently seen by the eye, and the symbol shown on the play device (before round 1 this play device shows #). The function δ then gives the next state (possibly PAUSE), a direction command (left, right, or stay) for moving the head of the work tape or reading the next symbol on the input tape, a symbol to be written on the work tape and a "play" (i.e., C or D). After the state of the machine is changed to GO, the latter is supposed to enter the state PAUSE, with either C or D on the play device, after a finite number of steps. The "referee" acts as follows. When both machines are in their respective PAUSE states, the referee reads the play devices and shows the contents to each other's eye. He then changes the states of both machines to their respective GO states.

A probabilistic machine is almost the same as a deterministic one with the addition of a "coin". This is another device that at any time reads either 0 or 1 with probability ½ each, independently of other parts of the system and the history of the run. This reading is also an argument of the next-move function of the machine.

Note that we do not assume any time limit. So, if a player uses a machine that may not enter its PAUSE state is a certain situation, it may be impossible to prove that he is using an illegal machine even though the game will not proceed.