Implementing SPE of Repeated Games with Finite Automata

by

Dale O. Stahl

Malcolm Forsman Centennial Professor
Department of Economics
University of Texas at Austin

stahl@eco.utexas.edu

April 2015

ABSTRACT

The game theory literature has explored the implementability of subgame-perfect equilibrium outcomes of infinitely-repeated games using finite automata and found limited results. This note shows by construction that if there is a finite number of vertices of the set of individually rational payoffs, then there is a finite automaton that can exactly implement any subgame perfect equilibrium.

Keywords: finite automata, repeated games, subgame perfection

JEL Classification: C61, C72

Beginning with Rubinstein (1986), the game theory literature has explored the implementability of subgame-perfect equilibrium outcomes of infinitely-repeated games using finite automata. Since it is difficult to imagine how an equilibrium could become common knowledge unless it can be unambiguously specified with a finite number of words and symbols in some shared language, restricting strategies to those that can be implemented by finite automata seems eminently reasonable. However, it turned out that almost all individually rational outcomes can be implemented approximately with finite automata [Kalai and Stanford, 1988]. On the other hand, exact implementation might require infinite automata. Subsequent research characterizing the set of all subgame-perfect equilibrium outcomes for infinitely-repeated games with a common discount factor [Abreu, Pearce and Stachetti, 1990], revealed the potentially infinite complexity of equilibrium outcomes. In this note, we provide a sufficient and non-vacuous condition such that any subgame-perfect equilibrium outcome can be exactly implemented using finite automata.

Let G^δ denote an N-player infinitely-repeated game with perfect monitoring and public randomization, in which the stage game is $G \equiv \{N, (A_i)_{i\in N}, (u_i)_{i\in N}\}$, and all players have a common discount factor $\delta < 1$. Let V* denote the set of all normalized pure-strategy subgame-perfect equilibrium payoffs of G^δ. Abreu and Sannikov (2014) show that when N = 2, V* has a finite number of vertices. They also provide an efficient algorithm for finding the vertices. For N > 2, it remains an open question whether V* has a finite number of vertices. We demonstrate that if V* has a finite number of vertices, every $v \in V^*$ can be implemented by a finite automaton with public randomization, and we provide a complete specification for such a finite automaton. As a corollary, when N = 2, every $v \in V^*$ can be implemented by a finite automaton with public randomization.

Assume V* has a finite number (M) of vertices, and let E denote the set of vertices. It is convenient to index these vertices by e^k, for k = 1, …, M. Because V* is a convex set, for any $v \in V^*$, there is a vector of M non-negative coefficients, $\lambda(v)$, such that $\sum_k \lambda_k(v) = 1$ and $\sum_k \lambda_k(v)e^k = v$. Therefore, a public randomization device can generate any $v \in V^*$ from just the vertices using $\lambda(v)$ as the probability weights on the vertices.

Let $\underline{v}_i$ denote the minimum payoff to player i in V*, and let $e^{p(i)} \in E$ denote a vertex that gives player i this minimum payoff. In other words, p(i) indexes the punishment vertex for player i. By definition of V*, for each $v \in V^*$, there exists an action profile $a(v) \in A \equiv X_i A_i$, and a continuation payoff vector $r(v) \in V^*$such that for all $i \in N$,

$$v_i = (1-\delta)u_i[a(v)] + \delta r_i(v) \geq (1-\delta)u_i[a_i, a_{-i}(v)] + \delta\underline{v}_i \text{ for all } a_i \in A_i. \qquad (1)$$

Since $r(v) \in V^*$, by the previous paragraph, there is a vector $\lambda[r(v)]$ such that $\sum_k \lambda_k[r(v)] = 1$ and $\sum_k \lambda_k[r(v)]e^k = r(v)$. Therefore, to generate an expected payoff vector of $v \in V^*$, all we need to do is specify current action a(v) followed in the next stage, in the event of no single-player deviation from a(v), by a public randomization over E using $\lambda[r(v)]$ as the probability weights on the vertices; and in the event of a single-player deviation from a(v), say by player j, the next stage will implement $e^{p(j)}$.

Applying the preceding algorithm to the vertices, for each $e \in E$, there is an action profile a(e), and a continuation payoff vector r(e) such that for all $i \in N$,

$$e_i = (1-\delta)u_i[a(e)] + \delta r_i(e) \geq (1-\delta)u_i[a_i, a_{-i}(e)] + \delta\underline{v}_i \text{ for all } a_i \in A_i. \qquad (2)$$

It is now a simple matter to construct a finite automaton that generates a vector $v \in V^*$. First, we create a state for each vertex. In Figure 1, these states are represented by solid circles labelled e^k for k = 1,…, M. Exiting each of these states we put N+1 directed arrows. One arrow will be followed if there is no single-player deviation from $a(e^k)$, and that arrow goes to a public randomization device with probabilities $\{\lambda_i[r(e^k)], i = 1,…,M\}$ on its exiting arrows. In Figure 1, public randomization devices are represented by small solid squares. To avoid depicting arrows looping back to the initial states, Figure 1 shows arrows going from the squares to dashed circles labeled e^k; the dashed circle signifies that it is not a new state but a duplicate depiction of the initial vertex state.

The other arrows exiting the initial states (solid circles) designate the path to be followed if there is a single-player deviation by player $j \in N$, and it goes to the initial state corresponding to $e^{p(j)}$, again depicted as a dashed circle in Figure 1 since it is not a new state but a duplicate depiction of the initial vertex state. While Figure 1 depicts the $N = 2$ case, the extension to $N > 2$ should be obvious: just add more dashed circles for $e^{p(j)}$, $j = 3,\dots,N$.

The large dashed box in Figure 1 illustrates the part of the automaton specified so far. Notice that if we designate state e^k as the starting state, we would have a finite automaton that generates expected payoff vector e^k. Thus, by appropriate choice of the starting state, the automaton depicted by the large dashed box is sufficient to generate all the vertices of V*. As usual simultaneous deviations can be ignored.

To complete the construction we precede the M initial vertex states by a public randomization device with arrows to each of the vertex states. To generate payoff vector $v \in V^*$, we assign probabilities r(v) to its exiting arrows. Thus, we have completely specified a finite automaton with public randomization that generates an arbitrary expected payoff vector $v \in V^*$.

Note that the finite automata displayed in Figure 1 has M+N states $\{e^k, e^{p(j)}$, $k = 1,\dots M$, $j = 1,\dots,N$. . Abreu and Sannikov (2014) show that when $N = 2$, $M \leq 3$ times the number of action profiles. Thus, for a two player J×H normal form game, any $v \in V^*$ can be implemented by a finite automata with $M \leq 3JH$ states.

Since the individually rational set of normalized payoffs, denoted V^0, has a finite number of vertices, it immediately follows that the Folk Theorem can be implemented by finite automata with public randomization. That is, for any $v \in V^0$, and any $\varepsilon > 0$, there is a $\delta^* < 1$, such that for all $\delta \in [\delta^*, 1)$, there is a $v' \in V^*$ with $|v - v'| < \varepsilon$ and v' can be implemented by a finite automaton as specified above.

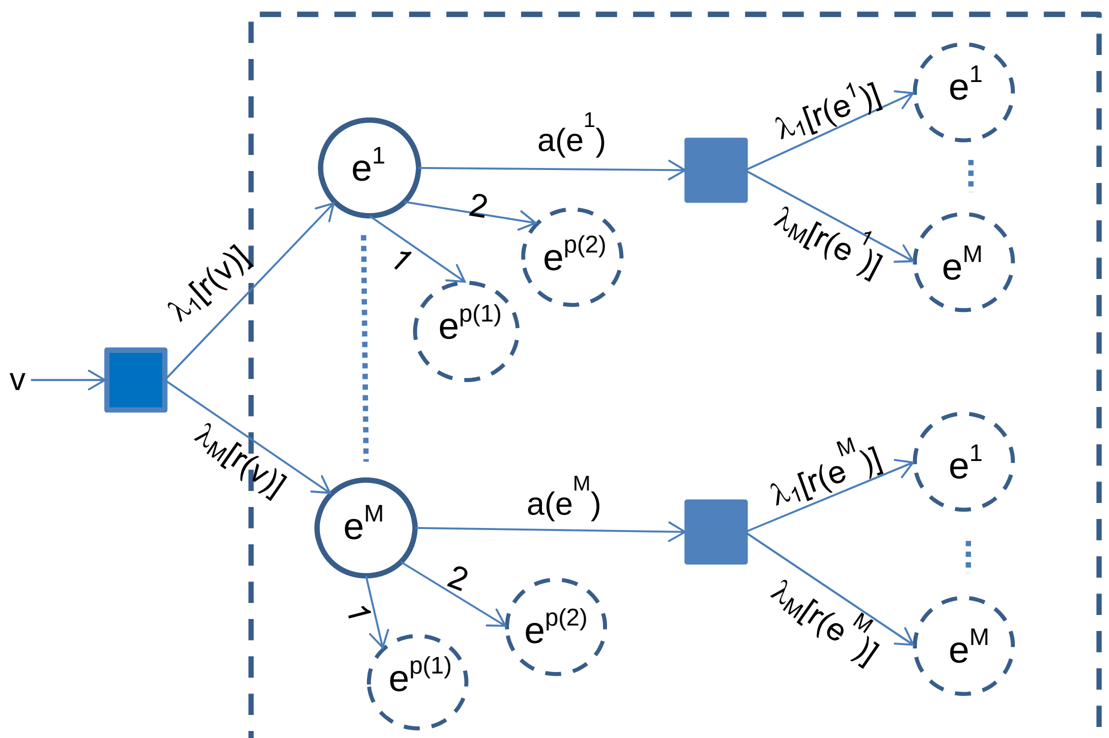
When V* does not have a finite number of vertices, we can still specify a finite automaton that approximately implements any $v \in V^*$ by using M extremal points of V* whose convex hull, V^M, approximates V*. Define ε^M to be the maximal distance between V* and V^M. W.l.o.g. assume $\varepsilon^M \to 0$ as $M \to \infty$. By "approximate implementation" we mean that for every $v \in V^M$ there exists an action profile $a(v) \in A$ and a continuation payoff vector $r(v) \in V^M$, such that for all $i \in N$,

$$v_i = (1-\delta)u_i[a(v)] + \delta r_i(v) \geq (1-\delta)u_i[a_i, a_{-i}(v)] + \delta \underline{v}_i - 2\varepsilon^M \text{ for all } a_i \in A_i. \qquad (3)$$

Of course, to improve the approximation it would be very helpful to have the N punishment vertices, $\{e^{p(i)}, i \in N\}$, in V^M.

The assumption of a common discount factor is crucial for exact implementation. Chen (2007) has shown that with unequal discount factors V* may not have a finite number of vertices. However, in such cases we can still obtain approximate implementation with finite automata as shown above.

Figure 1. A Generic Finite Automaton that Generates v ∈ V*.



References

Abreu, D., D. Pearce and E. Stachetti (1990). Toward a theory of discounted repeated games with imperfect monitoring, Econometrica, 58, 1041–1063.

Abreu, D., and Y. Sannikov (2014). An Algorithm for Two-player Repeated Games with Perfect Monitoring, Theoretical Economics, 9, 313–338.

Chen, B. (2007). The Pareto Frontier of a Finitely Repeated Game with Unequal Discounting, Economics Letters, 94, 177-184.

Kalai, E., and W. Stanford (1988). Finite Rationality and Interpersonal Complexity in Repeated Games, Econometrica, 56, 397-410.

Rubinstein, A. (1986). Finite Automata Play the Repeated Prisoner's Dilemma, Journal of Economic Theory, 39, 83-96.