

**Algebraic Complexity of
Strategy-implementing
Semiautomatons for
Repeated-play Games**

by
Mark R. Johnson

Working Paper No. 259

Working Paper Series
Department of Economics, Finance and Legal Studies
College of Commerce and Business Administration
200 Alston Hall
The University of Alabama
Tuscaloosa, Alabama 35487-0224
U. S. A.
Fax: (205) 348-0590

Algebraic Complexity of Strategy-implementing Semiautomatons for Repeated-play Games

Mark R. Johnson
Department of Economics, Finance and Legal Studies
The University of Alabama
Tuscaloosa, Alabama 35487
U.S.A.

Phone: (205) 348-8979
e-mail: mjohnson@alston.cba.ua.edu

Abstract

The standard measure of automaton complexity used in game theory is the number of states in the minimal automaton. This paper applies classical algebraic complexity measures, based on the Krohn-Rhodes Decomposition Theorem to automata used for strategy implementation in repeated-play games. Results demonstrate that state-based complexity is insufficiently fine to distinguish between automatons with the same number of states and different algebraic complexities. More important, the state-based measure is not monotonic in algebraic complexity, viz., there are automatons with three states, that are less complex than some two-states automatons. Further, the number of states in the automaton does not reveal the important properties of the automaton, such as the ability to count or detect differences in the sequence of play. These differences are reflected in algebraic complexity. When applied to the repeated-play prisoner's dilemma game, algebraic complexity identifies uniquely the "grim trigger" strategy implementing automaton as the simplest Nash equilibrium supporting the cooperative outcome.

KEYWORDS: Game Theory, Algebraic structure, Complexity, Automatons

August 23, 1994

Revised: October 14, 1995

NOTE: The results in this paper are preliminary materials circulated to stimulate discussion and critical comment. Reference to this paper in publication should be cleared with the author to protect the tentative character of this presentation.

*Richard Dean, Boris Schein, Ed Schlee, Daniel Arce M., Pam Turner Green, Patricia Rudolph, Martin Evans and William Stanford each gave valuable comments at various stages of this paper. I thank them. This paper benefited from comments made by the participants of the *Seventh World Congress of the Econometric Society*. This research was supported by the College of Commerce and Business Administration, University of Alabama. I am responsible for all errors.

1. INTRODUCTION

With increasing frequency automata theory is being used to address economic issues. The most active applications area is to strategy implementation in repeated-play games. Among the issues that arise in repeated-play games, one question of prime interest is the “complexity” of the automata used to implement specific strategies. For example, as Neyman (1985), Rubinstein (1986) and Abreu and Rubinstein (1988) have shown, incorporating complexity costs as a factor in strategy selection can shrink the set of Nash equilibrium outcomes. The results presented here concern the measurement of automaton complexity by means of “algebraic complexity” —a measure widely studied by automata theorists, but which has received limited attention by economists.¹

At issue in all automaton complexity measures is the question of what features the complexity measure should capture. Abreu and Rubinstein make suggestions about factors that make the “cost” of complexity an economically relevant matter, principally the cost of maintaining states. Others have linked complexity to memory or an ability to count (e.g., Neyman (1985)). Banks and Sundaram (1990) recognize the “state-maintenance cost” and argue for additional costs reflecting the “monitoring” of transitions. At the core of each of these suggestions seems to be an attempt to capture what Kalai and Stanford call the “computing power” of the automaton (Kalai and Stanford (1988, p. 398)).²

¹ Futia (1977) and Gottinger (1978, 1983) use algebraic complexity to address, respectively, individual and collective choice rules. Johnson (1990, 1994, 1995) examines the link between consistency axioms, like WARP and IP, and algebraic complexity. Turnbull (1994) uses related algebraic techniques to address organizational structure and efficiency. Stanford (1987) demonstrates the existence of groups in strategy-implementing automata and Rubinstein (1986) and Kalai (1990) mention the possibility of using algebraic complexity to analyze repeated-play games.

² In addition, there are a number of papers concerning related “complexity” based issues in game theory. Among these issues are (i) the complexity of computing the best response (e.g., Gilboa (1988), Ben-Porath (1990), Papadimitriou (1992)) and (ii) the complexity of the strategy itself (e.g., Lipman and Srivastava (1990)). Further alternative complexity measures are advanced in Page (1994). The issue addressed here concerns solely the complexity of an automaton implementing a strategy in a repeated-play game.

In applications such as finding the “simplest” automaton implementing an equilibrium strategy (e.g., Baron and Kalai (1993)), the scale may be critically important. Typically, objective functions are quite “flat” close to their optima with the result that “fineness” and order on the measurement scale are both important. For applications where the scale may be relevant, several different automaton complexity measures have been suggested. Most notable among the means for comparing complexities considered by economists are (1) the number of states in the automaton implementing the strategy,³ (2) the number of states in the minimal automaton implementing the strategy (Kalai and Stanford (1988))⁴ and (3) a criterion depending on both the number of states and the number of edges emanating from each state (Banks and Sundaram (1990)).⁵

Despite their widespread use, none of these measures fully captures the relevant aspects of what most economists mean when they use the term “complexity.”⁶ Specifically, there are two different activities a strategy-implementing automaton must accomplish; (1) the automaton must monitor the strategic choices of the other player(s) and (2) it must generate appropriate responses given the observations. For example, a strategy rule in a two-player game might be; if my opponent takes action “a” first and

³ This method might be the most commonly used complexity measure. Among those who have used this measure are Rubinstein (1986), Abreu and Rubinstein (1988), Binmore and Samuelson (1992). A more complete, though still partial, listing is available in Kalai (1990).

⁴ It should be clear that automata can have “extraneous” states. The Myhill-Nerode Theorem assures that for every automaton, there is a minimum-state automaton that does not contain extraneous states and that minimal automaton is unique up to isomorphism.

⁵ Banks and Sundaram actually compare several different “complexity” measures and demonstrate that different rankings obtain depending on the method used to scale the automata. Their principal measure, the increasing complexity criterion (ICC), is based on the *number* of states and the *number* of transitions, there by missing the *pattern* of the transitions and the *structure* of the system defined by the automaton.

⁶ This point has been noted in the literature. In speaking of the number-of-states measure, Abreu and Rubinstein (1988), for example, observe “The measure neglects the desire of players to simplify their calculations during the course of play. This is reflected in the measure being independent of the specification of the output and transition functions.”

then action “b” respond with action “1” but if they take action “b” first and then take action “a” respond with action “5.” To correctly reflect the “complexity” of the automaton it is necessary to identify the “mathematical power” required to implement the rule. In the above discussion, the monitoring power of the automaton must be able to distinguish between two different sequences of stimuli; (1) “a” first and then “b” or (2) “b” first and then “a.” Systems that can distinguish between these two sequences can not be abelian.⁷ Similarly, a strategy rule could require that the automaton “count” the number of times an action is taken and to respond in a different way depending on the number of times the specified action is taken. This requirement also impacts the required power of the mathematical system defined by the automaton. In terms of the responses generated, some automata can “stand pat” while other automata can return to previously visited states and a special class of automata can do both. These properties; detecting sequence of action, counting and the range of response patterns, seem to reflect more accurately the “complexity” of the automaton than any properties deriving from the number of states in the automaton. However, each of these properties is reflected directly in the algebraic structure of the mathematical system defined by the automaton and these properties can be used with classical algebraic complexity measures to rank automata.

As generally used, the term “complexity” combines two distinct goals (1) identification and (2) comparison. In mathematical applications, the goal of a complexity is to distinguish among all non-isomorphic members of a class. Generally this goal is addressed by identifying “basic” or “fundamental” (often prime) elements of a system that, when combined with a specified operation, are able to regenerate a system at least as

⁷ A non-automata theoretic example of this same phenomenon is to consider directions on a city block grid. Walking left for one block and then one block to the right will not take you to the same place as going one block to the right and then one block to the left. The left and right actions, in this case, are not abelian.

rich as the original system.⁸ Typically, the elements used to distinguish among the members of a class, also, can be used to rank structures by comparing the number and properties of the elements used for identification. One of the well known examples of an algebraic complexity measure is the Krohn-Rhodes complexity for automata. Since these complexity measures, at least crudely, reflect the mathematical power of automata, they seem a natural starting point for meeting Kalai and Stanford's goal.

Krohn-Rhodes complexity is based on the fact that every automaton defines a transformation semigroup and that every transformation semigroup has a decomposition into a wreath product of prime transformation groups and aperiodic transformation semigroups (semigroups that contain no subgroups).⁹ Given an automaton, the Krohn-Rhodes complexity is determined by the number groups and aperiodic semigroups in the decomposition (Krohn and Rhodes (1962, 1965), Tilson (1971)). Among other issues, the contrast between the algebraic representation adopted by Krohn-Rhodes and the pictorial representation adopted by most economists raises the question about what are the economically relevant "basic" elements of an automaton.¹⁰ To date, the pictorial representations of automata common in economics has lead to a treatment of complexity based on graph-theoretic terms and concepts. From the pictorial perspective, the states and transitions of an automaton become the vertices and edges of a graph representing the

⁸ In many applications, the operations might be addition or subtraction. For the structures arising from strategy implementing automata, the operations are lattice operations such as the meet or join and the operation of matrix multiplication.

⁹ More precisely, the wreath product of aperiodic semigroups and prime groups is a semigroup covering for the original semigroup (see Krohn-Rhodes (1962, 1965)). Groups are semigroups that possess an identity element and inverse elements. Thus, every group is a semigroup but only selected semigroups are groups. Aperiodic semigroups are semigroups that do not contain subgroups. These structures are discussed further in section 3.3 and the complexity results are addressed in section 4. Saari (1995) discusses links between wreath products, group structure and several examples from choice theory, statistics, probability, etc.

¹⁰ There is a sense in which this statement is over-drawn. Many introductions to automata theory begin by representing the mapping information as a directed graph and use the graph to define the algebra that is used to address the question of complexity. Precisely this approach is taken by Eilenberg (1974, 1976).

automaton. Since graphs are defined by their vertices and edges, these structures are the appropriate basic elements. When viewed as an algebra, the states and transitions define a transformation semigroup for which the subgroups and aperiodic semigroups are the natural basis for a complexity measure. Initially, this approach may seem abstract but it concisely captures the “power” of the mathematical system defined by the automaton. The algebraic approach also has the advantage of providing more intuitive interpretations of some issues than the pictorial approaches.¹¹

In addition to the question of what are the appropriate defining elements for strategy-implementing automata, the algebraic approach focuses attention on several features of the automata.

- Algebraic complexity uniquely identifies the automaton implementing the “grim trigger” strategy as the simplest automaton supporting the cooperative outcome as a Nash equilibrium. When ranked by algebraic complexity only one automaton is both simpler and implements a Nash equilibrium. That automaton implements the “defect” strategy.
- There is no elementary relationship between algebraic complexity and other measures of complexity based on cardinal aspects of pictorial automaton representations (i.e., the *number* of vertices or edges). In particular, (1) there are “simple” automata of all sizes,¹² (2) automata with the same number of states can have different algebraic complexities, (3) automata with a “small” number of states can be “more complex” than automata with a “large” number of states.

¹¹ For example, the algebraic property of idempotence (defined below) is directly linked to the ability to count. Idempotent algebras can not count while non-idempotent algebras can count. Similar comparisons can be made for the abelian property and the existence of identities and inverse elements. In contrast, a compelling basis for comparing the relative complexity of the vertices and edges of an automaton’s graph representation seems lacking.

¹² For this point, “simple” means automata whose associated semigroups do not contain subgroups.

- Since the results below identify several different mathematical structures with clear differences in their power (and, hence, their “complexity”), it is necessary to consider what level of “fineness” is appropriate for economic applications and, further, to consider whether the different types of basic or prime elements should impose different economic costs. Further, there is an open question about the economically appropriate way to measure the complexity of groups.¹³
- Since the algebra of interest is derived from the recognizable “words” defined by the automaton, it becomes clear that “language” is a critical feature of the automaton and that algebras can be useful in understanding the role of language in strategy implementation.¹⁴
- Algebraic complexity reveals that some strategy implementing automatons can have higher algebraic complexities than are allowed by commonly accepted economic consistency axioms.¹⁵

This paper is structured as follows. Section 2 uses the grim-trigger strategy-implementing automaton to introduce the basic intuition of algebraic automata theory. A key feature of this section is that transformation matrices are used to represent the state transitions induced by an opponent’s actions and that the algebra defined by the multiplicative closure of these matrices is the algebra defined by the automaton. Section 3 begins with a formal treatment of the basic finite-state machine. Given the automaton structure, a standard treatment of strategy-implementing automatons in repeated-play

¹³ Certain classes of groups (e.g., abelian groups) admit a number of different representations while other classes of groups (e.g., non-abelian groups) pose difficult and generally unresolved representation problems. Each representation will reflect different properties of the group structure. In section 4, two complexity scales for groups are offered and their differences are demonstrated in section 5.

¹⁴ Rubinstein (1991, page 921) noted that “although language plays a crucial role in resolving conflicts, game theory has so far been unable to capture this role.”

¹⁵ See Johnson (1994) for the algebraic complexities imposed on choice functions by different consistency axioms.

games is presented. Following these basics, the definitions for algebraic automata theory are provided. Section 4 provides a discussion of the decomposition theorem for semigroups and related issues concerning algebraic complexity. Section 5 begins by considering the one- and two-state semiautomatons implementing strategies in repeated-play two-by-two games. In the analysis, the algebraic structures and complexities for these machines are determined. Although all of the semiautomatons have one or two states, five different algebras are defined. Since these algebras are so restricted, they define invariants, identifying equivalence classes of the semiautomatons that are unique up to isomorphism. For these cases, any other “complexity” measure is implied once these invariants are known. Further for these small automata, each algebra’s complexity is unambiguously compared to the other four algebras.¹⁶ Following treatment of the one- and two-state machines, “simple” and “complex” three-state machines are presented. Section 5 concludes by addressing “larger” automata to highlight a problem in measuring the complexity of groups and to demonstrate how algebraic techniques can identify subsystem relationships. A few propositions formalizing the relationship between algebraic complexity and other complexity measures are provided in Section 6. Conclusions are provided in section 7.

2. EXAMPLE 1—THE GRIM TRIGGER AUTOMATON

Many investigations of behavior in 2-person repeated-play games focus on the repeated-play prisoner’s-dilemma game. The pay-off matrix for a one-shot prisoner’s dilemma game is presented in table I. In this game, each player has two strategies called “cooperate” and “defect.” To clarify the discussion, player 1’s cooperate and defect strategies are denoted by C and D while those for player 2 are labeled γ and δ . In the

¹⁶ This result is due to the restricted class considered.

repeated-play game, a strategy is a rule determining each player's action in the n^{th} play of the game as a function of the previous $n-1$ decisions by the other player.

To be concrete, consider one of the most well known repeated-play game strategies, the "grim trigger." Player 1 implements the grim trigger strategy by "cooperating" in the first round of the game and in each following round of the game as long as player 2 also plays the strategy "cooperate." If player 2 ever "defects," then player 1 plays "defect" forever. This rule can be implemented by the automaton depicted in figure 1, which, for later reference is called #1. This automaton has two states "C" and "D" for the "cooperate" and "defect" strategies of player 1. The arrow entering state C from the left indicates that the automaton starts in state C and that player 1 selects "cooperate" in the first play of the game. The choices of player 2 are indicated by the γ 's and δ 's. If player 2 selects γ ("cooperate") in a round k of the game while player 1's automaton is in state C then, the automaton will stay in state C and player 1 will play cooperate in round $k+1$ of the game. This is indicated by the arrow labeled γ beginning and ending at state C. Alternatively, if player 2 selects δ ("defect") in round k , while the automaton is in state C and player 1 is cooperating, then the automaton will switch to state D and player 1 will play "defect" in the $k+1$ play of the game. This transition is indicated by the arrow labeled δ beginning in state C and ending at state D. Once the automaton gets to state D, it will always stay in state D (indicated by the arrow labeled γ, δ beginning and ending at D). Because the automaton can be stopped in either state C or D, these states are called terminal states.¹⁷

¹⁷ Later an automaton lacking any initial or terminal state designations will be identified as a semiautomaton.

This same information about the repeated-play strategy also can be described by representing the current state as a zero-one, 1×2 row vector and the mappings γ and δ by a pair of 2×2 transformation matrices. In general, the state k of any n -state automaton can be represented by an n -vector with a 1 in the k^{th} column position and 0's in the other positions. For the two-state automaton in figure 1, state C can be represented by the

		Player 2	
		γ	δ
Player 1	C	(5,5)	(0,6)
	D	(6,0)	(1,1)

TABLE I: Prisoner's dilemma game pay-off matrix.

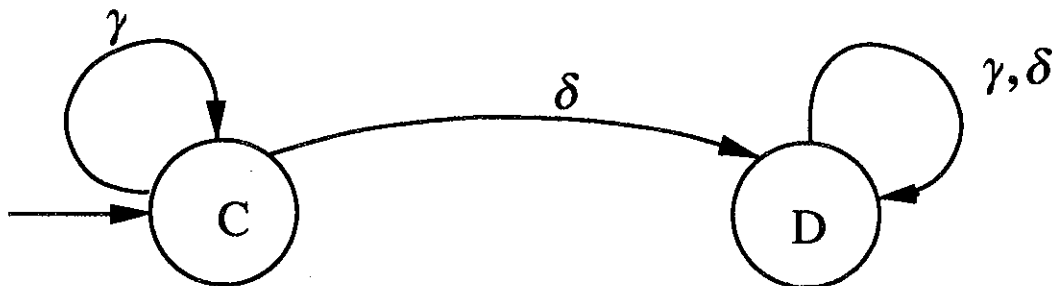


FIGURE 1: #1—Grim trigger automaton.

vector $(1,0)$ and state D by $(0,1)$. The mappings γ and δ are captured by 2×2 transformation matrices which have a 1 in the i^{th} row and j^{th} column indicating that, if the automaton is in state i , it should transfer to state j at the input γ or δ . Thus, the fact that the automaton stays in its original state when the opponent plays γ is captured by the

2×2 identity matrix $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.¹⁸ The fact that the automaton responds to δ by changing from state C to state D or by staying at state D (depending on the original state) is captured by the matrix $\begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$. Note that the resulting state of an automaton originally in state C when the opponent plays δ is captured by the product $(1,0)\begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} = (0,1)$.

What is significant about this description is that the possible behavior of this automaton is described by all of the possible matrix multiplication products of the matrices $\gamma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and $\delta = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$, that these matrix products are a closed set and the set of matrix products form an algebra called a semigroup under matrix multiplication.¹⁹ The “complexity” of the automaton is determined by the basic elements of the semigroup resulting from the matrix products. For the grim-trigger implementing automaton, the algebra, labeled S_1 for later use, is very simple. The operation table for S_1 can be obtained by taking the multiplicative closure and recording the products as in table II. The products are, $\gamma^2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \gamma$, $\delta^2 = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} = \delta$, and $\gamma\delta = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \delta\gamma = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} = \delta$. Thus, the semigroup S_1 is composed of just the original matrices $\gamma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and $\delta = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$ (because no new elements arise from the matrix multiplications, the set $\{\gamma, \delta\}$ is a closed set under matrix multiplication), and inspection of the operation table reveals that S_1 is both commutative

¹⁸ Because one of the matrices in the algebra for this automaton is the identity matrix, this strategy has the ability to stay in the current state or “stand pat.”

¹⁹ Additionally, this semigroup is precisely the algebra that Stanford (1987) noted sometimes has the structure of a “group.” See section 5.1, examples 3 and 4 for group-containing semigroups.

and idempotent. Later, commutative, idempotent semigroups are identified as semilattices and semilattices are identified as a simple class of algebras.

	γ	δ
γ	γ	δ
δ	δ	δ

TABLE II: S_1 Operation table for grim-trigger semiautomaton's algebra

3. NOTATION AND DEFINITIONS

This section begins with the definitions for automata theory followed by the definitions for the supergames in which the automata operate. Finally, the tools for analyzing the automata—principally transformation semigroups are introduced.

3.1 *Semiautomatons, automatons and languages*

The building block of automata theory is the semiautomaton. A *semiautomaton* is a triple $\mathfrak{M} = (Q, \Sigma, F)$ where Q is a finite set of *states*, Σ is a finite set called an *alphabet* and F is a partial function $F : Q \times \Sigma \rightarrow Q$. The elements $\sigma \in \Sigma$ are the *letters* of Σ . Given a set of letters Σ , the set of all n -tuples $s = (\sigma_1, \dots, \sigma_n)$ where $n \geq 0$ form the set Σ^* . The elements of Σ^* are called *words* and n is the *length* of a word s . Included in the set Σ^* is the *null word* $s = ()$ which has length zero. If $s = (\sigma_1, \dots, \sigma_n)$ and $t = (\tau_1, \dots, \tau_m)$ are two words, their *product* st is a word defined by concatenation, so that $st = (\sigma_1, \dots, \sigma_n, \tau_1, \dots, \tau_m)$. If F is a function then \mathfrak{M} is called *complete*. Given a semiautomaton \mathfrak{M} , $i \in Q$, a distinguished member of the set of states called the *initial state*, and $T \subseteq Q$ a distinguished subset of the machine states called *terminal states*, an *automaton* A is the triple $A = (\mathfrak{M}, i, T)$. The Myhill-Nerode Theorem (see Nerode (1958)) assures that for every automaton A , there is a minimum-number-of-states automaton

called the *minimal automaton* that is unique up to isomorphism.²⁰ A *Moore machine* can be obtained by appending an *output alphabet* Γ and a partial function $\lambda : Q \times \Sigma \rightarrow \Gamma$ which defines an *output module* (Q, λ) . The Moore machine is denoted by $R = (A, \lambda)$.

3.2 Automata implementing strategies in repeated play games

Let $G = \langle S_1, S_2, u_1, u_2 \rangle$ be a two-person game in normal form where S_i is a finite strategy set for player i and $u_i : S_1 \times S_2 \rightarrow R$ is player i 's *payoff function*. A strategy pair is called an *outcome*. Repeating the game G infinitely many times defines a *supergame* in which each period is labeled $t = 1, 2, 3, \dots$. In any period t , the players make simultaneous moves $s_t^i \in S_i$ which become common knowledge. A strategy in the supergame is a sequence of functions $\{\sigma_i^t\}_{t=1}^{\infty}$ where σ_i^t determines player i 's strategy in period t as a function of the previous $t - 1$ outcomes.

In the machine game G_m , player i chooses a Moore machine $R_i = (\bar{A}_i, \bar{\lambda}_i)$ where $\bar{A}_i = (\mathcal{M}_i, q_i^1, T_i)$. Here \mathcal{M}_i is the semiautomaton, q_i^1 is the initial state and T_i is the set of terminal states. The semiautomaton is $\mathcal{M}_i = \langle Q_i, \Sigma_i, \mu_i \rangle$ where Q_i is a finite set of states, Σ_i is the alphabet and $\mu_i : Q_i \times S_j \rightarrow Q_i$ is the transition function.²¹ Notably, for player i , the alphabet Σ_i is the strategy set for player j , $\sigma_j \in S_j$ and the words in Σ_i^* are the sequences of strategies for player j , $s_j = (\sigma_j^1, \dots, \sigma_j^m)$. Given two words $s_j = (\sigma_j^1, \dots, \sigma_j^m)$ and $t_j = (\tau_j^1, \dots, \tau_j^n)$, their product is the concatenation $s_j t_j = (\sigma_j^1, \dots, \sigma_j^m, \tau_j^1, \dots, \tau_j^n)$. Given Q_i and Σ_i , $\bar{\lambda}_i : Q_i \times \Sigma_i \rightarrow Q_i$ is the output function. In the machines below, $\bar{\lambda}_i$ is the

²⁰ The Myhill-Nerode Theorem (see Nerode (1958)) from computer science assures that, for every automaton there is a minimal automaton. The results of Kalai and Stanford (1988) are closely related to this theorem. In a minimal automaton, the operations are on representative elements of equivalence classes of states. The act of minimizing an automaton involves identifying the correct partition of the states in the automaton so that the equivalence classes can be defined. For extended discussion of the techniques for minimizing an automaton and examples of the minimization process see Hopcroft and Ullman (1979) or Holcombe (1982).

²¹ Following the approach of section 2, the transition functions are represented by transformation matrices.

identity function which reports the current state as the next response. For this reason, the output module is not modeled explicitly.

3.3 Semigroups, transformation semigroups and groups

The definitions of binary systems and system properties are provided in terms of an arbitrary non-empty set N , which is used for the domain and range, and a binary operation denoted by (\cdot) . Thus, $\cdot : N \times N \rightarrow N$, and the binary system for N under the operation (\cdot) is denoted by $\langle N; \cdot \rangle$. Algebraic properties defined for all $v_1, v_2, v_3 \in N$ are,

(B-1) Closure: $v_1 \cdot v_2 \in N$.

(B-2) Associative: $v_1 \cdot (v_2 \cdot v_3) = (v_1 \cdot v_2) \cdot v_3$.

(B-3) Commutative: $v_1 \cdot v_2 = v_2 \cdot v_1$.

(B-4) Idempotence: $v_i \cdot v_i = v_i$.

A binary system satisfying (B-1) and (B-2) is called a *semigroup* and a semigroup satisfying (B-3) is called an *abelian semigroup*. A semigroup for which every element satisfies (B-4) is called an *idempotent semigroup*. Special members of the binary systems used in automata theory include,

- (i) An element z of a binary system $T = \langle N; \cdot \rangle$ is a *zero* if $x \cdot z = z \cdot x = z, \forall x \in T$.
- (ii) An element e of a binary system $T = \langle N; \cdot \rangle$ is an *identity* if $t \cdot e = e \cdot t = t, \forall t \in T$.
- (iii) An element t of the binary system with identity $T = \langle N; \cdot \rangle$ is said to have an *inverse* if there exists a unique element t^{-1} such that $t^{-1} \cdot t = t \cdot t^{-1} = e$.

Semigroups may have zeros or identities or both. A semigroup with an identity is called a *monoid*. Semigroups (monoids) with a zero are called *semigroups (monoids) with zero*. Every monoid has at least one element with an inverse—the identity element of the monoid. A monoid in which every non-zero element has an inverse is called a

group. A closed subset of a semigroup for which there is an identity and for which every non-zero element has an inverse is called a *subgroup* of the semigroup. The number of elements in a group or subgroup is called the *order* of the group or subgroup. If the order of the subgroup is one, the subgroup is called *trivial*. A semigroup with only trivial subgroups is called *aperiodic* or *combinatorial*.

Three specific algebras arise from the automata considered in section 5. Two of these algebras are special classes of aperiodic semigroups and the other algebra is a class of group. The most restricted semigroup class considered is that which is idempotent and commutative. These semigroups form a semilattice under the natural partial ordering of the semigroup (see Clifford and Preston (1961)) and are known as a *semilattice*. Slightly less restricted, is the class of semigroups that are just idempotent. Idempotent semigroups are called *bands*. A particular class of band known as a *rectangular band* arises in section 5.1 (Clifford and Preston (1961)).²² The only class of group that arises in section 5 is the *cyclic group* (see for example, Dean (1990)). These groups are abelian. The non-abelian *dihedral group* is referenced in section 6 (see Dean (1990)).

The fundamental semigroups in algebraic automata theory are the transformation semigroup and the action semigroup (see Holcombe (1982)). These semigroups are defined as follows: Let Q be a finite set and let $PF(Q)$ be the monoid of partial functions $Q \rightarrow Q$ under the operation of concatenation. The identity partial function is the unit denoted by 1_Q . A *transformation semigroup* $X = (Q, S)$ is a finite set Q and S a

²² Rectangular bands get their name from the fact that they are isomorphic to the following structure. Let X and Y be any two sets and define multiplication on $S = X \times Y$ as follows; for $x_1, x_2 \in X, y_1, y_2 \in Y$, $(x_1, y_1)(x_2, y_2) = (x_1, y_2)$. If $X \times Y$ is thought of as a rectangular array of points, then $a_1 = (x_1, y_1)$ and $a_2 = (x_2, y_2)$ are opposite vertices of a rectangle and $a_1 a_2 = (x_1, y_2)$ and $a_2 a_1 = (x_2, y_1)$ are the other two vertices of the rectangle.

subsemigroup of $PF(Q)$. The set Q is called the *underlying set* of X and the members of Q are called *states*. The semigroup S is called the *action semigroup* of X and the elements of S are called *transformations* of X . If S contains the identity transformation 1_Q , then X is a *transformation monoid*.

4. ALGEBRAIC COMPLEXITY

Mathematically, the term “complexity” most often is used to identify the type and number of basic components of a system that can be used to regenerate an equally rich system. When the basic elements distinguish between all non-isomorphic members of a class, the complexity is called an *invariant*. Given a set of basic elements, complexity comparisons between different systems are accomplished by comparing the number and type of elements. The most obvious comparisons are where one system is isomorphic to a subsystem of another. Where there is not a subsystem isomorphism, comparison is made by examining the properties of the components.

The classic method for determining the algebraic complexity of a given automaton is (1) minimize the automaton, (2) construct the transformation semigroup for the semiautomaton inside the minimal automaton²³ and (3) identify the Krohn-Rhodes Decomposition of the transformation semigroup.²⁴ This process identifies the basic elements of the machine’s transformation semigroup in terms of a wreath product of prime transformation groups and aperiodic transformation semigroups and, thus, provides a natural measure of the machine’s complexity. Given a specific semigroup and its

²³ The examples presented in section 5 detail the construction of the associated semigroup for several strategy-implementing automata used in some repeated-play games. Holcombe (1982) provides computer code for evaluating the semigroup of a machine with up to five states and nine inputs.

²⁴ The Krohn-Rhodes Theorem assures that every transformation semigroup is covered by a wreath product of transformation groups and aperiodic transformation semigroups. One accepted means for identifying this decomposition is the Holonomy Decomposition (see Holcombe (1982)). Semigroup coverings are also discussed in Holcombe.

decomposition, the most well known complexity measure is the number of prime groups (groups containing no subgroups) in the decomposition. This measure has been axiomatically characterized (Rhodes (1973)). The Krohn-Rhodes Decomposition Theorem, also, is the basis of a more refined complexity measure offered by Tilson (1971). Tilson's measure is a 2-tuple providing information both about the number of elements in the decomposition and the composition of the transformation groups and aperiodic transformation semigroups in the decomposition.²⁵ While neither of these complexity measures is an invariant, both have proven useful in automata theory.

While a useful starting point, there are problems with the Krohn-Rhodes and Tilson complexity measures. First, both are quite crude. In particular, every group contributes equally to the complexity of the product. For example, when simply counting the number of subgroups, abelian and non-abelian groups make the same contribution to the complexity. Second, when applied to automata constrained to meet consistency axioms, the group measure is too coarse to discriminate among classes of choice functions satisfying the different consistency axioms (see Johnson (1994, 1995)). In the section 5 examples, it is seen that a similar lack of adequate resolution occurs for strategy-implementing semiautomatons.

A more targeted approach to measuring semiautomaton complexity is to focus on the action semigroup of the transformation semigroup and to analyze this structure more closely. The foundation for this approach lies in the fact that all the complexity of a transformation semigroup is contained in its action semigroup (Eilenberg (1976, Proposition XII, 1.3)). Thus, identifying the structure of the action semigroup and

²⁵ Given a Tilson 2-tuple, it is possible to determine the number of groups in the decomposition, however, the number of groups alone is not sufficient to determine Tilson's 2-tuple.

keeping track of the number and type of subgroups and aperiodic semigroups allows a fine scaling of semiautomaton complexity to be obtained without directly confronting the wreath product decomposition. In section 5.1, this approach provides invariants for the algebras of one- and two-state semiautomata implementing strategies in repeated-play. There, it is seen that some strategy implementing semiautomatons are “simple” enough to have semilattice representations while others are “complex” enough to contain subgroups.

By looking more finely at the algebraic structure of the automata semigroups, it is possible to identify several classes of automata. First, the simplest class are automata implementing rules that do not depend on either the sequence of play for past actions or on the number of times an action (or sequence of actions) is taken and cannot revisit previously visited states. The algebras of automata implementing rules of this type are abelian, idempotent aperiodic semigroups (i.e., semilattices). Second, slightly more complex, are automata implementing rules that depend on differences in the sequence of actions but cannot count. The algebras of automata implementing rules of this type are non-abelian, idempotent aperiodic semigroups (i.e., bands). Third are automata implementing rules where the responses may or may not depend on the sequence of actions and does depend on the number of times the action (or sequence of actions) are taken but where there is either not an ability to stand pat or where previously visited states cannot be revisited. The algebras of these automata are non-idempotent aperiodic semigroups and they are the first of these automata that can count. Fourth are automata implementing rules where the responses do not depend on the sequence of actions but where there is a stand pat capability and where previously visited states can be revisited. The algebras arising from these automata are abelian groups which, also, have a counting ability. Finally, there are automata implementing rules where the responses depend of the

sequence of play and where there is a stand pat ability and where previously visited states can be revisited. The algebras for these automata are non-abelian groups which can both detect sequence of action and count.²⁶

Comparison of the algebraic structures and basic elements in the systems alone provides essentially the same ranking. Specifically, semilattices are the simplest structure because they are the most restricted class of semigroups and because their prime elements are join-irreducibles (basically, single points), bands are next because they are less restricted than semilattices and because they are a semilattice of rectangular bands (essentially, two-point sets), non-idempotent semigroups are heterogeneous enough that elementary representations are rare and groups are the even more complex because of their capabilities compared to any semigroups. Within the class of groups, abelian groups are generally regarded as simpler than non-abelian groups.²⁷ Because the type of basic elements changes from join-irreducibles to rectangular bands to arbitrary semigroups to classes of groups, the complexity scales need to reflect these differences.

The following definition applies when a subsystem isomorphism exists.

DEFINITION: Given semiautomata $\mathfrak{M}_1, \mathfrak{M}_2$ with action semigroups $S_{\mathfrak{M}_1}, S_{\mathfrak{M}_2}$; if $S_{\mathfrak{M}_1}$ is isomorphic to a subsystem of $S_{\mathfrak{M}_2}$, then \mathfrak{M}_1 is *i-simpler* than \mathfrak{M}_2 denoted as $\mathfrak{M}_2 \geq_i \mathfrak{M}_1$.

When a subsystem isomorphism does not exist, the following scales can be applied. The coarsest scale is group complexity adopted from Krohn and Rhodes (1962,

²⁶ Since idempotent semigroups cannot contain groups, these criteria identify only two classes of groups, abelian groups and non-abelian groups.

²⁷ In part, this agreement comes from the existing theorems on representations for abelian groups and the difficulty of identifying representations for non-abelian groups.

1965). For an action semigroup S of a strategy-implementing semiautomaton \mathfrak{M} , the *group complexity*, denoted by $\gamma(S)$ is the number of non-trivial, prime subgroups in the action semigroup. Because the number-of-subgroups measure is not sensitive to the structure of the groups being counted, an alternative measure based on the matrix representation of the group is offered. Let $\mu(S)$ be the cardinality of the smallest set of matrices generating an isomorphic group.²⁸ This measure identifies cyclic groups as the simplest class of groups.²⁹ For the two specific classes of aperiodic semigroups two different complexity measures are appropriate. The bands that arise satisfy the identity $aba = a$, $\forall a, a \in S$ and, therefore are a semilattice of rectangular bands. For these bands, a coarse *band complexity* measure is the number of rectangular bands in the semilattice; this measure is denoted by $\beta(S)$.³⁰ For semilattices, the *semilattice complexity* measure $\sigma(S)$ is the number of join-irreducible elements in the semilattice (see Schein (1992)).³¹ Because semilattices are a more restricted subclass of semigroups, semilattices are simpler than bands. Applied lexicographically, the measures $\gamma(S)$, $\beta(S)$ and $\sigma(S)$ define an invariant appropriate for the algebras in section 5.1. The measure $\mu(S)$ is used elsewhere in sections 5 and 6.

DEFINITION: Let \mathfrak{M} be a complete semiautomaton with $|Q| \leq 2$ implementing a strategy in a two-by-two game and let $S_{\mathfrak{M}}$ be the action semigroup for the semiautomaton, then the *2-strategy complexity* is the triple $\tilde{c}(\mathfrak{M}) = (\gamma(S_{\mathfrak{M}}), \beta(S_{\mathfrak{M}}), \sigma(S_{\mathfrak{M}}))$.

²⁸ Note that every group can be represented by a group of permutations (see Dean (1966)) and each permutation can be represented by matrix (in the manner introduced in section 2). Isomorphism obtains directly.

²⁹ Every cyclic group is abelian and is generated by a single matrix. Although not employed here, there are several possible refinements of this measure, the most obvious of which is to incorporate the dimension of the generating matrices.

³⁰ A more complete complexity for bands would probably include (1) the maximal rectangular subbands, (2) the structural semilattice and (3) specification of the product rule for the rectangular components. (Boris Schein, private communication).

³¹ Johnson (1994) applied this measure to semigroups meeting different economic consistency axioms.

Given two semiautomatons \mathfrak{M}_1 and \mathfrak{M}_2 with complexities $\tilde{c}(\mathfrak{M}_1)$ and $\tilde{c}(\mathfrak{M}_2)$, their complexities can be compared as follows. The semiautomaton \mathfrak{M}_1 is at least as complex as \mathfrak{M}_2 , denoted by $\tilde{c}(\mathfrak{M}_1) \geq_c \tilde{c}(\mathfrak{M}_2)$, if (i) $\gamma(S_{\mathfrak{M}_1}) \geq \gamma(S_{\mathfrak{M}_2})$, or (ii) $\gamma(S_{\mathfrak{M}_1}) = \gamma(S_{\mathfrak{M}_2})$ and $\beta(S_{\mathfrak{M}_1}) \geq \beta(S_{\mathfrak{M}_2})$, or (iii) $\gamma(S_{\mathfrak{M}_1}) = \gamma(S_{\mathfrak{M}_2})$, $\beta(S_{\mathfrak{M}_1}) = \beta(S_{\mathfrak{M}_2})$ and $\sigma(S_{\mathfrak{M}_1}) \geq \sigma(S_{\mathfrak{M}_2})$.

5. EXAMPLES

In the following, strategy-implementing semiautomatons for several classic repeated-play games are examined. For each case, the action semigroups are constructed, their structures identified and complexities determined. Section 5.1 examines the one- and two-state semiautomatons implementing strategies in repeated-play two-by-two games, section 5.2 considers two three-state semiautomatons and section 5.3 considers larger automata. Each example provides a different point of focus. The one- and two-state semiautomatons expost the basics of constructing semigroup representations, identifying languages, component parts (e.g., subsemigroups and subgroups), and the properties (e.g., commutativity) of the component parts. This section also demonstrates that minimal automata with the same number of states can have different algebraic complexities. The three-state semiautomata in section 5.2 demonstrate that algebraic complexity is not monotonically increasing in the number of states. Specifically, the semigroup for semiautomaton 3-1 is aperiodic while the semigroup for automaton 3-2 contains *two* subgroups. In section 5.3 larger automata are used to highlight aspects of group representation and subsystem relationships. Notably, all of the section 5.1, 5.2 and 5.3 automata are complete. Appendix II addresses the Baron-Kalai semiautomaton implementing the Baron-Ferejohn equilibrium. Although incomplete, since its associated

semigroup contains no subgroups, this four-state semiautomaton is simpler than many of the semiautomata considered in earlier sections.

5.1 One- and two-state semiautomata implementing strategies in repeated play two-by-two games

Construction of the transformation semigroup for a strategy-implementing semiautomaton requires specification of the underlying set and action semigroup.³² Given the matrix representation introduced for grim-trigger type semiautomaton in section 2, this process is straightforward. The other possible two-state, two-transition semiautomata are depicted in figure 2. In this depiction, the states are labeled “1” and “2” while the transitions are labeled “A” and “B.” The states 1 and 2 can be represented by the 1×2 row vectors (1,0) and (0,1) respectively.³³ The state-transition information of the semiautomaton is captured by 2×2 transformation matrices, one row and column for each state. Because the semiautomata considered in this section are complete, each of these matrices must have exactly one “1” in each row and a “0” in every other position.³⁴ Thus, there are two possible places for the “1” in each of two rows and four possible complete 2×2 transformation matrices; $\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$, $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, $\begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$. All strategies in two-by-two games implementable by a complete two-state semiautomaton are

³² This discussion presents only the unique semiautomata. Binmore and Samuelson (1992) present a larger number of automata, however, the semiautomaton in each of these automata is isomorphic to one of these semiautomata through some re-labeling of the states. Thus, it is sufficient to address these machines alone. The discussion begins with the semiautomaton because it contains the information for constructing the transformation semigroup. The starting state does not affect the automaton complexity.

³³ The one-state automaton is a degenerate case of the two-state machine in which the states “1” and “2” are represented but the only transformation matrix is the identity matrix. The resulting algebra is the trivial, one-element algebra referred to in table VI as “#0.”

³⁴ This requirement results from the fact that, for these games, each player must make a move in every stage of the game, independent of the state of the semiautomaton. The result of this requirement is that the partial functions which are the foundation of algebraic automata are, in the case of these games, *functions*. Without this requirement, each row in the transformation matrix could have *at most* one “1” leaving open the possibility that some rows could be composed entirely of zeros. If incomplete automata are allowed, then there can be as many as nine distinct transformation matrices (see example in Appendix II).

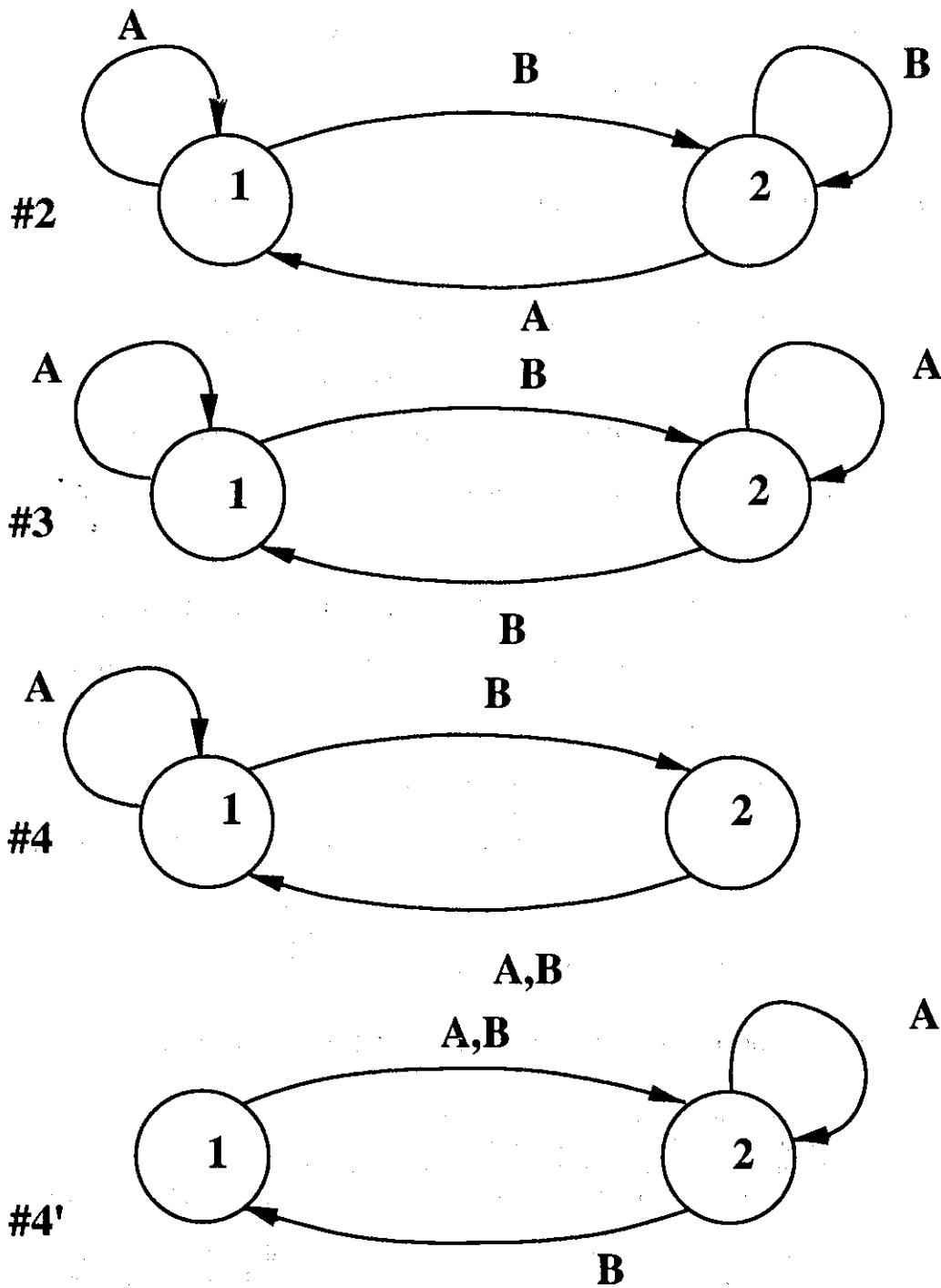


FIGURE 2: Two-state semiautomaton implementing strategies in repeated-play, two-by-two games.

represented by a pair of these four matrices (one transformation matrix for each of the two choices that the opposing player can take) and all of the actions that can occur with that strategy are represented by the products of the chosen pair.³⁵ The effect on state 1 of a transition that has the machine switch states is obtained by post-multiplying the vector $[1,0]$ by the transformation matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ to obtain $[0,1]$ indicating a transition from state 1 to state 2.

The vectors $[1,0]$ and $[0,1]$ are the underlying set of the transformation semigroup and the semigroup obtained by taking the multiplicative closure of the two transformation matrices is the action semigroup of the transformation semigroup representing the semiautomaton. The operation table for the action semigroup is defined by the matrix products and state transitions are effected by matrix multiplication on the state vectors.³⁶

This approach is applied to the remaining two-state semiautomata implementing strategies in repeated-play two-by-two games. The strategies are discussed using the terms of the repeated-play prisoner's dilemma game. The structure and complexity results are completely general for two-by-two games. Note that the first two strategies considered have similar pictorial representations but different algebraic complexities. Semiautomata #4 and #4' are more complex semiautomata whose algebras contain the algebras of the earlier examples as subalgebras and provide access to the concepts of semigroup decompositions and subsystem isomorphism.

³⁵ Since all of the products also must be one of these four transformation matrices, there can be no more than four unique words in a complete two-state automaton implementing a strategy in a two-by-two game.

³⁶ See Assmus and Florentin (1968) for an early exposition on applying these techniques to semiautomata.

Example 2: Tit-for-tat—a “simple” strategy. The semiautomaton for this strategy is depicted in figure 2#2. Letting the states 1 and 2 represent the “cooperate” and “defect” strategies respectively and the A and B transitions represent the γ and δ actions by the opponent respectively, the tit-for-tat strategy obtains. The response to γ is captured by the matrix $\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$ while the response to δ is captured by the matrix $\begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$ and the tit-for-tat strategy is the pair $\{\gamma, \delta\} = \left\{ \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \right\}$. Observe that both γ and δ are idempotent matrices and that because $\gamma\delta = \delta$ and $\delta\gamma = \gamma$ the pair is closed under multiplication. These products are presented in table III. The matrices γ and δ are the letters of the tit-for-tat semiautomaton alphabet and their products form the words of the semiautomaton. Because γ and δ are a closed set under multiplication, the alphabet is the same as the words. Call this semigroup S_2 , its properties are; (1) it is idempotent (each element repeats itself on the main diagonal), (2) it is not abelian (the operation table is not symmetric), (3) it has no identity, (4) it has no zero and (5) it satisfies a non-trivial identity ($aba = a, \forall a, b \in S$). These properties identify the semigroup as a rectangular band. Because the action semigroup contains no non-trivial subgroups, its 2-strategy complexity is $\tilde{c}_2 = (0,1,0)$. The transformation semigroup is $X_2 = (\{(1,0), (0,1)\}, S_2)$. In terms of the “power” of this system, it can recognize differences in the order of play (the algebra is non-abelian) but it can not count (the algebra is idempotent). Since the semigroup does not have an identity or inverse elements, there is no group structure.

	γ	δ
γ	γ	δ
δ	γ	δ

TABLE III: S_2 Action semigroup operation table for the tit-for-tat semiautomaton

Example 3: Tweedledum—a “complex” strategy. The semiautomaton for this strategy is depicted in figure 2#3. Let states 1 and 2 represent the cooperate defect strategies respectively and let the A and B transitions be the γ and δ actions respectively. At casual glance it is possible to think that the tweedledum and tit-for-tat semiautomata have the same complexity—their pictorial representations are similar. The difference, is the pattern for their transitions and, because of its pattern, the tweedledum semiautomaton defines a more powerful mathematical system. While the tit-for-tat semiautomaton algebra is an *aperiodic semigroup*, the tweedledum semiautomaton defines a *cyclic group*. Thus, the tweedledum semiautomaton algebra has a higher algebraic complexity than the tit-for-tat semiautomaton. As before, the states are represented by the vectors $(1,0)$ and $(0,1)$. The transformation matrices for tweedledum are $\gamma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and $\delta = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ and the strategy is the pair $\{\gamma, \delta\} = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \right\}$. Because γ is the identity matrix and δ is self-inverting these matrices define the two-element cyclic group S_3 with products as in table IV. The transformation semigroup is $X_3 = (\{(1,0), (0,1)\}, S_3)$ and $\tilde{c}_3 = (1,0,0)$. These strategies cannot depend on the sequence of play (the group is abelian) but it can count ($\delta \neq \delta^2$). Group power comes from γ being the identity matrix providing a stand pat ability and δ is self-inverting.

	γ	δ
γ	γ	δ
δ	δ	γ

TABLE IV: S_3 Action semigroup operation table for tweedledum strategy semiautomaton (n.b., this semigroup is a group).

Example 4: Tweedledee—a “decomposable” strategy. The semiautomaton for this strategy is given in figure 2#4. The tweedledee strategy is obtained by letting states 1 and

2 represent cooperate and defect and the A and B transitions represent the γ and δ actions. This semiautomaton *looks* “simpler” than the semiautomatons for tit-for-tat and tweedledum (ignoring the transition patterns) but, in fact, its algebra is the product of the algebras from earlier examples. The states again are the vectors (1,0) and (0,1). The transitions for γ are represented by the matrix $\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$ and the responses to δ are captured by $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ so that $\{\gamma, \delta\} = \left\{ \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \right\}$. Observe that $\delta^2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and $\gamma\delta = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$ neither of which is a letter in the alphabet for this semiautomaton, so the products δ^2 and $\gamma\delta$ are additional words in the tweedledee semiautomaton language. The operation table for S_4 and its cyclic subgroup are presented in tables V (a) and (b). From examples 1, 2 and 3, $\{\delta^2, \gamma\delta\} = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \right\}$ is a semilattice, $\{\gamma, \gamma\delta\} = \left\{ \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \right\}$ is a band and $\{\delta, \delta^2\} = \left\{ \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right\}$ is a two-element cyclic group. Because this system contains the earlier systems as subsystems each of those systems is i -simpler than S_{m_4} .

This semigroup admits two distinct decompositions; (1) a two-element semilattice and a two-element cyclic group with complexity (1,0,1) and (2) a two-element band and a two-element cyclic group which has complexity (1,1,0). Either pair of structures will generate the semigroup for the tweedledee semiautomaton.³⁷ Because the action semigroup for the semiautomaton implementing the tweedledee strategy can be obtained from two simpler algebras, it is called *decomposable*. Since the first decomposition (into a semilattice and a cyclic group) is simpler than the second, $\bar{c}_4 = (1,0,1)$ for tweedledee’s semigroup. The transformation semigroup is $X_4 = (\{(1,0), (0,1)\}, S_4)$.

³⁷ Because the elements of the action semigroup are expressed as transformation matrices, the “product” here is matrix multiplication.

	γ	δ	δ^2	$\gamma\delta$
γ	γ	$\gamma\delta$	γ	$\gamma\delta$
δ	γ	δ^2	δ	$\gamma\delta$
δ^2	γ	δ	δ^2	$\gamma\delta$
$\gamma\delta$	γ	γ	$\gamma\delta$	$\gamma\delta$

(a)

	δ	δ^2
δ	δ^2	δ
δ^2	δ	δ^2

(b)

TABLE V (a) and (b): (a) S_4 Action semigroup operation table for tweedledee strategy semiautomaton and (b) subsemigroup operation table.

The complexities of the one- and two-state automata implementing strategies in two-by-two games are presented in table VI in order of increasing complexity. It can be seen that the non-minimal two-state semiautomaton #0 has the same algebraic complexity as a one-state semiautomaton. The minimal two-state semiautomata implementing repeated-play strategies are uniquely characterized by four algebras; two that are different aperiodic semigroups and two that are, or contain, a group. Specifically, the semigroups for the grim-trigger and tweety pie semiautomata are isomorphic to S_1 , the tit-for-tat semigroup is isomorphic to S_2 , the tweedledum and tat-for-tit semigroups are isomorphic to S_3 and the semigroup for tweedledee is isomorphic to the decomposable semigroup S_4 .³⁸ Once the semigroup structures are identified, complexity comparisons are direct.

From the table of semiautomaton algebras it, also, is possible to reinforce the role of some properties affecting behavior. Specifically, commutativity requires independence of order; only the actions by the other player are relevant, rather than the sequence of actions by the other player. Idempotence (present in the first three algebras) requires that the response to the other player's actions is the same whether that action is

³⁸ Using the Binmore and Samuelson (1992) labeling: "aa" and "dd" give #0; "ba" and "ca" give #1; "bc" and "cb" give #2; "ad," "bb," and "cc" give #3; and "ab," "ac," "bd," and "cd" give either #4 or #4'.

taken two, three, four or any number of times; that is the semiautomaton can't "count."

Perhaps the biggest difference occurs when both identity and inverse elements are present in the algebras.

Semiautomata	Common Strategy Names	Algebraic Structure	Complexity
#0	cooperate, defect	point (trivial semigroup)	(0,0,0)
#1	grim trigger, tweetypie	2-element semilattice	(0,0,1)
#2	tit-for-tat	2-element rectangular band	(0,1,0)
#3	tweedledum, tat-for-tit	2-element cyclic group	(1,0,0)
#4 and #4'	tweedledee	4-element semigroup product of 2-element semilattice and 2-element cyclic group	(1,0,1)

TABLE VI: Algebraic structure and complexity of complete semiautomata implementing strategies in repeated-play, two-by-two games.

Structurally, the presence of both identity and inverse elements is the difference between the aperiodic semigroups S_1 and S_2 , and the group containing semigroups S_3

and S_4 . Because of the identity elements, S_3 and S_4 have a stand pat ability. Because S_3 and S_4 have inverse elements, they are able to “recover” from “mistakes” or to return to their initial patterns after a deviation such as might be required to “punish” an opponent for only a finite number of rounds. Alternatively, the ability to return to an earlier state allows the semiautomaton to engage in experimentation and test the effect of certain actions without permanent commitment. This is not possible for the aperiodic semigroups S_1 and S_2 . For example, S_1 which can implement the grim-trigger strategy by starting in the cooperate state, has only one alternate action—switch to the defect state. Once in the defect state, it is not possible to return to the cooperate state. The S_2 type automata implementing the tit-for-tat strategy can return to previous states but they can’t stand pat in every state. The S_3 and S_4 type machines can effect both of these actions.

Proposition 1 and corollary 1.1 summarize the properties of the one- and two-state machine semigroups. Corollary 1.2 applies the complexity rankings to the repeated-play prisoner’s dilemma game.

PROPOSITION 1: Let $\mathfrak{M} = (Q, \Sigma, F)$ where $|Q| \leq 2$, $|\Sigma| \leq 2$, be a complete minimal semiautomaton implementing a strategy in a two-by-two game and let $S_{\mathfrak{M}}$ be the action semigroup for \mathfrak{M} , then $S_{\mathfrak{M}}$ is isomorphic to one of the following algebras;

- (i) single-element semigroup,
- (ii) two-element semilattice,
- (iii) two-element rectangular band,
- (iv) two-element cyclic group or
- (v) product of a two-element semilattice and a two-element cyclic group.

COROLLARY 1.1: Let $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$ and \mathcal{M}_4 be complete, minimal semiautomata as described in proposition 1 with associated semigroups be $S_{\mathcal{M}_0}, S_{\mathcal{M}_1}, S_{\mathcal{M}_2}, S_{\mathcal{M}_3}$ and $S_{\mathcal{M}_4}$.

Further, let the algebras be (0) single element semigroup, (1) two-element semilattice, (2) two-element rectangular band, (3) two-element cyclic group and (4) product of a two-element semilattice and a two-element cyclic group respectively. Then the complexity ranking is $\mathcal{M}_4 >_i \mathcal{M}_i, i \in \{0,1,2,3\}$ and $\tilde{c}(S_{\mathcal{M}_4}) >_c \tilde{c}(S_{\mathcal{M}_3}) >_c \tilde{c}(S_{\mathcal{M}_2}) >_c \tilde{c}(S_{\mathcal{M}_1}) >_c \tilde{c}(S_0)$.

COROLLARY 1.2: The grim-trigger semiautomaton is the simplest minimal one- or two-state semiautomaton implementing a Nash equilibrium strategy in the repeated-play prisoner's dilemma game and supporting the cooperative outcome.

REMARK 1: The grim-trigger semiautomaton is the simplest semiautomaton within the class of non-degenerate semilattices. The result follows because any simpler semilattice would have to have fewer join-irreducibles which would make the semilattice degenerate.

5.2 A "simple" and a "complex" three-state semiautomaton

Two examples demonstrate that algebraic complexity is not monotonic in the number of states.³⁹ The semiautomatons 3-1 and 3-2 depicted in figures 4 and 5. Each has three states; C-1, C-2 and D which are represented algebraically by the 1×3 vectors $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$ respectively. Although there are three states, there are only two actions γ and δ , an opposing player can make so, the transformation matrices are 3×3 and a strategy is captured by a pair of these matrices.

³⁹ Semiautomaton 3-1 is designed to be simple while semiautomaton 3-2 is a "small" version of machines considered by Osborne and Rubinstein (1994, Chapter 9).

Example 5: A “simple” three-state strategy. The semiautomaton 3-1 is depicted in figure

5. The transformation matrices for the strategy are

$$\{\gamma, \delta\} = \left\{ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \right\}$$

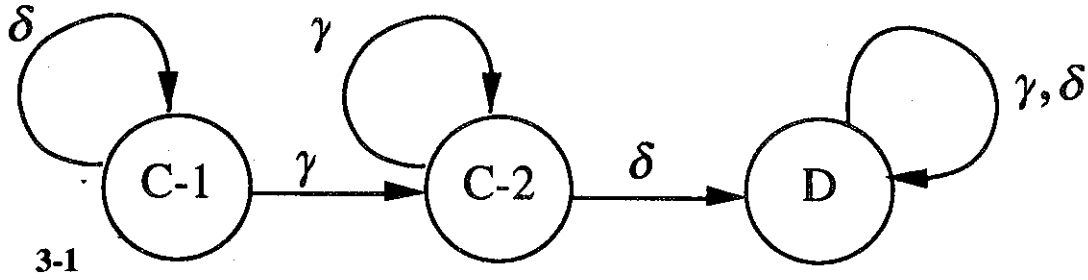


FIGURE 5: Simple three-state machine

and the products for S_{3-1} are presented in table VII. The operation table reveals that the semigroup is neither abelian ($\gamma\delta \neq \delta\gamma$) nor idempotent ($((\delta\gamma)(\delta\gamma) \neq \delta\gamma)$) and contains no non-trivial subgroups. Thus, semigroup 3-1 is relatively simple, more complex than the algebra for the “tit-for-tat” strategy (there is more than one subsemigroup) but less complex than the “tweedledum” strategy algebra (there are no non-trivial subgroups).

Thus, even though machine 3-1 has more states and as large a language as the section 5.1 machines, its algebra is simpler than some of those machines. Structurally, S_{3-1} does have a zero, $\gamma\delta$, which results in the “D” state independent of the automaton’s initial state.

	γ	δ	$\gamma\delta$	$\delta\gamma$
γ	γ	$\gamma\delta$	$\gamma\delta$	$\gamma\delta$
δ	$\delta\gamma$	δ	$\gamma\delta$	$\delta\gamma$
$\gamma\delta$	$\gamma\delta$	$\gamma\delta$	$\gamma\delta$	$\gamma\delta$
$\delta\gamma$	$\delta\gamma$	$\gamma\delta$	$\gamma\delta$	$\gamma\delta$

TABLE VII: S_{3-1} Action semigroup operation table for machine 3-1

Example 6: A “complex” three state machine. Semiautomaton 3-2 is depicted in figure 6 and its transformation matrices are

$$\{\gamma, \delta\} = \left\{ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \right\}.$$

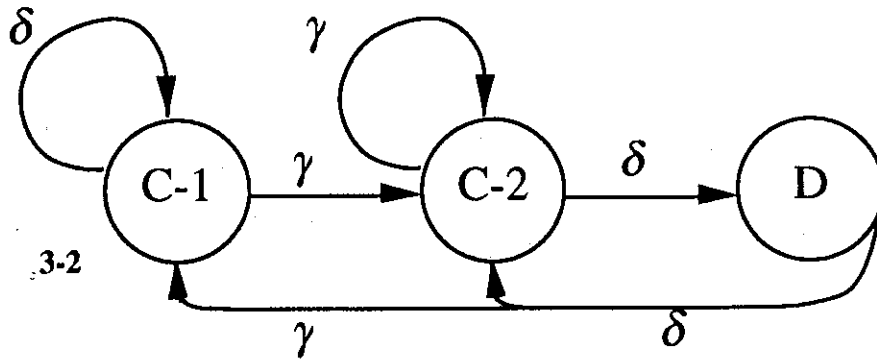


FIGURE 6: Complex three state machine

The algebra S_{3-2} is presented in table VIII. With regard to complexity, the most important feature of this semigroup is that fact that it contains *two* subgroups. Thus, S_{3-2} is more complex than the algebra for any of the semiautomata considered in this section. By design, machine 3-2 has more states than the section 5.1 machines and its algebra has more words than those semiautomata. The subgroups are each a two-element cyclic group; $\{\delta, \delta\delta\}$ and $\{\gamma\delta, \gamma\delta\gamma\delta\}$. To more easily see these subgroups, their operation tables are presented separately as tables IX (a) and (b). The sets on which each of the subgroups acts are different; the subgroup $\{\delta, \delta\delta\}$ interchanges elements of the set $\{C1, C2, D\}$ while the subgroup $\{\gamma\delta, \gamma\delta\gamma\delta\}$ interchanges elements of the set $\{C1, D\}$.⁴⁰ Because S_{3-2}

⁴⁰ This aspect of the semiautomaton algebra is most evident in the mapping table M-3-2 in appendix II. It can also be seen in the fact that the underlying sets for the transformation groups in the Krohn-Rhodes decompositions differ. Formally, the transformation groups in the Krohn-Rhodes decomposition for this algebra are $TG_1 = (\{(1,0,0), (0,1,0), (0,0,1)\}, (\delta, \delta^2))$ and $TG_2 = (\{(1,0,0), (0,0,1)\}, (\gamma\delta, \gamma\delta\gamma\delta))$.

has two subgroups, it is more complex than machine 3-1 and more complex than any of the two-state semiautomata addressed in section 5.1.⁴¹

	γ	δ	γ^2	$\gamma\delta$	δ^2	$\delta\gamma$	$\gamma^2\delta$	$\gamma\delta\gamma$	$\delta\gamma\delta$	$\delta\gamma\delta\gamma$	$\gamma^2\delta\gamma$	$\gamma\delta\gamma\delta$	$\delta\gamma\delta\gamma\delta$
γ	γ^2	$\gamma\delta$	γ^2	$\gamma^2\delta$	γ	$\gamma\delta\gamma$	$\gamma^2\delta$	$\gamma^2\delta\gamma$	$\gamma\delta\gamma\delta$	γ	$\gamma^2\delta\gamma$	$\gamma^2\delta\gamma$	$\gamma\delta$
δ	$\delta\gamma$	δ^2	γ^2	$\delta\gamma\delta$	δ	γ	$\gamma^2\delta$	$\delta\gamma\delta\gamma$	$\gamma\delta$	$\gamma\delta\gamma$	$\gamma^2\delta\gamma$	$\delta\gamma\delta\gamma\delta$	$\gamma\delta\gamma\delta$
γ^2	γ^2	$\gamma^2\delta$	γ^2	$\gamma^2\delta$	γ^2	$\gamma^2\delta\gamma$	$\gamma^2\delta$	$\gamma^2\delta\gamma$	$\gamma^2\delta\gamma$	γ^2	$\gamma^2\delta\gamma$	$\gamma^2\delta\gamma$	$\gamma\delta$
$\gamma\delta$	$\gamma\delta\gamma$	γ	γ^2	$\gamma\delta\gamma\delta$	$\gamma\delta$	γ^2	$\gamma^2\delta$	γ	$\gamma^2\delta$	$\gamma^2\delta\gamma$	$\gamma^2\delta\gamma$	$\gamma\delta$	$\gamma^2\delta\gamma$
δ^2	γ	δ	γ^2	$\gamma\delta$	δ^2	$\delta\gamma$	$\gamma^2\delta$	$\gamma\delta\gamma$	$\delta\gamma\delta$	$\delta\gamma\delta\gamma$	$\gamma^2\delta\gamma$	$\gamma\delta\gamma\delta$	$\delta\gamma\delta\gamma\delta$
$\delta\gamma$	γ^2	$\delta\gamma\delta$	γ^2	$\gamma^2\delta$	$\delta\gamma$	$\delta\gamma\delta\gamma$	$\gamma^2\delta$	$\gamma^2\delta\gamma$	$\delta\gamma\delta\gamma\delta$	$\delta\gamma$	$\gamma^2\delta\gamma$	$\gamma^2\delta\gamma$	$\delta\gamma\delta$
$\gamma^2\delta$	$\gamma^2\delta\gamma$	γ^2	γ^2	$\gamma^2\delta\gamma$	$\gamma^2\delta$	γ^2	$\gamma^2\delta$	γ^2	$\gamma^2\delta$	$\gamma^2\delta\gamma$	$\gamma^2\delta\gamma$	$\gamma^2\delta$	$\gamma^2\delta\gamma$
$\gamma\delta\gamma$	γ^2	$\gamma\delta\gamma\delta$	γ^2	$\gamma^2\delta$	$\gamma\delta\gamma$	γ	$\gamma^2\delta$	$\gamma^2\delta\gamma$	$\gamma\delta$	$\gamma\delta\gamma$	$\gamma^2\delta\gamma$	$\gamma^2\delta\gamma$	$\gamma\delta\gamma\delta$
$\delta\gamma\delta$	$\delta\gamma\delta\gamma$	$\delta\gamma$	γ^2	$\delta\gamma\delta\gamma\delta$	$\delta\gamma\delta$	γ^2	$\gamma^2\delta$	$\delta\gamma$	$\gamma^2\delta$	$\gamma^2\delta\gamma$	$\gamma^2\delta\gamma$	$\delta\gamma\delta$	$\gamma^2\delta\gamma$
$\delta\gamma\delta\gamma$	γ^2	$\delta\gamma\delta\gamma\delta$	γ^2	$\gamma^2\delta$	$\delta\gamma\delta\gamma$	$\delta\gamma$	$\gamma^2\delta$	$\gamma^2\delta\gamma$	$\delta\gamma\delta$	$\delta\gamma\delta\gamma$	$\gamma^2\delta\gamma$	$\gamma^2\delta\gamma$	$\delta\gamma\delta\gamma\delta$
$\gamma^2\delta\gamma$	γ^2	$\gamma^2\delta\gamma$	γ^2	$\gamma^2\delta$	$\gamma^2\delta\gamma$	γ^2	$\gamma^2\delta$	$\gamma^2\delta\gamma$	$\gamma^2\delta$	$\gamma^2\delta\gamma$	$\gamma^2\delta\gamma$	$\gamma^2\delta\gamma$	$\gamma^2\delta\gamma$
$\gamma\delta\gamma\delta$	γ	$\gamma\delta\gamma$	γ^2	$\gamma\delta$	$\gamma\delta\gamma\delta$	γ^2	$\gamma^2\delta$	$\gamma\delta\gamma$	$\gamma^2\delta$	$\gamma^2\delta\gamma$	$\gamma^2\delta\gamma$	$\gamma\delta\gamma\delta$	$\gamma^2\delta\gamma$
$\delta\gamma\delta\gamma\delta$	$\delta\gamma$	$\delta\gamma\delta\gamma$	γ^2	$\delta\gamma\delta$	$\delta\gamma\delta\gamma\delta$	γ^2	$\gamma^2\delta$	$\delta\gamma\delta\gamma$	$\gamma^2\delta$	$\gamma^2\delta\gamma$	$\gamma^2\delta\gamma$	$\delta\gamma\delta\gamma\delta$	$\gamma^2\delta\gamma$

TABLE VIII: S_{3-2} Action semigroup operation table for semiautomaton 3-2.

	δ	δ^2
δ	δ^2	δ
δ^2	δ	δ^2

(a)

	$\gamma\delta$	$\gamma\delta\gamma\delta$
$\gamma\delta$	$\gamma\delta\gamma\delta$	$\gamma\delta$
$\gamma\delta\gamma\delta$	$\gamma\delta$	$\gamma\delta\gamma\delta$

(b)

TABLE IX (a) and(b): Operation tables for subgroups in the semiautomaton 3-2.

5.3 "Large" semiautomata and group complexities

In the previous examples, following Krohn and Rhodes, group complexities are scaled by the subgroups. Because simply counting the number of subgroups doesn't

⁴¹ An additional feature of this semigroup is the ideal $\{\gamma\gamma, \gamma\delta, \gamma\delta\gamma\}$ in which $\gamma\gamma$ maps every state into C2, $\gamma\delta$ maps every state into D and $\gamma\delta\gamma$ maps every state into C1. See Johnson (1995) for definition and further discussion of ideals in economics.

reveal the properties of the group, this approach may not be the most appropriate for game theory. In addition, there is at least one other aspect of the number-of-subgroups measure that is bothersome. Specifically, a classic theorem in Group Theory is that a group of order n has subgroups of every order that divides n . Thus, if a group complexity is based on the number of subgroups in the group, then the complexity of non-prime order groups can grow arbitrarily large while the complexity of prime order groups will be 1 independent of how large is the group.

There are a number of alternative ways of measuring the group complexity. In the earlier examples, the principle criteria for ranking semigroup complexities were dependence on order and counting ability. The measure $\mu(S)$ was introduced because it allows identification of a very simple class of abelian groups—specifically, cyclic groups. Every cyclic group is (1) abelian and (2) generated by a single matrix. Because they are abelian and because they are “easily” generated, these groups are a natural class to identify as low complexity. Example 7 demonstrates the relationship between $\gamma(S)$ and $\mu(S)$ while example 8 highlights the subsystem relationships.

Example 7: Modified three period trigger. As depicted in figure 7, the strategy is, if you see “A” then stay where you are and, if you see “B” then advance to the next state in the loop. Both of these rules are independent of the state. The algebra for this machine is presented in table X. Significant features of this algebra are that it is a cyclic group of order 4 and that $B^4 = I = A$. Equally important, the algebra is the product of two groups of order 2. For similar rules implemented by a five-state semiautomaton (five is prime), there would be no decomposition while the six-state machine algebra would be the product of a group of order 2 and a group of order 3 and a seven-state machine’s algebra

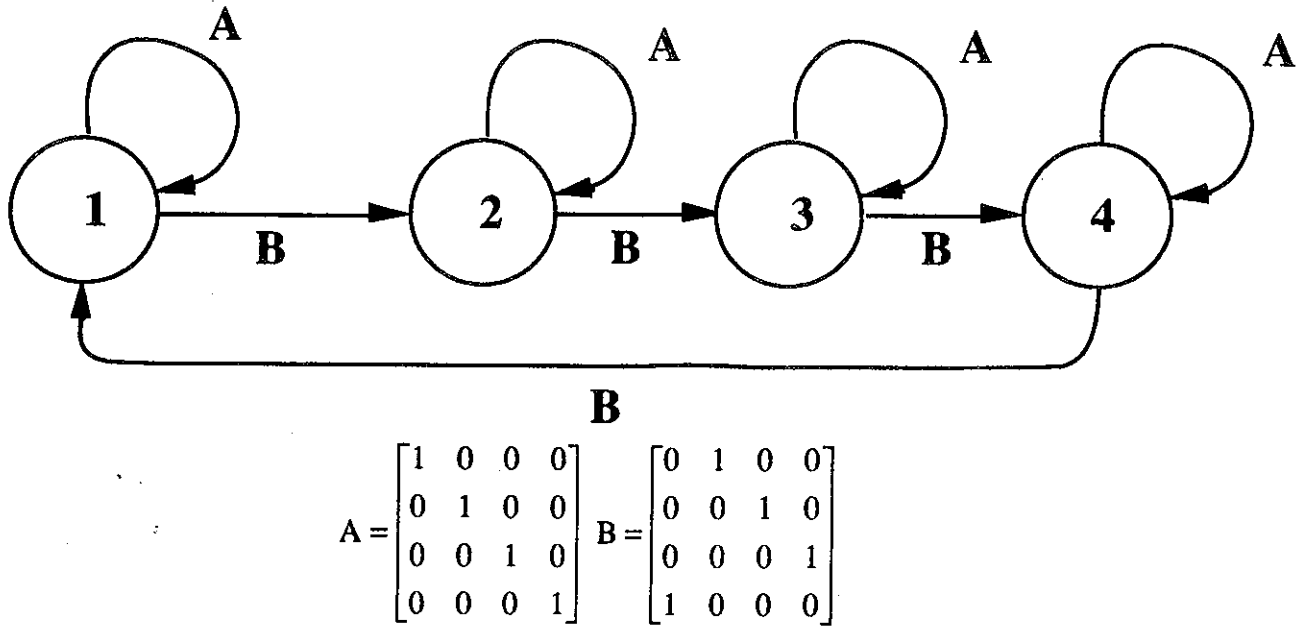


FIGURE 7: Example 7, four-state modified 3-period trigger automaton and transformation matrices

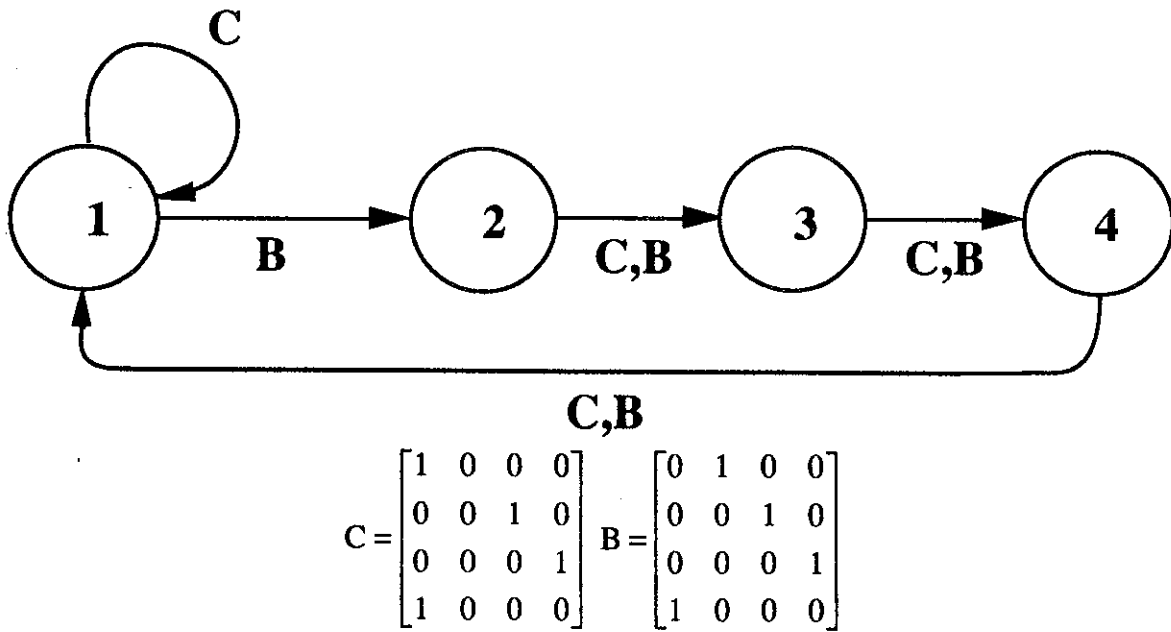


FIGURE 8: Example 8, four-state 3-period trigger automaton and transformation matrices

again would not have a decomposition (seven is prime). Thus, the measure $\gamma(S)$ behaves erratically while the power of the machines changes little.⁴² In contrast, because each of the semiautomata are generated by a single matrix, the measure $\mu(S)$ is constant at 1.

	B	B ²	B ³	I
B	B ²	B ³	I	B
B ²	B ³	I	B	B ²
B ³	I	B	B ²	B ³
I	B	B ²	B ³	I

TABLE X: Operation table for cyclic group generated by the “B” transformation matrix.

Example 8: Three-period trigger. The semiautomaton depicted in figure 8 implements a strategy that is, if you see “C” and you are in state 1, then stay where you are otherwise, advance to the next state in the loop and, if you see “B” advance to the next state in the loop. This strategy is not independent of the state. Although the pictorial representation of example 7 looks “complicated” relative to example 8, the algebra for the example 7 automaton is simple compared to the algebra for example 8. The intuition for why example 8 is more complicated than example 7 is that the strategy implemented in example 8 requires that the automaton count to keep track of which state it is in while the strategy in example 7 depends only on the observation and not the state. Formally, it is easy to demonstrate that the example 7 automaton algebra is a subsystem of that for the example 8 automaton, thus $\mathfrak{M}_8 \geq \mathfrak{M}_7$.⁴³

⁴² Perhaps the most significant change results from the fact that the order of the group determines how high the machines can count. This feature would be reflected if $\mu(\)$ incorporated the dimension of the generating matrices.

⁴³ To see this observe that because $B^4 = I = A$, the system for example 7 is the cyclic group of order 4 while the system for example 8 includes the cyclic group of order 4 generated by the B matrix as well as a three element aperiodic semigroup generated by the C matrix and interaction terms from these two systems.

6. ALGEBRAIC COMPLEXITY AND “STATE-BASED” MEASURES OF COMPLEXITY

The examples in the previous sections demonstrate that there can be large differences in the power of various automata defined on the same number of states. While these examples are useful in understanding the concepts behind algebraic complexity, they do not make the case for arbitrary sized automata. It is natural to ask if these same structures can be replicated on arbitrary sized automata. The answer is yes—with the implication that, if the “power” of an automaton depends on specific abilities such as; detecting order, counting, ability to return to previous positions, etc., then the number of states in the automaton reveals little about the power of the automaton. While the semigroup classes identified in the examples are not exhaustive and they do not provide invariants for all automata, they do distinguish between relatively more complex and simple machines and on a finer scale that provided by Krohn-Rhodes complexity. For specific applications—as in the case of invariants for the one- and two-state automata from two-by-two games—it may be possible to design more precise complexities.

Proposition 2 assures that each of the structures identified in section 5.1 can be recreated for some minimal automaton of any arbitrary size greater than or equal to 2. Specifically, provided n is at least two, there are n -state automata whose algebras are groups, band or semilattices. So, automata defined on n states can have very different algebraic complexities. Further, since there are “simple” and “complex” machines of all sizes, it is clear that there are “large” automata whose algebras are semilattices and therefore are “simpler” than automata with fewer states but which have the structure of bands or groups. Thus, simply comparing the number of states (even in minimal

automata) can be misleading in trying to determine the relative algebraic complexities of different automata. Proposition 3 assures that, in addition to the structures identified in section 5.1, for three or more states, there is an abelian, aperiodic “counting” automaton whose complexity lies between bands and groups. Tilson’s theorem assures that for any integer n , there is an automaton whose algebra has a group complexity of n .

PROPOSITION 2: Let $m \geq 2$ be an integer, then there exists a minimal automaton A with m states and action semigroup S_A such that S_A is,

- (i) a cyclic group of order m with $\gamma(S_A) \geq 1$ and $\mu(S_A) = 1$,
- (ii) a band, or
- (iii) a semilattice with $\sigma(S_A) = m - 1$.

Proof: See Appendix I.

REMARK 2: A consequence of proposition 2 is that for integers $m \geq n \geq 2$, there are minimal automata A_m and A_n with m and n states respectively and action semigroups S_m and S_n respectively such that $\gamma(S_n) > \gamma(S_m)$, $\mu(S_n) > \mu(S_m)$ and $\beta(S_n) > \beta(S_m)$.

REMARK 3: If the number of states is prime, then the only group is the cyclic group. If $m \geq 4$ is an even integer, then there exist a m -state automata A_1 and A_2 such that S_{A_1} is a cyclic group and S_{A_2} is a non-abelian dihedral group (Dean (1990)). Since A_2 ’s algebra is not abelian, it is not cyclic and it has more than one generator, thus $\mu(S_{A_2}) > \mu(S_{A_1})$.

PROPOSITION 3: Let $m \geq 3$ be an integer, then there exists a minimal automaton A such that S_A is an abelian, non-idempotent, aperiodic semigroup.

Proof: See Appendix I.

REMARK 4: The automaton identified in the proof of proposition 3 is an abelian “counting” automaton that counts up to one less than the number of states. After that, it attains the zero for its algebra and can’t advance further. The cyclic group of order m , in proposition 2, can “count” up to the number of states (i.e., it counts mod m).

THEOREM (Tilson):⁴⁴ Let $n \geq 0$ be given, then there exists an automaton A_n with action semigroup S_n such that $\gamma(S_n) = n$.

Finally, Banks and Sundaram do not define the complexity of a machine but, instead, offer a means for comparing the complexities of two or more machines.⁴⁵ They call their method of comparison the *increasing complexity criterion* (ICC). The examples provided in section 5.1 allow algebraic complexity to be distinguished from Banks and Sundaram’s (1990) increasing complexity criterion. For two minimal automata $A_1 = (\mathcal{M}_1, q_1^1, T_1)$ with $\mathcal{M}_1 = \langle Q_1, \Sigma_1, \mu_1 \rangle$ and $A_2 = (\mathcal{M}_2, q_2^1, T_2)$ with $\mathcal{M}_2 = \langle Q_2, \Sigma_2, \mu_2 \rangle$, the relation \succeq_{ICC} is defined as follows,

- (i) $|Q_1| \geq |Q_2|$, and
- (ii) there is a subset $\hat{Q}_1 \subseteq Q_1$, and a bijection $\phi: Q_2 \rightarrow \hat{Q}_1$ such that $R(q_2^i) \leq R(\phi(q_2^i))$ for $q_2^i \in Q_2$. ($R(q)$ is the number of transitions, or edges, emanating from state q .)

⁴⁴ Tilson actually provides several theorems similar to this result. The two most relevant results are Theorem 4.22 and Proposition 4.23 in Tilson (1971). Theorem 4.22 is proved for both the standard number of groups complexity measure and Tilson’s own 2-tuple complexity. Proposition 4.23 offers a specific construction on n states with a group complexity of $n-1$.

⁴⁵ Banks and Sundaram actually consider several related means for comparing complexities. The measure addressed here is their *increasing complexity criterion* (ICC). Their alternative comparison methods build on this basic criterion. One of their main points is that how complexity is measured can lead to different complexity rankings.

If (i) and (ii) hold, then $A_1 \succeq_{ICC} A_2$. Parallel to the construction above, the notation $A_1 \succ_{ICC} A_2$ is used to denote the case where $A_1 \succeq_{ICC} A_2$ and $\neg(A_2 \succeq_{ICC} A_1)$ while $A_1 \approx_{ICC} A_2$ is used for the case where $A_1 \succeq_{ICC} A_2$ and $A_2 \succeq_{ICC} A_1$. Banks and Sundaram note that this relation need not be complete.

REMARK 5: Given the examples in section 5, it should be clear that there exist automata A_1 and A_2 such that $A_1 \succ_{ICC} A_2$ but $\gamma(A_2) > \gamma(A_1)$ and $\mu(A_2) > \mu(A_1)$.

7. CONCLUSIONS

Several authors have demonstrated that including a “complexity” cost to the structure of repeated-play games can reduce the number of equilibria. In most of these applications, the complexity measure has been either the number of states in the machine or the number of states in the minimal machine implementing the strategy. Banks and Sundaram have suggested that automaton complexity might depend on features other than the number of states (i.e., the number of edges in the graph representation of the automaton) and demonstrate that several different “complexity” rankings can be obtained depending on how these other features impact the “complexity scale.” By making this observation, they raise the question of what is the “right” complexity measure for automata.

The appropriate measure of automaton complexity depends on what features are viewed as having economic impacts. In the preceding, properties of the algebra defined by the automaton provide a basis for comparing abilities of the system defined by the automata. Each of the abilities identified is associated with a specific property of the algebra defined by the automaton, of responding to differences in order (abelian),

counting (idempotence), standing pat (identity element) and returning to previously visited states (inverse elements). Despite the fact that there is no elementary means for identifying the algebraic complexity of an automaton, algebraic complexity has several appealing features. By design, it reflects the “power” of the mathematical system defined by the automaton (precisely the property Kalai and Stanford seek). When pushed to the standard of invariants, both structure and complexity are captured concisely. Indeed, when invariants are obtained, as in section 5.1, these items alone are sufficient to determine other complexity scalings.

The first step in determining algebraic complexity is to identify the minimal automaton. For strategy-implementing automata, this step was accomplished by Kalai and Stanford. Algebraic complexity is determined by identifying the basic elements of the algebra determined by the automaton. Although the results presented here demonstrate that algebraic complexity can distinguish among automata with the same number of states, algebraic complexity is not properly “finer” measure than the number of states in the automaton. This is because algebraic complexity is not monotonic in the number of states of a machine (even if the machine is minimal). Specifically, automata with a “small” number of states can be have higher algebraic complexity than automata with a “large” number of states.

When applied to automata implementing strategies in the repeated-play prisoner’s dilemma game, the fact that algebraic complexity can differentiate between several strategies implementable by two-state automata allows a unique simplest Nash equilibrium supporting the cooperative outcome in the repeated play prisoner’s dilemma

game—the grim trigger—to be identified. The only simpler Nash equilibrium automaton is the “defect” strategy machine and that machine’s algebra is trivial.

Because of the different structures identified, the question of what features impart “cost” arises naturally. Specifically, for strategy-implementing automata used in repeated-play games is it necessary to identify invariants or is something less precise, like Krohn-Rhodes number-of-prime-groups complexity, sufficient? In considering these questions, it is worth remembering, that only very fine scaling of algebraic complexity is sufficient to identify the grim trigger strategy as the simplest Nash Equilibrium implementing the cooperative outcome for the repeated-play prisoner’s dilemma game.

APPENDIX I

Proof of Proposition 2: Let $A_1 = (\mathfrak{M}_1, i_1, T_1)$, $\mathfrak{M}_1 = (Q_1, \Sigma_1, F_1)$, $Q_1 = \{q_0, \dots, q_{m-1}\}$, $\Sigma_1 = \{\sigma\}$ and F_1 be given and defined so that $q_k \sigma = q_{k+1(\text{mod } m)}$. Note that $S_{\mathfrak{M}_1}$ is a cyclic group so that $\mu(S_{\mathfrak{M}_1}) = 1$. Let $A_2 = (\mathfrak{M}_2, i_2, T_2)$, $\mathfrak{M}_2 = (Q_2, \Sigma_2, F_2)$, $Q_2 = \{q_0, \dots, q_{m-1}\}$, $\Sigma_2 = \{\sigma_0, \dots, \sigma_{m-1}\}$ F_2 be given and defined so that q_i is an $1 \times m$ vector with a "1" in the $i+1$ position and "0"s elsewhere, and the transformation for σ_i is represented by an $m \times m$ matrix with "1"s in the $i+1$ column and zeros elsewhere. Note that $S_{\mathfrak{M}_2}$ is a band. Let $A_3 = (\mathfrak{M}_3, i_3, T_3)$, $\mathfrak{M}_3 = (Q_3, \Sigma_3, F_3)$, $Q_3 = \{q_0, \dots, q_{m-1}\}$, $\Sigma_3 = \{\sigma_0, \dots, \sigma_{m-1}\}$ and F_3 be given and defined so that $q_i \sigma_j = q_{\min\{i,j\}}$. Observe that $S_{\mathfrak{M}_3}$ is an idempotent, abelian semigroup and, therefore, has a representation as a semilattice (in fact, a chain) and $\sigma(S_{\mathfrak{M}_3}) = m-1$ but $\gamma(S_{\mathfrak{M}_3}) = 0$. Therefore, these machines are minimal and have the same number of states but different algebraic complexities. ♦

Proof of Proposition 3: Let $A = (\mathfrak{M}, i, T)$, $\mathfrak{M} = (Q, \Sigma, F)$, $Q = \{q_0, \dots, q_{m-1}\}$, $\Sigma = \{\sigma_0, \dots, \sigma_{m-1}\}$ F be given and defined so that q_i is an $1 \times m$ vector with a "1" in the $i+1$ position and "0"s elsewhere, and the transformation σ_i is represented by an $m \times m$ matrix M with a "1" in the (1,1) a "1" in the (2,3) position and, for row i less than m a "1" in the $i+1$ column and for row m a "1" in the $(m,1)$ position and zeros elsewhere. Note that $S_{\mathfrak{M}}$ is abelian but not idempotent. Further observe that M^{m-1} is a matrix with "1"s in the first column and zeros elsewhere. This matrix is a zero for the algebra. ♦

APPENDIX II

The Baron-Kalai semiautomaton

A recent instance where automaton complexity is minimized occurs in Baron and Kalai (1992). In their analysis, Baron and Kalai argue that because they can implement the Baron-Ferejohn (1989) strategy with a four-state machine and because there are no three-state machines capable of implementing a strategy in this voting game, that the Baron-Ferejohn strategy must be the simplest. As seen above, the number of states in an semiautomaton is not a reliable indicator of algebraic complexity. Figure II-1 presents the graphical representation of the Baron-Kalai machine.

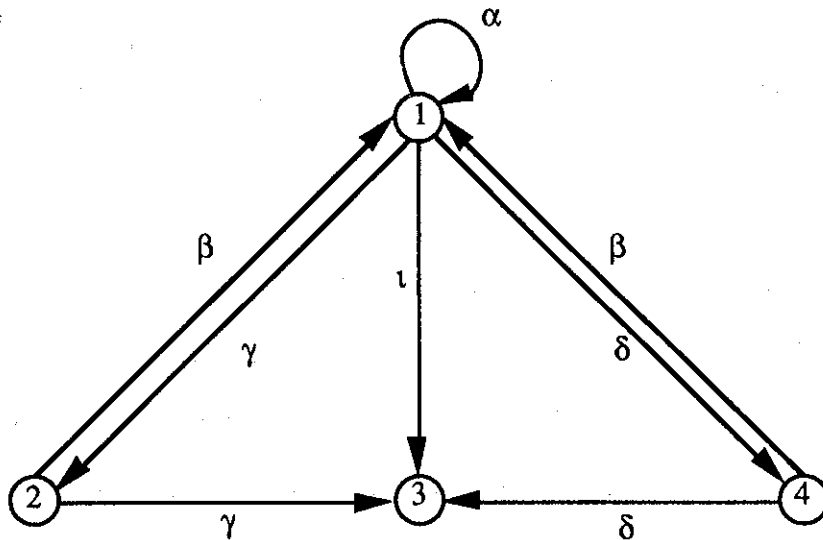


FIGURE II-1: Baron-Kalai semiautomaton implementing the Baron Ferejohn equilibria

Represent the states by the vectors $(1,0,0,0)$, $(0,1,0,0)$, $(0,0,1,0)$ and $(0,0,0,1)$ labeled in the natural order. The transformation matrices are;

$$\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \beta = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad \gamma = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\delta = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \text{ and } \iota = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

The incompleteness of the semiautomaton results in the transformation matrices having some rows that do not contain a “1.” The operation table for this machine’s semigroup is presented in table II-1. In this table, Θ denotes the zero matrix representing the null mapping. It can be shown that this semigroup is composed of several subsemigroups but no subgroups. Although the semigroup contains no subgroups, it is not at the lowest end of semigroup complexities—in particular, it is neither abelian nor idempotent and, therefore, is not a semilattice or a band. Thus, the Baron-Kalai automaton is “simple”—not because its semiautomaton has only four states but because its algebra has no subgroups and a small number of subsemigroups. In comparison to the strategies considered in section 5.1, the Baron-Ferejohn strategy’s complexity is between that of tit-for-tat and tweedledum (recall that tit-for-tat is a two-element idempotent semigroup and tweedledum is a two-element subgroup). To assure that there are not simpler strategies, Baron and Kalai need to demonstrate that there are no strategies with fewer or simpler subsemigroups (e.g., subsemigroups that are abelian and idempotent).

REMARK AII-1: This incomplete machine can be completed by introducing a sink state “5” and completing all partial functions by a mapping to the sink state. This construction increases the number of states but it does not change the semiautomaton’s algebraic complexity.

REMARK AII-2: A more complex machine can be created on these same four states in several ways. One method for increasing the complexity is to define γ and δ so that

they both map state 3 to state 1 rather than be undefined at state 3. This modification endows the machine semigroup with two cyclic subgroups, thereby increasing its complexity.

	\emptyset	α	β	γ	δ	ι	$\alpha\gamma$	$\alpha\delta$	$\beta\gamma$	$\beta\delta$	$\beta\iota$
\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset
α	\emptyset	α	\emptyset	$\alpha\gamma$	$\alpha\delta$	ι	$\alpha\gamma$	$\alpha\delta$	\emptyset	\emptyset	\emptyset
β	\emptyset	β	\emptyset	$\beta\gamma$	$\beta\delta$	$\beta\iota$	$\beta\gamma$	$\beta\delta$	\emptyset	\emptyset	\emptyset
γ	\emptyset	\emptyset	α	ι	\emptyset	\emptyset	\emptyset	\emptyset	$\alpha\gamma$	$\alpha\delta$	ι
δ	\emptyset	\emptyset	α	\emptyset	ι	\emptyset	\emptyset	\emptyset	$\alpha\gamma$	$\alpha\delta$	ι
ι	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset
$\alpha\gamma$	\emptyset	\emptyset	α	ι	\emptyset	\emptyset	\emptyset	\emptyset	$\alpha\gamma$	$\alpha\delta$	ι
$\alpha\delta$	\emptyset	\emptyset	α	\emptyset	ι	\emptyset	\emptyset	\emptyset	$\alpha\gamma$	$\alpha\delta$	ι
$\beta\gamma$	\emptyset	\emptyset	β	$\beta\iota$	\emptyset	\emptyset	\emptyset	\emptyset	$\beta\gamma$	$\beta\delta$	$\beta\iota$
$\beta\delta$	\emptyset	\emptyset	β	\emptyset	$\beta\iota$	\emptyset	\emptyset	\emptyset	$\beta\gamma$	$\beta\delta$	$\beta\iota$
$\beta\iota$	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset

TABLE II-1: Operation table for Baron-Kalai semiautomaton for playing majority-rule cake-division game.

REFERENCES

- D. Abreu and A. Rubinstein, "The Structure of Nash Equilibrium in Repeated Games with Finite Automata," *Econometrica* (1988), 1259-1281.
- E. F. Assmus, Jr. and J. J. Florentin, Algebraic Machine Theory and Logical Design, in "Algebraic Theory of Machines, Languages, and Semigroups," Academic Press, New York, (1968), 15-35.
- J. S. Banks and R. K. Sundaram, Repeated games, finite automata and complexity, *Games and Economic Behavior* 2 (1990), 97-117.
- D. Baron and J. Ferejohn, Bargaining in legislatures, *American Political Science Review* 83 (1989), 1181-1206.
- D. Baron and E. Kalai, The simplest equilibrium of a majority-rule division game, *Journal of Economic Theory* 61 (1992), 290-301.
- E. Ben-Porath, The complexity of computing a best response automaton in repeated games with mixed strategies, *Games and Economic Behavior* 2 (1990), 1-12.
- K. G. Binmore and L. Samuelson, Evolutionary stability in repeated games played by finite automata, *Journal of Economic Theory* 57 (1992), 278-305.
- A. H. Clifford and G. B. Preston, *The Algebraic Theory of Semigroups, Vol. 1*, Providence, American Mathematical Society (1961).
- R. A. Dean, "Elements of Abstract Algebra," John Wiley & Sons, New York, 1966.
- R. A. Dean, "Classic Abstract Algebra," Harper & Row, New York, 1990.
- S. Eilenberg, *Automata, Languages and Machines, Vol. A*, New York, Academic Press (1974).
- S. Eilenberg, *Automata, Languages and Machines, Vol. B*, New York, Academic Press (1976).
- C. Futia, The complexity of economic decision rules, *Journal of Mathematical Economics* 4 (1977), 289-299.
- I. Gilboa, The complexity of computing best response automata in repeated games, *Journal of Economic Theory* 45 (1988), 342-352.
- H. W. Gottinger, Complexity in social decision rules, in "Decision Theory and Social Ethics," (H. W. Gottinger and W. Leinfellner, eds.), D. Reidel, Dordrecht, Holland, (1978), 251-269.
- H. W. Gottinger, "Coping with Complexity," D. Reidel, Dordrecht, Holland, (1983).

- W. M. L. Holcombe, "Algebraic Automata Theory," Cambridge University Press, Cambridge (1982).
- J. E. Hopcroft and J. D. Ullman, "Introduction to Automata Theory, Languages, and Computation," Addison-Wesley, Reading, (1979).
- M. R. Johnson, Information, associativity and choice requirements, *Journal of Economic Theory* **52** (1990), 440-452.
- M. R. Johnson, The complexity of economic choice, mimeo presented at the North American Summer Meetings of the Econometric Society, Québec City, Canada (June, 1994).
- M. R. Johnson, Ideal structures of path independent choice functions, *Journal of Economic Theory* **65** (1995), 468-504.
- E. Kalai, Bounded rationality and strategic complexity in repeated games, in "Game Theory and Applications," (T. Ichiishi, A. Neyman, Y. Tauman, eds.) Academic Press, San Diego (1990), 131-157.
- E. Kalai and W. Stanford, Finite Rationality and interpersonal complexity in repeated games, *Econometrica* **56** (1988), 397-410.
- K. B. Krohn and J. L. Rhodes, Algebraic theory of machines, *Proceedings of Symposium on the Mathematical Theory of Automata*, New York: Polytechnic Institute of Brooklyn, (1962) 341-378.
- K. B. Krohn and J. L. Rhodes, Algebraic Theory of Machines, I Prime decomposition theorem for finite semigroups and machines," *Transactions of the American Mathematical Society* **116** (1965), 450-464.
- B. L. Lipman and S. Srivastava, Informational requirements and strategic complexity in repeated games, *Games and Economic Behavior* **2** (1990), 273-290.
- A. Nerode, Linear automaton transformations, *Proceedings of the American Math Society* **9** (1958) 541-544.
- A. Neyman, Bounded complexity justifies cooperation in the finitely repeated prisoner's dilemma, *Economics Letters* **19** (1985), 227-229.
- M. J. Osborne and A. Rubinstein, "A Course in Game Theory," MIT Press, Cambridge, (1994).
- S. E. Page, Two measures of complexity, paper presented at NBER Decentralization Conference, University of Toronto, Toronto, Canada, April 1994.
- C. H. Papadimitriou, On players with a bounded number of states, *Games and Economic Behavior* **4** (1992), 122-131.

J. Rhodes, Axioms for complexity for all finite semigroups, *Advances in Mathematics* **11** (1973), 210-214.

A. Rubinstein, Finite automata play the repeated prisoner's dilemma, *Journal of Economic Theory* **39** (1986), 83-96.

D. G. Saari, Mathematical complexity of simple economics, *Notices of the American Mathematics Society* **42** (1995), 222-230.

B. M. Schein, "The minimal degree of a finite inverse semigroup," *Transactions of the American Mathematical Society*, **333**, (1992) 877-888.

W. G. Stanford, "Some simple equilibria in repeated games," mimeo University of Illinois at Chicago, (August, 1987).

B. Tilson, Decomposition and complexity of finite semigroups, *Semigroup Forum* **3** (1971), 189-250.

S. J. Turnbull, Organizations as teams of automata, *Games and Economic Behavior* **7** (1994), 116-138.