

# The Structure and Two Complexities of Economic Choice Semiautomata

Mark R. Johnson  
Department of Economics  
W. P. Carey School of Business  
Arizona State University  
Tempe, AZ 85287  
email: mark.johnson.1@asu.edu

March 9, 2004

## Abstract

The structure of choice implementing semiautomata is characterized. In contrast to the more commonly used directed graph representation of the semiautomaton, the primary representation here is as a transformation semigroup. For convenience, the means for determining the directed graph representation also is presented. Associated with the choice implementing semiautomata are two complexities; algebraic complexity, which is linked to the mathematical power required to implement a choice rule and computational complexity, which is determined by the difficulty of constructing a particular choice semiautomaton. Each of these complexities is demonstrated to be associated with the type of consistency axiom satisfied by the choice function being implemented by the semiautomaton. Moreover, these two complexities are demonstrated to be dual to each other.

**Keywords:** complexity, algebraic complexity, choice functions, automata

I thank Richard A. Dean for many helpful conversations. Many of the results presented here are based on our joint work. Early versions of the paper were presented at the Econometric Society North American Meetings, College Park 2001.

# 1 Introduction

With increasing frequency, economists encounter issues of “complexity” in addressing research questions. While the topic has been on the economic agenda since Simon’s early work much of the current interest in complexity stems from more recent work in problems arising from game theory. There, as in other economic investigations where “complexity” arises, investigators have identified not one but several complexities.<sup>1</sup> In the following an automaton model, or more precisely, a semiautomaton model, is used to investigate complexity in individual choice models.<sup>2</sup> The first step in addressing complexity issues inherent in economic choice models is to characterize the structure of an economic choice semiautomaton. Only once this structure has been identified can the complexity questions be investigated. Early results accomplish the task of identifying the structure of choice semiautomata and specify the relationship between classes of choice functions and properties of the choice semiautomaton. Subsequent results identify two distinctly different complexities — algebraic complexity and computational complexity — and differentiate between them. Intriguingly, while these complexities are equally relevant to consumer choice models, there is a natural sense in which they are inversely related. Fundamental to obtaining these results is the fact that the semiautomaton is modeled as a transformation semigroup rather than as a directed graph and that algebraic techniques are used both to characterize the structure and to expose complexity differences.

While there are multiple economic uses of the term “complexity”, two arise most often. Perhaps the most common use of the term is what is called “computational” complexity. This is the complexity of solving a specific problem. The computational complexity is determined by some measure of the effort required to solve the problem. In consumer choice for the case where preferences are representable by a linear order, the problem of determining the “right” linear ordering of a set of  $n$  objects is a classic example. The presumption that the objects are, in fact, linearly ordered by some criterion like the individual’s preferences may be suspect but, given that assumption, the difficulty of determining which linear order is the correct one has economic implications. For this class of problem, a natural “fundamental operation” might be taken as a binary comparison in which two alternatives are compared and one of them is determined to be preferred to the other. A sequence of such comparisons will reveal the true linear ordering of the preferences.<sup>3</sup> The number of these comparisons is the “computational complexity” of solving the problem of which linear ordering is correct. In the following, the computational complexity scale applied to choice functions is classic in that it depends on the number of “prime intervals” that must be identified in order to construct a representation of a particular choice function.

Given a particular linear order preference it might seem straight forward to determine the

---

<sup>1</sup>Even a cursory look at game theory reveals at least three distinct applications of the term “complexity”. Specifically, (i) Rubinstein[36], Abreu and Rubinstein [1], Neyman [33] Banks and Sundaram [3], Kalai [25], Kalai and Stanford [26] use complexity to refer to the requirements of implementing a strategy in a repeated play game (ii) Lipman and Srivastava [30] have considered the complexity of the strategy itself and (iii) Gilboa [13], Ben-Porath [4] and Papadimitriou [34] use complexity to refer to the difficulty of computing the best response in game situations.

<sup>2</sup>A semiautomaton differs from an automaton in that the semiautomaton does not specify either initial or terminal states while an automaton specifies both initial and terminal states.

<sup>3</sup>The process described here is a loose description of what is called a “binary sort”. There are other, “more efficient”, sorting processes (e.g., merge-insertion) which more closely approximate theoretical limits. Independent of the specific algorithm, the number of comparisons required grows roughly on the order of  $n \ln(n)$ . See Knuth [28] for a discussion of sorting algorithms, their limits and history.

choice from any set. The problem of finding the choice from a set, however, is not without its own complexity. This is the “algebraic complexity” of choice, the mathematical power required to effect the choice. Papadimitriou [34] calls this “implementation” complexity. Given the information about which linear order is correct, how is the choice actually made from each subset? If the ordering is represented by its incidence matrix, one way of determining the choice from any subset is to take row sums of each alternative in the feasible set; the element with the largest row sum is the chosen element. This summing up does not come free, however, it requires some “power”. In particular, not all automata are capable of “counting”.<sup>4</sup> Moreover, the power to count or sum is independent of the number of states in the semiautomaton (Johnson [17]). Accordingly, the concept of algebraic complexity used here is based on the “mathematical power” of the algebra associated with the semiautomaton and not the number of states in the semiautomaton.<sup>5</sup> Indeed, the results demonstrate that the mathematical systems associated with different classes of choice implementing automata are nested so that there is no ambiguity about the relative “power” of the systems.

While starting with the common foundation of automata theory, this paper differs in several respects from most other economic applications of automata theory. Most important, rather than asserting that an automaton implementing economic choices exists, here a precise characterization of the structure of choice semiautomata is presented. Second, the focus here is to determine the algebraic and computational complexities of semiautomata constrained to satisfy specific economic consistency axioms like the weak axiom of reveal preference. Third, most applications of automata theory represent the automata in their finite state form. In that form, the semiautomaton consists of states, environmental inputs (or stimuli) and partial functions that dictate the result of applying a particular stimulus to a specific state. In contrast, this investigation represents the semiautomaton as a transformation semigroup consisting of an underlying set, an action semigroup and an action. The transformation semigroup is directly linked to classical automata theory in the sense that every semiautomaton (the basic building block for any computer system, e.g., automata, Moore machines, Turing machines, etc.) defines a transformation semigroup and for every transformation semigroup, there exists at least one semiautomaton that defines that semigroup.

The approach to understanding the economic impact of complexities is through analysis of semiautomata constrained to implement choice rules meeting different consistency axioms. There are several rationalizations for this approach, (1) a consumer decides to adopt a “rule of thumb” for making choices and the semiautomaton represents the rule of thumb, (2) the individual delegates his choices to a robot that does the actual shopping for him and the semiautomaton contains the instructions for making choices and (3) the consumer “acts as if” they were a semiautomaton implementing choices according to a particular consistency axiom.<sup>6</sup> In this context, then, the questions of “algebraic complexity” and “computational complexity” may be phrased, respectively, as: “Given a specific class of choice function, how powerful a class of mathematical system is required in order to implement it?” and, “How hard is it to construct the automaton that will implement a choice function of a particular type?” These are distinctly different questions and, as noted above, it will be shown that choice automata with high algebraic complexity can have

---

<sup>4</sup>Not all mathematical systems are capable of counting and specifically not those arising from path independent choice functions. This limitation of the choice implementing automata results from the fact that the relevant algebra is idempotent.

<sup>5</sup>This application of the term “mathematical power” is adopted from Kalai and Stanford [26].

<sup>6</sup>Rational (3) is effectively what we do when we represent consumers by utility functions and find constrained optima.

low computational complexity while choice automata with low algebraic complexity have high computational complexity.

Four classes of choice function are considered; (i) those rationalized by linear orders, (ii) those rationalized by weak orders, (iii) those rationalized by quasi-transitive relations and (iv) path independent choice functions. The results demonstrate that the classes are ranked by algebraic complexity as depicted below.

$$\left( \begin{array}{c} \text{linear order} \\ \text{choice functions} \end{array} \right) < \left( \begin{array}{c} \text{weak order} \\ \text{choice functions} \end{array} \right) < \left( \begin{array}{c} \text{quasitransitive} \\ \text{choice functions} \end{array} \right) < \left( \begin{array}{c} \text{path independent} \\ \text{choice functions} \end{array} \right)$$

In this ranking the non-rational path independent choice functions require the highest level of mathematical power to implement. Notably, for the subset of choice functions common between the two papers, this ranking agrees with the information processing cost ranking obtained in Johnson [15].<sup>7</sup>

When computational complexity is considered, the ranking is reversed and has the linear order choice functions posing the highest computational complexity demands. Thus, choice functions rationalized by a linear order have the lowest algebraic complexity and the highest computational complexity while non-rational path independent choice functions have the highest algebraic complexity and the lowest computational complexity. This situation is depicted below.

$$\left( \begin{array}{c} \text{linear order} \\ \text{choice functions} \end{array} \right) > \left( \begin{array}{c} \text{weak order} \\ \text{choice functions} \end{array} \right) > \left( \begin{array}{c} \text{quasitransitive} \\ \text{choice functions} \end{array} \right) > \left( \begin{array}{c} \text{path independent} \\ \text{choice functions} \end{array} \right)$$

In addition to identifying the duality of algebraic and computational complexities for choice implementing semiautomata, the results presented below provide the foundation for several different applications. First, the broad class of semiautomata strictly contains the semiautomata implementing path independent choice functions (Johnson [15]). As such, it is a natural tool for extending the economic model of either individual or collective decision making on finite domains to cases beyond those assuming path independence. In particular, because the transitions are defined relative to each state, the semiautomaton model is ideally formed for decision problems where the status quo is an important decision variable. If that direction is pursued then the results presented here provide a baseline against which to compare those broader models of choice. Similarly, the model can easily be formed to allow the final decision to depend on the sequence in which the feasible set is expanded or contracted. Thus histories can matter. A simple demonstration of this feature is offered in section 3.2.2 where the powers required to implement rational and non-rational choice functions are

---

<sup>7</sup>In that ranking, Johnson did not make a distinction between quasitransitive rational Path Independent (PI) choice functions and those that were not rationalizable. Johnson [16] did make a distinction between quasitransitive path independent choice functions and non-rationalizable PI choice functions. In that case, Johnson demonstrated that the algebra associated with quasitransitive rational PI choice functions was a subclass of the algebra associated with PI choice functions. In Johnson [15] the intuition was based on rational choice functions and used the number of binary comparisons required to determine the choice as the feasible set expanded. The number of these comparisons was given the intuition of the “bit cost” of preserving path independence. Further, for path independent choice functions, Johnson identified the existence of a “representative element” as a difference between choice functions satisfying the Weak Axiom of Revealed Preference as opposed to Path Independence.

compared. Further, having a semiautomaton model of an individual “economic” decision maker provides a basic building block for constructing automata theoretic models of organizations. By specifying how the basic units are interlinked and aggregated, several different larger automata can be constructed. Depending on how the aggregations are specified, different organizational structures such as firms can be modeled and their properties, such as their complexities as well as other features such as the flow of information through the structures, can be compared. Finally, the algebraic structures of the the choice implementing semiautomata identified here can be compared directly to those derived from strategy implementing semiautomata. The foundations for such a comparison are presented in Johnson [17]. In that comparison, only the grim trigger implementing semiautomaton has an algebraic complexity as low as that required to implement path independent choice.

Section 2 provides the basic definitions and notation for choice functions and the algebraic structures used in this paper. The form, structure and algebraic complexity of the transformation semigroups for choice semiautomata are described in section 3. This section highlights two aspects of choice semiautomata demonstrating the presence of both simplifying aspects of economic assumptions about choice behavior and concrete differences in the relative powers of automata implementing choice functions satisfying different consistency axioms. While, as noted above, the number of states is not used to indicate algebraic complexity, one of the simplifying features of economic choice allows a reduction in the number of states in the underlying set. The state reduction is demonstrated in this section. One of the main differences in the mathematical power required to implement different classes of choice functions is demonstrated by observing that each choice function can be decomposed into two choice semiautomata and considering the interaction between these two semiautomata. An example comparing the difference in how the two semiautomata associated with rational and non-rational choice functions interact concludes the section. Section 4 introduces the concepts and definitions for computational complexity and the results arising from application to economic choice. As is typical for computational complexities, algorithms are involved but the final result rests on concepts more fundamental than a particular algorithm. Conclusions are discussed in section 5.

## 2 Definitions and notation

The technical tools used in the results presented below are choice functions and the elementary algebra of sets, primarily semigroups and lattices. The definitions and prerequisites of choice functions and consistency requirements on choice functions are presented in section 2.1. Algebras are covered in section 2.2. Semiautomata and the links to algebra are covered in 2.3.

### 2.1 Notation, Choice Functions and Consistency Requirements

The universal set  $V$  is composed of a finite number of distinct alternatives and  $2^V$  is the power set of  $V$ . Subsets of  $V$ , denoted by  $v$  are elements of  $2^V$ . Unless otherwise stated, the cardinality of  $V$ , denoted by  $|V|$ , is  $t$ , and the cardinality of  $v \in 2^V$  is  $n$ ; note  $n \leq t$ . Distinct subsets of  $V$  are subscripted with an integer  $i \in \{1, \dots, 2^V\}$ ; where  $\{v_i\} = \{v_j\}$  if and only if  $i = j$ .

A *choice function* is a mapping  $C : 2^V \rightarrow 2^V$ , such that  $C(v) \subseteq v$  and  $C(v) = \emptyset$  if and only if  $v = \emptyset$ . A choice function  $C$  is *rational* if and only if there exists a relation  $R$  such that, for every  $v \in 2^V$ ,  $C(v) = G(v; R)$  where,  $G(v; R) = \{x \in v \mid xRy, \forall y \in v\}$ . The function  $G(v; R)$  selects the  $R$ -maximal elements. A choice function  $C$  defined on  $V$  is defined as *discriminating* if there is some  $v \in 2^V$  for which  $C(v) \neq v$ .

The classes of choice functions considered are those satisfying the Strong Axiom of Preference, the Weak axiom of Revealed Preference, the conjunction of Path Independence and Extension, and Path Independence (alone). The consistency axioms are defined formally as follows.

Strong Axiom of Preference (SAP):

- (i)  $\forall x, y \in V, x \in C(\{x, y\}) \Rightarrow y \notin C(\{x, y\})$ , and
- (ii)  $\forall v_1, v_2 \subseteq V, v_1 \subseteq v_2 \Rightarrow \{v_1 \cap C(v_2)\} = \begin{cases} \emptyset, \text{ or} \\ C(v_1) \end{cases}$ .

Weak Axiom of Revealed Preference (WARP)

$$\forall v_1, v_2 \subseteq V, v_1 \subseteq v_2 \Rightarrow \{v_1 \cap C(v_2)\} = \begin{cases} \emptyset, \text{ or} \\ C(v_1) \end{cases}.$$

Rational Path Independence

- (i)  $\forall v_1, v_2 \subseteq V, C(C(v_1) \cup C(v_2)) = C(v_1 \cup v_2)$ , and
- (ii) Extension (E):  $\forall v \subseteq V, (x \in v \text{ and } (\forall y_{y \in v}, x \in C(\{x, y\})) \Rightarrow x \in C(v))$ .

Path Independence

$$\forall v_1, v_2 \subseteq V, C(C(v_1) \cup C(v_2)) = C(v_1 \cup v_2).$$

The first two of these classes always can be rationalized by a complete, reflexive and transitive binary relation (Arrow [2]). Choice functions satisfying the strong axiom are always single-valued and rationalized by linear orders while choice functions meeting WARP need not be single-valued and are rationalized by weak orders. The two classes of path independent choice functions are distinguished by whether or not they are rationalizable; choice functions satisfying both PI and E are rationalizable by a quasi-transitive relation while choice functions satisfying PI need not be rationalizable (Plott [35]).

## 2.2 Algebras

The definitions of binary systems and system properties are provided in terms of an arbitrary non-empty set  $N$ , which is used as both the domain and the 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  $\langle N; \cdot \rangle$ . Algebraic properties defined for all  $v_1, v_2, v_3 \in N$  are,

$$(B-1) \text{ Closure: } v_1 \cdot v_2 \in N,$$

$$(B-2) \text{ Associative: } v_1 \cdot (v_2 \cdot v_3) = (v_1 \cdot v_2) \cdot v_3 ,$$

$$(B-3) \text{ Commutative: } v_1 \cdot v_2 = v_2 \cdot v_1 ,$$

$$(B-4) \text{ Idempotence: } v_i \cdot v_i = v_i ;$$

A binary system satisfying (B-1) and (B-2) is called a *semigroup* and a semigroup satisfying (B-3) is called a *commutative semigroup*. A semigroup for which every element satisfies (B-4) is called an *idempotent semigroup*. A commutative idempotent semigroup has a representation as a *semilattice* under the natural partial ordering of the semigroup where the *natural partial ordering* is defined as follows,  $a \cdot b = b \Leftrightarrow a \leq b$ .<sup>8</sup>

Here, the power set of the universal set  $V$  is used as both the domain and the range and the binary operation  $(\bullet)$  is adopted from Plott [35]. Formally,  $\bullet : 2^V \times 2^V \rightarrow 2^V$ , where  $\forall v_1, v_2 \in 2^V, v_1 \bullet v_2 = C(C(v_1) \cup C(v_2))$ . The binary system for  $V$  under the operation  $(\bullet)$  is denoted by  $\langle 2^V; \bullet \rangle$ . Plott [35] proved that this system is a commutative semigroup.

In addition to the properties of the operation it is useful to identify two special members of binary systems.

**Definition 2.1** Given a binary system  $T = \langle N; \cdot \rangle$  an element  $z$  such that  $x \cdot z = z \cdot x = z, \forall x \in T$  is called a zero, and an element  $e$  such that  $t \cdot e = e \cdot t = t, \forall t \in T$  is called an identity.

A semigroup that has an identity is a *monoid*. An idempotent commutative monoid with a zero is a *lattice*. Johnson [16] identified a subsemigroup of Plott's semigroup that has precisely these properties. Further Johnson conjectured that this semigroup might be relevant to economic applications of automata theory. The results below validate that conjecture. While initially identified by means of Plott's single operation  $(\bullet)$  lattices actually have two operations, typically called the *join* denoted by  $\vee$  and the *meet* denoted by  $\wedge$ . A lattice  $L$  is denoted by  $\langle L; \vee, \wedge \rangle$ . An element  $x$  in a lattice is called *join-irreducible* if  $x \vee y = a$  implies  $x = a$  or  $y = a$ . By convention bottom elements of a lattice are not called join-irreducible. Dually, an element  $y$  in a lattice is called *meet-irreducible* if  $a \wedge b = y$  implies  $y = a$  or  $y = b$ . For both the join irreducibles or the meet irreducibles the partially ordered set of irreducibles  $\langle P; \leq \rangle$  will be important. In a partially ordered set  $P$ ,  $x$  covers  $y$  if  $x > y$  and for no  $a \in P$   $x > a > y$ . Lattices are well covered in such classics as Birkhoff [5], however a few especially useful properties are summarized here. One important property of some lattices is the distributive law. A lattice  $\langle L; \vee, \wedge \rangle$  is a *distributive lattice* if it satisfies the *distributive law*:

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c) \text{ for all } (a, b, c \in L).$$

---

<sup>8</sup>The natural partial ordering is adopted from Clifford and Preston [7] [8].

Given a lattice  $L$  with a zero  $0$  and an identity  $1$ , and some element  $a \in L$ , for which there is an element  $b \in L$  such that  $a \wedge b = 0$  and  $a \vee b = 1$  then  $a$  is said to have a *compliment*. If  $a$  has a unique compliment, then the compliment is denoted by  $a'$ . Taking the compliment is a *unary* operation. A *Boolean algebra* is a system  $\langle B; \wedge, \vee, ', 0, 1 \rangle$  such that (i)  $\langle B, \wedge, \vee \rangle$  is a distributive lattice, (ii)  $a \wedge 1 = a$  and  $a \vee 0 = a$  for all  $a \in B$ , and (iii)  $a \wedge a' = 0$  and  $a \vee a' = 1$  for all  $a \in B$ . The finite Boolean algebras considered here are isomorphic to  $2^V$  under set union, intersection and complementation.

Within the Boolean algebra  $2^V$  for sets  $V \supseteq T \supseteq B$ , the collection of sets  $K$  such that  $T \supseteq K \supseteq B$  is called an *interval* and the interval is denoted by  $T/B$ .  $T$  is the *top* of the interval and  $B$  is the *bottom* of the interval. An interval  $T/B$  is called *proper* if  $T \neq B$ . If  $T = B \cup \{x\}$  then  $T$  *covers*  $B$  and the interval  $T/B$  is a *prime* interval.

A particular class of lattices initially identified by Dilworth [9] and now known as *lower locally distributive lattices* (LLDs) is relevant for choice functions. A lattice is an LLD if every element in the lattice has a unique irredundant representation as the join of join-irreducibles. Johnson and Dean [18] [19] and independently, Koshevoy [29] demonstrated a direct link between path independent choice functions and LLD lattices in that every PI choice function has a representation as an LLD lattice and for every LLD lattice, there is an associated PI choice function.<sup>9</sup> Further Johnson and Dean demonstrated characterization results between the predominant classes of PI choice functions and subclasses of LLD lattices. Significantly, not all of these LLD lattices are distributive.

### 2.3 Semiautomata, Transformation Semigroups and Action

Although employed here only as a link to other literature, a common means for representing a semiautomaton, or finite state machine, in economics is through the directed graph. A semiautomaton  $\mathcal{M} = (Q, \Sigma, F)$  consists of a finite number of states  $Q$ , an alphabet  $\Sigma$  and partial functions  $F$ . In the directed graph, the states become the vertices, the partial functions  $F$  are the edges and the alphabet labels the edges.<sup>10</sup>

The fundamental semigroups of algebraic automata theory are the transformation semigroup and the action semigroup (see Eilenberg [10] [11] or Holcombe [24]). These semigroups are defined as follows.<sup>11</sup>

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  $S$  is a 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  $S$  and the elements of  $S$  are called the *transformations* of  $X$ . If an arbitrary semigroup  $S$  is not a subsemigroup of  $PF(Q)$ , the  $S$  may be embedded in  $PF(Q)$  if there exists an

<sup>9</sup>Koshevoy [29] used convex geometries to obtain results related to a subset of the Johnson and Dean results. Here the full range of the Johnson and Dean characterizations are used.

<sup>10</sup>For perspective, the more commonly employed automaton is a semiautomaton that has been augmented by identification of an *initial* state  $i$  and a collection of *terminal* states  $T$ . Thus an automaton  $\mathcal{A} = (\mathcal{M}, i, T)$ .

<sup>11</sup>This summary borrows from Eilenberg [11].



action  $\alpha : Q \times S \rightarrow Q$  satisfying two conditions for  $s, s' \in S$  and  $q \in Q$ <sup>12</sup>

(a)  $((q, s)\alpha, s')\alpha = (q, ss')\alpha$

(b)  $s \neq s'$  implies  $qs \neq qs'$  for some  $q \in Q$ .

Both the transformation semigroup and the action semigroup are important items in the study of automata theory. However, while the transformation semigroups characterize semiautomata, all the mathematical power or algebraic complexity is contained in the action semigroup (Eilenberg [11]). For this reason, much of the remaining analysis is focused on the action semigroup.

### 3 Structure and Algebraic Complexity of Choice Semiautomata

The structure of choice automata is introduced in this section. Construction of the choice automaton is accomplished by use of Eilenberg's embedding technique [11]. The semigroup used for the embedding is the previously mentioned subsemigroup of Plott's semigroup identified by Johnson [16] and characterized by Johnson and Dean [18] [19]. Johnson and Dean demonstrate that for a path independent choice function  $C$  the image of the domain  $2^V$  under the mapping  $C$  is a lower locally distributive lattice. The elements of this lattice are the idempotent members of Plott's semigroup. Given a specific Plott semigroup  $\langle 2^V; \bullet \rangle$  construct the idempotent subsemigroup  $J = \langle I(V); \bullet \rangle$  as follows. First, define  $I(V) = \{v_i \in 2^V \mid \exists v_j \in 2^V \ni v_i = C(v_j)\}$  so that the members of  $I(V)$  are those members of the domain that are chosen from some set. The operation  $(\bullet)$  used for  $J$  is carried over from Plott's semigroup and  $J$  is used to signify the lattice join operation. Proposition 1 verifies that Plott's operation  $(\bullet)$  is an action that allows  $J = \langle I(V); \bullet \rangle$  to be embedded in the semigroup of partial functions defined on  $2^V$  and thus guarantees that  $T = \langle 2^V; J \rangle$  is a transformation semigroup.

**Proposition 3.1** *Let  $C$  satisfy PI on  $V$  and let  $(\bullet)$  be as defined above, then*

1.  $(\bullet) : 2^V \times J \rightarrow 2^V$  is an action, and
2.  $T = \langle 2^V; J \rangle$  is a complete transformation semigroup

*Proof:* See Appendix I

Observe that although the action  $(\bullet)$  is defined as a mapping from  $2^V \times J$  to  $2^V$ , in fact the result of the mapping always will be an element in  $I(V)$ .

From this structure the finite state machine representation is easily recovered by defining  $\mathcal{M} = \langle 2^V, J, F \rangle$  where  $F : 2^V \times J \rightarrow 2^V$  is given by  $F(v, j) = v \bullet j$  for all  $v \in 2^V, j \in J$ . Thus the states of the directed graph representation are the members of the domain and the alphabet

---

<sup>12</sup>Condition (a) is called the *associativity* condition while condition (b) is called the *faithfulness* condition.

$\Sigma$  is the subsets that can be chosen from some set in the domain (and, thus, members of  $J$ ) and, since the image of a  $v \in 2^V$  is an element of  $I(V)$  and that algebra is idempotent, these are the words of the machine as well.<sup>13</sup> The fact that the action semigroup is both idempotent and commutative means that the machine can not tell how many times it has seen a particular input (the machine can't count) and the machine can't tell in which order it has seen different inputs (the machine can't detect differences in sequence or pattern). These limitations are imposed on the semiautomata by the path independence assumption and not by the automaton model itself. Example 1a demonstrates the process for starting with a choice function and moving to the transformation semigroup representation of the semiautomaton.

**Example 1a:** Let the choice function  $C$  defined on  $V = \{1, 2, 3\}$  be specified as follows;

$$\begin{aligned} C(V) &= C(\{1, 2\}) = C(\{1, 3\}) = C(\{1\}) = \{1\} \\ C(\{2, 3\}) &= C(\{2\}) = \{2\}; C(\{3\}) = \{3\}; C(\emptyset) = \emptyset \end{aligned}$$

Then the idempotent elements are  $I(V) = \{\{1\}, \{2\}, \{3\}, \emptyset\}$ , the action semigroup is  $J = \langle I(V); \bullet \rangle$  and the transformation semigroup is  $T = (2^V; J)$ . $\diamond$

The technique for constructing the example 1a transformation semigroup easily is extended to any path independent choice function. As important as the specific constructions is the fact that the action semigroups for these choice automata are precisely the choice lattices characterized in Johnson and Dean [18] [19]. Thus the algebraic structure of the action semigroups associated with the major classes of path independent choice functions is fully characterized for finite sets. An important feature of those characterization results is a simplification of the choice process described below.

### 3.1 Simplicity and Structure

The first simplification evident in the transformation semigroup structure is the relationship between the underlying set (the feasible sets for the choice problem) and the action semigroup. In particular, for this structure, it is useful to look at the inverse image of the idempotent elements that define the action semigroup. For any particular idempotent element, this set is the collection of sets in  $2^V$  from which the idempotent element could have been chosen. For path independent choice functions, the remarkable fact is that this set is necessarily an interval in the Boolean algebra of the domain for the choice function (Johnson and Dean, [18] [19]). This interval identifies an equivalence relation on the states in the underlying set of the transformation semigroup. While this equivalence relation is defined on the states of the semiautomaton, lacking initial and terminal states, it is directly analogous to the equivalence relation defined on the states in the construction of the minimum number of state automaton assured by the Myhill-Nerode Theorem (Myhill [31], Nerode [32]).

---

<sup>13</sup>This description does not give the minimum number of state machine. The minimum number of state machine is presented in example 1b of section 3.1 after introduction of the interval property. In this minimum number of state machine, the states are the same as the alphabet. That is, both are the idempotent elements  $I(V)$ . For the example on three elements, the number of elements in the underlying set is reduced from eight to four.

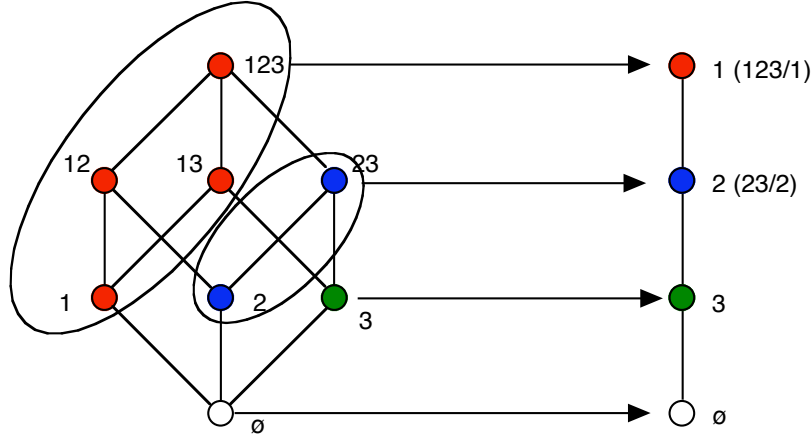


Figure 1: Interval property of Path Independent choice functions.

**Theorem 3.1 (Interval Property)** *For path independent choice functions  $C$ , the inverse sets of  $C$  are intervals in the Boolean algebra. For each  $A \subseteq V$  there exist subsets  $T$  and  $B$  such that*

$$\text{arc}(A) = \{Y \subseteq V : B \subseteq Y \subseteq T\}.$$

*The element  $T$  is called the top of the interval and  $B$  is called the bottom of the interval. The interval is denoted by  $T/B$ .<sup>14</sup>*

What this property does is assure that when the domain is represented as a Boolean algebra, all the sets from which a particular subset is chosen are between two elements in the domain of the choice function. This situation is depicted in figure 1 for the example 1a choice function.

In figure 1, the Boolean algebra on the left is the domain of the choice function and the chain on the right is both the image of the domain under  $C$  and the action semigroup of the automaton. The elements of the domain are coded so that all the members of the domain are shaded the same as the shade of the element in the range into which they are mapped. Here it can be seen that the inverse image of the idempotent  $\{1\}$  is the interval  $\{\{1\}, \{1, 2\}, \{1, 3\}, \{1, 2, 3\}\}$  denoted  $\{1, 2, 3\}/\{1\}$  where  $\{1, 2, 3\}$  is the top of the interval and  $\{1\}$  is the bottom of the interval. The only other non-trivial inverse image for this choice function is the interval  $\{2, 3\}/\{2\}$ . The interval property holds for all path independent choice functions.

While identifying the equivalence relation appropriate for constructing the partition of the states necessary for the minimum number of state automaton, there is nothing in the Myhill-Nerode theorem that requires those partitions to have such a convenient and easily identified structure as to be an interval. The impact of the interval property can be seen by examining Johnson's [15]

<sup>14</sup>Johnson and Dean (1996) (2001a)

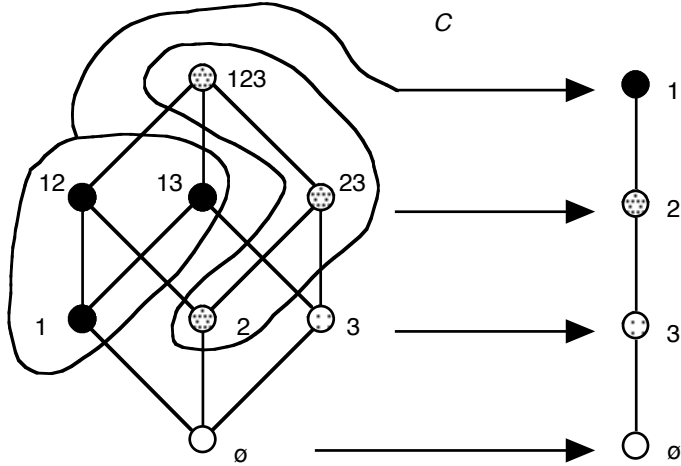


Figure 2: Failure of the interval property on a choice function that is not Path Independent

example of a semigroup for which the associated choice function is not path independent.<sup>15</sup>

**Example 2:** Let the choice function  $C$  defined on  $V = \{1, 2, 3\}$  be specified as follows,  $C(i) = \{i\}$ ,  $C(V) = \{2\}$ ,  $C(\{1, 2\}) = C(\{1, 3\}) = \{1\}$ ;  $C(\{2, 3\}) = \{2\}$ ;  $C(\emptyset) = \emptyset$ . The operation table for the associated semigroup of idempotent elements is depicted in figure 3.  $\diamond$

Examination reveals that the semigroup associated with the example 2 choice function is commutative and idempotent and has a representation as a chain, but the choice function is not path independent (Johnson [15]). When depicted as in figure 2, failure of the interval property is evident. It is possible that the interval property is a short hand making it easier to identify the sets from which a particular choice is made. Rather than having to remember every element in the relevant partition, just the top and bottom elements of the interval are sufficient to identify the entire partition.

As noted above, the interval property allows a reduction in the number of states in the underlying set of the transformation semigroup. Specifically, for a transformation semigroup  $T = (2^V; J)$  with  $I(V) = \{v_i \in 2^V | \exists v_j \in 2^V \ni v_i = C(v_j)\}$ , the underlying set  $2^V$  can be replaced by  $I(V)$  so that the “smaller”  $\bar{T} = (I(V); J)$  is the analogue to the minimum number of state machine in the case where the automaton can be started in any state and all states are terminal states.

**Example 1b:** For the choice function in example 1a, with  $I(V) = \{1, 2, 3, \emptyset\}$  and  $J = \langle I(V); \bullet \rangle$  the smaller transformation semigroup is  $\bar{T} = (I(V); J)$ . Observe that the number of states has

<sup>15</sup>At the time Johnson noted that the example was not capricious. Indeed, the example is precisely the case where the Condorcet winner is not chosen on the whole set.

	$\emptyset$	1	2	3	1,2	1,3	2,3	1,2,3
$\emptyset$	$\emptyset$	1	2	3	1	1	2	2
1	1	1	1	1	1	1	1	1
2	2	1	2	2	1	1	2	2
3	3	1	2	3	1	1	2	2
1,2	1	1	1	1	1	1	1	1
1,3	1	1	1	1	1	1	1	1
2,3	2	1	2	2	1	1	2	2
1,2,3	2	1	2	2	1	1	2	2

Figure 3: Operation table for semigroup associated with the Example 2 choice function

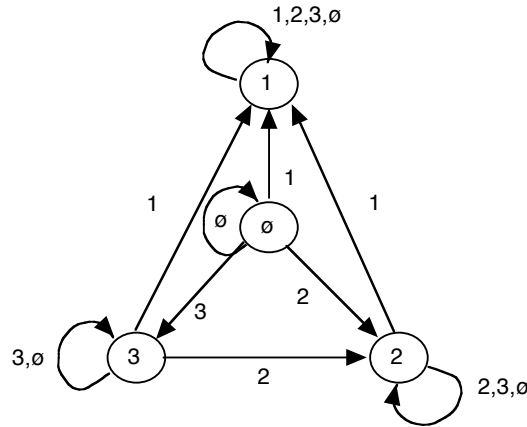


Figure 4: Directed graph representation of semiautomaton defined by  $\bar{T} = (I(V); J)$  in example 1b.

been reduced from the entire Boolean algebra of feasible sets to just  $I(V)$ . Thus the underlying set is reduced from eight ( $2^3$ ) states to four states. The directed graph representation of this semiautomaton is presented in figure 4.  $\diamond$

The relations between the consistency axioms satisfied by the choice function and the action semigroup of the associated choice semiautomaton are identified in Proposition 3.2. Representative examples of each type are presented in example 3 and figure 5. Of particular significance is the fact that the example lattices have been chosen so that each algebra, except the linear order, has precisely nine elements. The exception for the class of choice functions rationalized by linear orders where the associated algebra can be only a five element chain. The reason for having selected the representative lattices in this manner is to emphasize the fact that the class of mathematical system does not depend on the number of states in the action semigroup associated with the semiautomaton. Because the mathematical systems have different powers, the associated semiautomata have different abilities to implement choice rules.

**Proposition 3.2** *Let  $C$  satisfy PI on  $V$  and let  $(\bullet)$  be as defined above and let  $T = (2^V; J)$  be the*

complete transformation semigroup derived from  $C$ . Then

1.  $C$  satisfies PI if and only if  $J$  is an LLD lattice,
2.  $C$  satisfies PI and E if and only if  $J$  is a distributive lattice,
3.  $C$  satisfies WARP if and only if  $J$  is a chain of Boolean algebras, and
4.  $C$  satisfies SAP if and only if  $J$  is a chain.

*Proof:* See Appendix I

**Example 3:** Figure 5 depicts the lattice representations of the action semigroups for the following choice functions. These action semigroups are based on the following choice functions. The transformation semigroups are presented in their “small” forms.

*Example 3.1:* PI choice function with action semigroup that is the LLD lattice diagramed in figure 5.1:

$$C_P(\{1, 2, 3, 4\}) = C_P(\{1, 2, 4\}) = C_P(\{1, 3, 4\}) = C_P(\{1, 4\}) = \{1, 4\}, C_P(\{1, 2, 3\}) = C_P(\{1, 2\}) = \{1, 2\}, C_P(\{2, 3, 4\}) = C_P(\{2, 4\}) = \{2, 4\}, C_P(\{1, 3\}) = C_P(\{1\}) = \{1\}, C_P(\{3, 4\}) = C_P(\{4\}) = \{4\}, C_P(\{2, 3\}) = \{2, 3\}, C_P(\{3\}) = \{3\}, C_P(\{2\}) = \{2\}, C_P(\emptyset) = \emptyset.$$

Thus  $I_P(V) = \{\{2, 4\}, \{2, 3\}, \{1, 4\}, \{2\}, \{4\}, \{1, 3\}, \{1\}, \{3\}, \emptyset\}$  and the action semigroup is  $J_P = \langle I_P(V); \bullet \rangle$ .

*Example 3.2:* RPI choice function with action semigroup that is the distributive lattice diagramed in figure 5.2:

$$C_R(\{1, 2, 3, 4\}) = C_R(\{1, 2, 4\}) = C_R(\{2, 3, 4\}) = C_R(\{2, 4\}) = \{2, 4\}, C_R(\{1, 2, 3\}) = C_R(\{2, 3\}) = \{2, 3\}, C_R(\{1, 3, 4\}) = C_R(\{1, 4\}) = \{1, 4\}, C_R(\{1, 2\}) = C_R(\{2\}) = \{2\}, C_R(\{3, 4\}) = C_R(\{4\}) = \{4\}, C_R(\{1, 3\}) = \{1, 3\}, C_R(\{1\}) = \{1\}, C_R(\{3\}) = \{3\}, C_R(\emptyset) = \emptyset.$$

Thus  $I_R(V) = \{\{2, 4\}, \{2, 3\}, \{1, 4\}, \{2\}, \{4\}, \{1, 3\}, \{1\}, \{3\}, \emptyset\}$  and the action semigroup is  $J_R = \langle I_R(V); \bullet \rangle$ .

*Example 3.3:* WARP choice function with action semigroup that is the chain of Boolean algebras<sup>16</sup> diagramed in figure 5.3:

$$C_W(\{1, 2, 3, 4\}) = C_W(\{1, 2, 4\}) = C_W(\{1, 3, 4\}) = C_W(\{2, 3, 4\}) = C_W(\{1, 4\}) = C_W(\{2, 4\}) = C_W(\{3, 4\}) = C_W(\{4\}) = \{4\}, C_W(\{1, 2, 3\}) = \{1, 2, 3\}, C_W(\{1, 2\}) = \{1, 2\}, C_W(\{1, 3\}) = \{1, 3\}, C_W(\{2, 3\}) = \{2, 3\}, C_W(\{1\}) = \{1\}, C_W(\{2\}) = \{2\}, C_W(\{3\}) = \{3\}, C_W(\emptyset) = \emptyset.$$

<sup>16</sup>Here note that there are two Boolean algebras in this chain. The top Boolean algebra, composed of the elements labeled 1234/4 and 123, is isomorphic to  $2^1$  while the bottom Boolean algebra composed of the elements 123, 12, 13, 23, 1, 2, 3, and  $\emptyset$  is isomorphic to  $2^3$ . As assured by Johnson and Dean [18] [19] element 123 is “shared” by the two Boolean algebras in the chain.

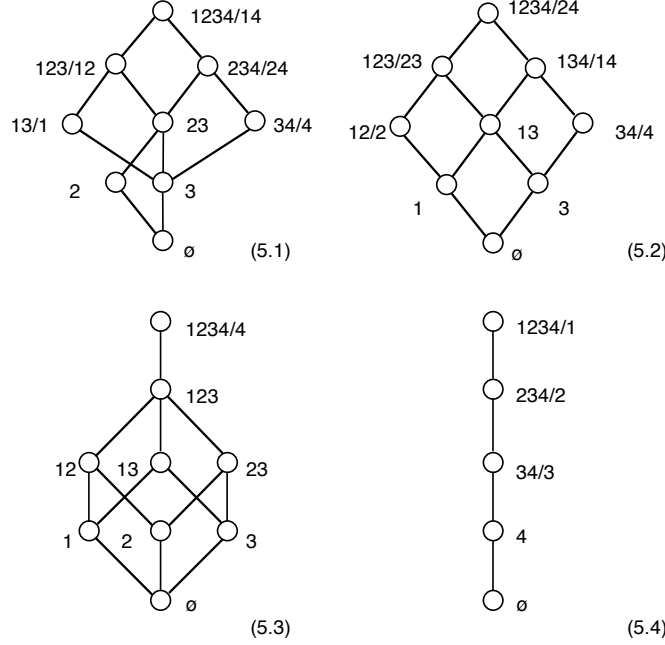


Figure 5: Figures 5.1, 5.2, 5.3 and 5.4. Action semigroups for Example 3 choice functions.

Thus  $I_W(V) = \{\{1, 2, 3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1\}, \{2\}, \{3\}, \{4\}, \emptyset\}$  and the action semigroup is  $J_W = \langle I_W(V); \bullet \rangle$ .

*Example 3.4:* SAP choice function with action semigroup that is the chain diagrammed in figure 5.4:

$$\begin{aligned}
 C_S(\{1, 2, 3, 4\}) &= C_S(\{1, 2, 3\}) = C_S(\{1, 2, 4\}) = C_S(\{1, 3, 4\}) = C_S(\{1, 2\}) = C_S(\{1, 3\}) = \\
 &C_S(\{1, 4\}) = C_S(\{1\}) = \{1\}, C_S(\{2, 3, 4\}) = C_S(\{2, 4\}) = C_S(\{2, 3\}) = C_S(\{2\}) = \\
 &\{2\}, C_S(\{3, 4\}) = C_S(\{3\}) = \{3\}, C_S(\{4\}) = \{4\}, C_S(\emptyset) = \emptyset.
 \end{aligned}$$

Thus  $I_S(V) = \{\{1\}, \{2\}, \{3\}, \{4\}, \emptyset\}$  and the action semigroup is  $J_S = \langle I_S(V); \bullet \rangle$ .  $\diamond$

There are several features of these examples that demand attention. First, as noted above, with the exception of the chain associated with the choice function rationalized by a linear, all of the lattices have nine members. Second, despite this commonality in the number of elements in the algebras, the systems have substantially different powers. Significantly (a point that is expanded on in the next section), the example 3.1 algebra is a non-distributive lattice while all the other examples are distributive. Third, most would agree that, of all the diagrams, the chain appears to be visually the “simplest”. Indeed, the lattice diagrams are arguably simpler as they proceed from figure 5.1 through 5.4. There are solid foundations for this apparent relationship.

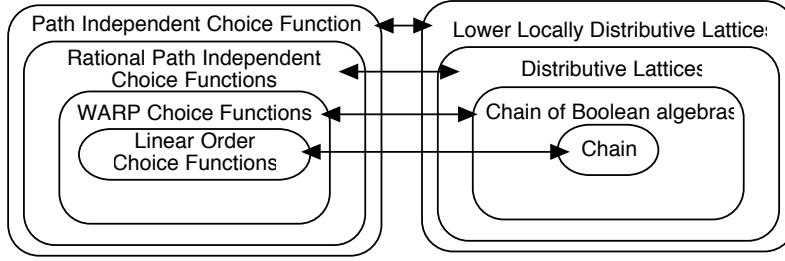


Figure 6: Lattice class containments for action semigroups associated with different consistency axioms.

Proposition 3.2 provides a convenient and intuitively appealing means for categorizing the mathematical power (and, hence, algebraic complexities) of choice semiautomata. While there are several means for “measuring” the complexity of these semiautomata such as the number of join irreducibles in the lattices, a more direct means of reflecting the relative powers of the semiautomata by the class of the lattice each of the consistency axioms imposes on the action semigroup. In particular, lattices arising from PI choice functions are lower locally distributive lattices and, as such, strictly contain the class of distributive lattices. In turn, distributive lattices strictly contain the class of chains of Boolean algebras. Naturally, a chain is a special case of a chain of Boolean algebras in which each Boolean algebra is a singleton. These relationships are depicted in figure 6. Notice that each of the classes of lattices are conveniently nested so that there is no ambiguity about the relative mathematical power of the action semigroups.

To formalize this intuition let each of the classes be labeled as:

$$P = \{\text{class of lower locally distributive lattices}\},$$

$$D = \{\text{class of distributive lattices}\},$$

$$W = \{\text{class of chains of Boolean algebras}\},$$

$$S = \{\text{class of chains}\}.$$

For each pair of these, a class algebraic complexity comparison can be identified.

**Definition 3.1** Given  $A$  and  $B$ , two classes of LLD lattices,  $L(A)$  the algebraic complexity of the class  $A$  is greater than  $L(B)$ , denoted as  $L(A) > L(B)$ , if and only if  $B$  is a strict subclass of  $A$ .

**Definition 3.2** Given two choice functions  $C^1$  and  $C^2$  with transformation semigroups  $T^1 = (2^V; M^1)$  and  $T^2 = (2^V; M^2)$  respectively, the algebraic complexity of  $C^1$ ,  $L(C^1) = L(M^1)$ , is greater than the algebraic complexity of  $C^2$ ,  $L(C^2) = L(M^2)$ , if and only if  $L(M^1) > L(M^2)$ .



The above definitions combined with propositions 1 and 2 allow the algebraic complexities of the economically relevant classes of lattices to be ranked. In addition, the example 3 choice functions demonstrate that the ranking is strict. The corollary applies the results of propositions 3.1 and 3.2 to classes of path independent choice functions

**Proposition 3.3** *For LLD lattices the algebraic complexity of the subclasses  $P$ ,  $D$ ,  $W$ , and  $S$  are ordered as follows:*

$$L(P) > L(D) > L(W) > L(S).$$

*Proof:* see Appendix 1

**Corollary 3.4** *For the classes of choice functions*

$PI = \{\text{class of path independent choice functions}\}$ ,

$RPI = \{\text{class of rational path independent choice functions}\}$ ,

$WARP = \{\text{class of weak order choice functions}\}$ ,

$SAP = \{\text{class of linear order choice functions}\}$ ,

*the choice function algebraic complexities are ordered as follows:*

$$L(PI) > L(RPI) > L(WARP) > L(SAP).$$

## 3.2 Algebraic Power

While the above results demonstrate that the mathematical systems are nested and, therefore, have a strict hierarchy in their relative powers to implement choice rules some intuition about how these power differences evince themselves may be useful. Two issues appear most meaningful; (1) the partial ordering of the join-irreducibles of the lattice action semigroups and the relation of this partial ordering to the existence of “representative elements”, and (2) representation and sufficient statistics for the choice process. These issues are most apparent when the choice functions considered are cleaved on the line between rational choice functions with algebras that must be some type of distributive lattice and non-rational choice functions whose associated algebras will be non-distributive lattices.

The reason for this partitioning of the classes of choice functions and their associated lattices is because for distributive lattices, the number join irreducibles and the number of meet irreducibles must be the same (and equal to the number of alternatives in the domain of the choice function). For the more general class of PI choice functions whose associated lattices need not be distributive, the number of join irreducibles and meet irreducibles need not be the same. This distinction between distributive LLDs and non-distributive LLDs has significant implications that are examined after the rational choice functions are considered.

### 3.2.1 Rational Choice Functions with Distributive Lattice Action Semigroups.

For the class of rational choice functions whose algebras are distributive lattices, it is useful to look at the join irreducibles of the lattice to see differences in the algebraic power to implement choices. Recall that the join irreducibles are simply the singleton set alternatives in  $V$ . Because all the lattices resulting from PI choice functions are LLDs, the join irreducibles are especially important. Specifically, every element in the lattice is uniquely representable as the irredundant join of the join irreducibles.

For the rational PI choice functions, one important reflection of the power present in the algebra is apparent in the partial ordering of the join irreducibles. Specifically, the covering relation in the partially ordered set (POS) of join irreducibles mirrors the strict preference structure of the relation rationalizing the choice function. For the SAP choice functions, whose algebras are chains, the chain orders the alternatives that can be chosen. In this class, all of the alternatives are comparable (no indifference). In the case of WARP choice functions, whose algebras are chains of Boolean algebras, again the POS of the join irreducibles reflects the ordering of the alternatives while the non-comparable elements will be those between which indifference exists in the relation that rationalizes the choice function.<sup>17</sup> In the case of RPI choice functions, the chains of Boolean algebras are augmented by fins that reflect the additional power required to accommodate the fact that only the complete choice set was what Johnson [15] [16] termed a “representative element” and the associated intransitivity of indifference for the quasi-transitive relation rationalizing the choice function. These fins are not required for the WARP choice functions because their rationalizing relation is fully transitive and each member of a choice set is a “representative element” for the entire choice set. The main points about the relationship between the partial ordering of the join irreducibles and the strict preference portion of the rationalizing relation is discussed in the following extension of Example 3. The “fin” feature is addressed more completely in a later example.

**Example 3b:** Figure 7 depicts the lattice representations of the action semigroups for the choice functions  $C_R$ ,  $C_W$  and  $C_S$  from Example 3. To highlight the significance of the join irreducibles, both the POS of join irreducibles and the lattice itself are presented. At the left is the POS of join irreducibles and on the right is the lattice itself with the join irreducibles filled in so that their role in the lattices can more easily be seen.

Reference back to the definition of the example 3.2 choice function  $C_R$  reveals that the rationalizing relation for this choice function has strict preference between just two pairs of alternatives; alternative 2 is strictly preferred to alternative 1 and alternative 4 is strictly preferred to alternative 3. These relations are matched precisely by the covering relation evident in the top left of figure 7. The associated lattice is distributive as is required for a choice function rationalized by a quasitransitive relation. The lattice diagram on the top right has two “fins”; one consisting of the subchain consisting of 123/23, 12/2 and 1 and the other consisting of the subchain 134/14, 34/4 and 3. This structure is examined more closely in example 3c.

For the example 3.3 choice function  $C_W$ , which is rationalized by a weak order and whose

---

<sup>17</sup>Given the ordering of the join irreducibles then, the Boolean algebras accomplish the task of constructing the equivalence classes while the chain (of Boolean algebras) orders these equivalence classes. Because choice functions rationalized by linear orders do not have equivalence classes, they do not need the Boolean algebras required to implement the WARP choice functions.

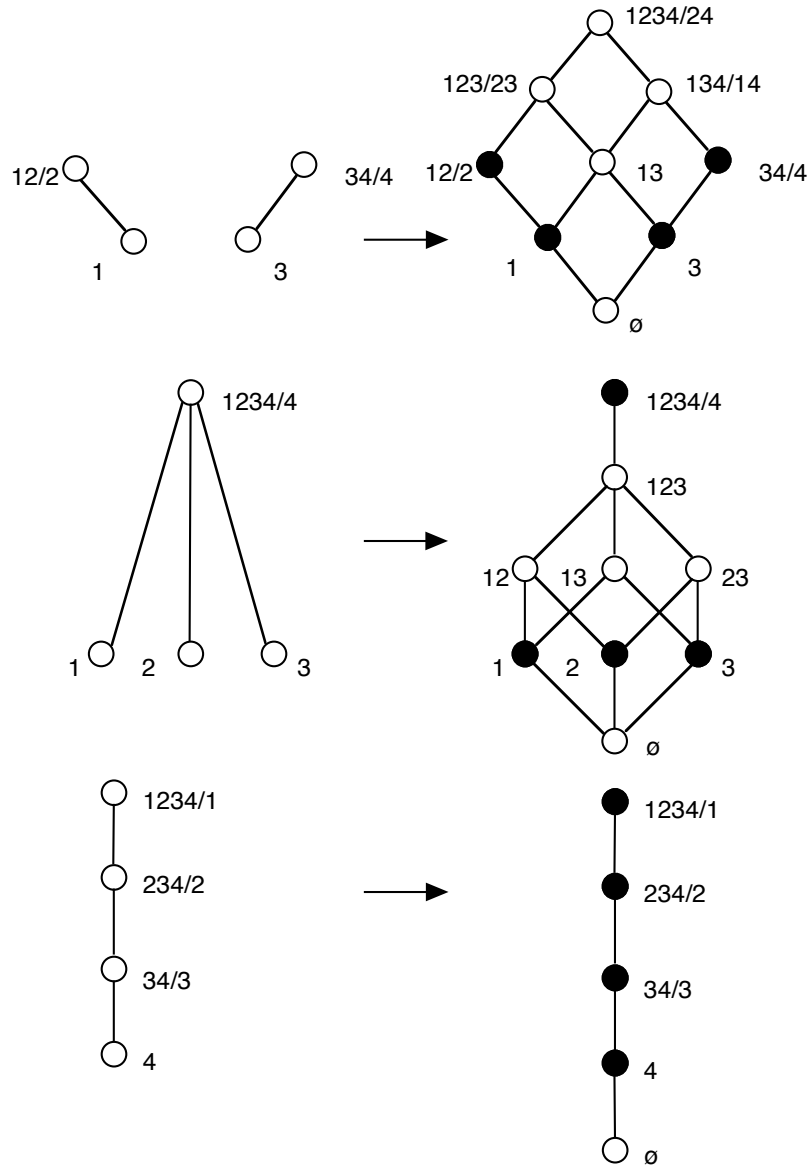


Figure 7: POS of join irreducibles for the rational choice functions of Example 3b

associated lattice must be a chain of Boolean algebras, the strict preference relation is; alternative 4 is strictly preferred to each of the alternatives 1, 2 and 3. As can be seen in the POS of the join irreducibles in the center left of figure 7, this relationship is matched by the covering relation of the POS of join irreducibles. Focusing on the lower Boolean algebra in the chain, it is possible to see the role this structure plays in the operations of the semiautomaton. For any collection of alternatives among the subset  $\{1, 2, 3\}$ , the Boolean algebra constructs the final choice set. For example  $1 \bullet 2 = 1 \vee 2 = 1 \cup 2 = 12$  because the join and union operations correspond in the Boolean algebra.

Finally, the example 3.4 choice function  $C_S$ , which is rationalized by a linear order, has a strict preference relation with alternative 1 preferred to alternatives 2, 3 and 4; alternative 2 preferred to alternatives 3 and 4; and alternative 3 preferred to alternative 4. This preference relationship also is matched by the covering relations of the POS in the lower left of figure 7. This system needs no width because it does not have to construct any equivalence classes.  $\diamond$

Our next example compares the example 3.2 choice function with an additional choice function to give a finer look at the role of the “fin” structure and the existence of “representative elements”.

**Example 3c:** At the top of Figure 8 is the lattice for the example 3.2 choice function  $C_R$ . On the left is the POS of join irreducibles and on the right is the associated lattice. Below this now familiar lattice is new POS and lattice associated with the following choice function.

$$\begin{aligned} C_{W_1}(\{1, 2, 3, 4\}) &= C_{W_1}(\{1, 2, 4\}) = C_{W_1}(\{2, 3, 4\}) = C_{W_1}(\{2, 4\}) = \{2, 4\}, C_{W_1}(\{1, 2, 3, \}) = \\ C_{W_1}(\{1, 2\}) &= C_{W_1}(\{2, 3\}) = C_{W_1}(\{2\}) = \{2\}, C_{W_1}(\{1, 3, 4\}) = C_{W_1}(\{1, 4\}) = C_{W_1}(\{3, 4\}) = \\ C_{W_1}(\{4\}) &= \{4\}, C_{W_1}(\{1, 3\}) = \{1, 3\}, C_{W_1}(\{1\}) = \{1\}, C_{W_1}(\{3\}) = \{3\}, C_{W_1}(\emptyset) = \emptyset. \end{aligned}$$

In the relation rationalizing the  $C_{W_1}$  choice function alternative 2 is strictly preferred to both alternatives 1 and 3 and alternative 4 also is strictly preferred to alternatives 1 and 3. The POS of the join irreducibles is presented on the bottom left of figure 8. That the strict preference relation and the covering relation match is immediately evident. The lattice for this choice function is composed of a chain of two Boolean algebras, both isomorphic to  $2^2$ . The top Boolean algebra consists of the elements labeled 1234/24, 123/2, 134/4 and 13 while the bottom Boolean algebra consists of the elements labeled 13, 1, 3 and  $\emptyset$ .

Now, it can be seen that it is the absence of the strict preference between 2 and 3 in the rationalizing relation accounts for the “fin” 123/23, 12/2, and 1 on the left of the distributive lattice at the top right of figure 8 while the absence of strict preference between 4 and 1 accounts for “fin” 134/14, 34/4 and 3 on the right side of the distributive lattice. In this  $C_R$  choice function, neither 2 or 4 is a representative element for the choice set. In contrast, for the  $C_{W_1}$  choice function the covering relation in the POS of join irreducibles has 123/2 covering both 1 and 3 and 134/4 covering both 1 and 3 and the “fin” structure is not present as either 2 or 4 is a representative element for the choice set.  $\diamond$

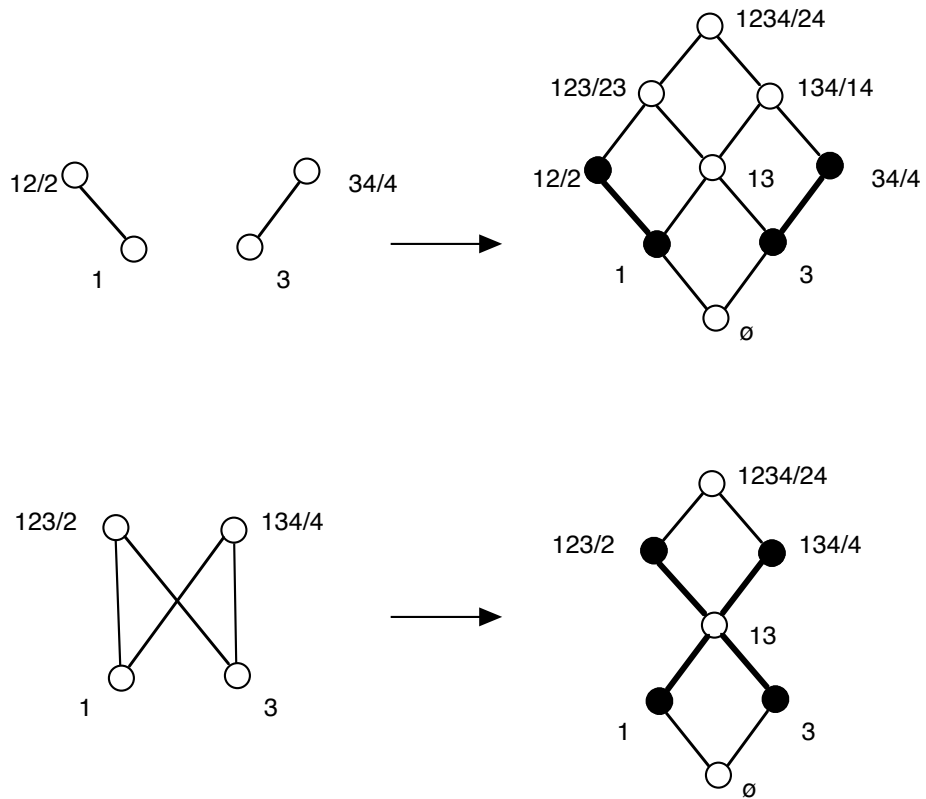


Figure 8: POS of join irreducibles for a distributive lattice and a chain of Boolean algebras

### 3.2.2 One lattice, two semigroups and power differences

Moving from the classes of rational to non-rational choice functions brings in a different type of distinction. Rationalizable PI choice functions have distributive lattices as their action semigroups while non-rationalizable choice functions have non-distributive lattices for their action semigroups. These differences are most evident when it is recognized that every lattice defines two semilattices, each of which is an idempotent, commutative semigroup and, therefore, defines a semiautomaton. In the case of choice lattices, one of the semigroups is associated with the join semilattice defined by Plott’s operation ( $\bullet$ ). This operation effectively determines choice as the feasible sets are “built up” from smaller sets. The other operation is derived from the meet semilattice and is induced by the requirements of path independence. This induced operation determines choice when the feasible sets become smaller. Here, the choice function operation associated with the meet semilattice operation is denoted ( $\star$ ) and is defined as follows.

**Definition 3.3** *For idempotents  $A$  and  $B$  in the range of a path independent choice function  $C$  let  $\text{arc}(A) = \hat{A}/A$  and let  $\text{arc}(B) = \hat{B}/B$  where  $\hat{A}$  and  $\hat{B}$  are the largest sets from which  $A$  and  $B$  respectively could have been chosen. Then the meet of  $A$  and  $B$  in the range of  $C$  is defined as  $A \wedge B = C(\hat{A} \cap \hat{B})$ .<sup>18</sup> To differentiate this operation from Plott’s operation ( $\bullet$ ), the induced operation for path independent choice functions is denoted by  $A \star B = A \wedge B = C(\hat{A} \cap \hat{B})$ .*

Now the lattice derived from a path independent choice function can be separated into its two semilattices; the join semilattice determined by Plott’s operation ( $\bullet$ ) and the meet semilattice induced by the requirements of path independence. Each of these semilattices defines an idempotent commutative semigroup. The join semilattice is  $(J; \bullet)$  while the meet semilattice is  $(M; \star)$ . Intuitively, the Plott operation tells what happens as the feasible set is enlarged (through set union) while the induced operation tells what happens when the feasible set is shrunk (through set intersection).

**Example 1c:** For the choice function in example 1, with  $I(V) = \{\{1\}, \{2\}, \{3\}, \emptyset\}$ , the meet semilattice is  $M = \langle I(V); \star \rangle$  and the associated “small” transformation semigroup is  $T' = (I(V); M)$ . The directed graph representation of the associated semiautomaton is presented in figure 9. Note that the set of “choice elements” or idempotents is common between the join and meet semigroups.  $\diamond$

Because the initial intuition for path independence was drawn from situations where the feasible set was rummaged through in some sequential search process, breaking the action semigroup lattice of a choice automaton into two semilattice semigroups may seem counterintuitive. However, one of the most basic variations in economic environments is the price change and, in most instances, a price change will both add some alternatives to the feasible set as well as remove some from consideration. Thus it is natural that a choice automaton should be able to make choice both in circumstances where the feasible set expands as well as those where it shrinks. In fact, this view is entirely consistent with the decomposition of the PI axiom into “expansion” and “contraction” consistency requirements.<sup>19</sup> When choice is modeled by semiautomata, this view specifies two different semiautomata and focuses attention on how these semiautomata interact.

<sup>18</sup>Johnson and Dean [18] [19].

<sup>19</sup>See Sen [37], Kelly [27] or Suzumura [38] for a treatment of the relationships between various consistency axioms and their decompositions into expansion and contraction consistency requirements.

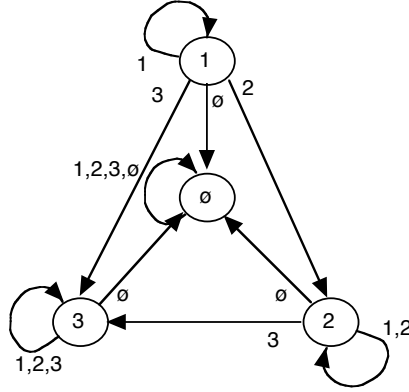


Figure 9: Directed graph of the meet choice semiautomaton for example 1c.

Most significantly, for the meet semilattice, the result of taking elements away is determined not necessarily by the elements that are removed but by the largest set from which the relevant idempotent elements in the action semigroup lattice could have been chosen. Thus, while the choice sets (which are the bottom of the intervals in the Boolean algebra that is the domain of the choice function) are a “sufficient statistic” for the choice process as the feasible sets are expanded, they are not for the case where the feasible set is shrunk. *Operationally, this means that as the feasible sets are shrunk, the final choice may depend on not just the alternatives in the choice set but on alternatives that were previously passed over.* This fact allows the differences in the power of the semiautomata to be demonstrated. Most dramatically, rationalizable choice function automata have action semigroups that are distributive lattices while non-rationalizable path independent choice function automata have action semigroups that are only guaranteed to be lower locally distributive. Thus differences between the classes of automata become clear. In particular, if the intent is that the result of a choice process should depend on the sequence in which the feasible sets are expanded and contracted, then the rationality requirement must be dropped. While contrary to the historic (economic) view that choice should not depend on the status quo or the sequence in which the feasible set is altered, some recent views on choice emphasize the importance of the status quo and the sequence in which choice is made. These results demonstrate that incorporating these aspects into the choice model may require more powerful mathematical systems to implement them.

**Example 4:** Consider two choice functions and their associated lattices as depicted in figures 10 and 11. Respectively these are the lattices for the RPI and PI choice functions described in example 3. Notice that the lattice depicted in figure 10 is distributive (and, thus, its choice function is rational) while the lattice in figure 11 is an LLD (with a non-rational choice function). The following calculation demonstrates the distributive property of the figure 10 lattice. The left side of this calculation is depicted in dotted lines on the figure while the right side of the calculation is depicted on solid lines. Consistent with the fact that the lattice is distributive, the result of the calculation is the same.

$$\{1\} \bullet (\{2\} \star \{4\}) = (\{1\} \bullet \{2\}) \star (\{1\} \bullet \{4\})$$

$$(\{1/1\} \vee (\{12/2\} \wedge \{34/4\})) = (\{1/1\} \vee \{12/2\}) \wedge (\{1/1\} \vee \{34/4\})$$

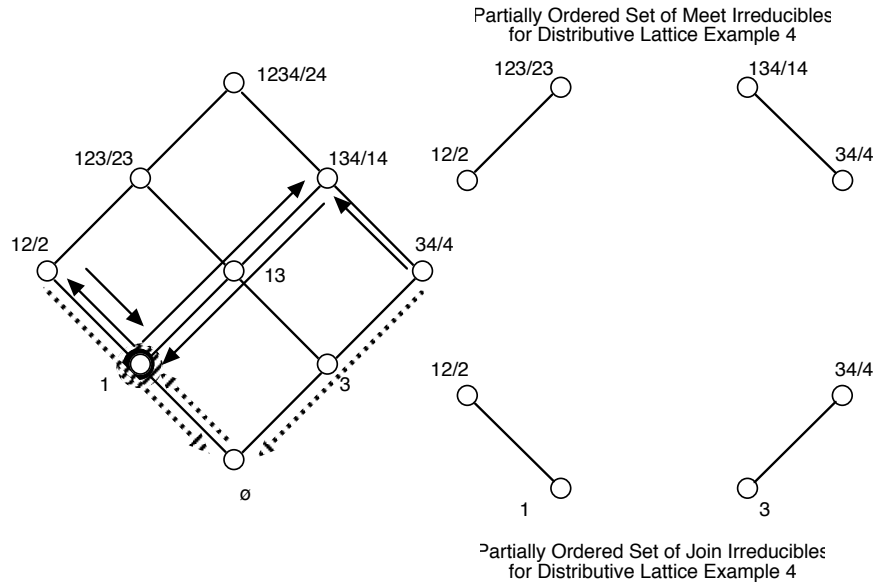


Figure 10: Distributive lattice for example 4

$$(\{1/1\} \vee \emptyset) = (\{12/2\} \wedge \{134/4\})$$

$$\{1/1\} = \{1/1\}$$

.

Figure 11 is different. In this case, the lattice is not distributive and the calculations below demonstrate that the result of expansions and contractions need not be independent of the sequence in which they occur. In fact, the calculations on the left side do not lead to the same result as those on the right side. This feature is depicted graphically in figure 11 with the left side being in dotted lines and the right side calculation being in solid lines.

$$\{1\} \star (\{2\} \bullet \{4\}) \neq (\{1\} \bullet \{2\}) \star (\{1\} \bullet \{4\})$$

$$(\{13/1\} \vee (\{2/2\} \wedge \{34/4\})) \neq ((\{13/1\} \vee \{2/2\}) \wedge (\{13/1\} \vee \{34/4\}))$$

$$(\{13/1\} \vee \emptyset) \neq (\{123/12\} \wedge \{1234/14\})$$

$$\{13/1\} \neq \{123/12\}$$

.



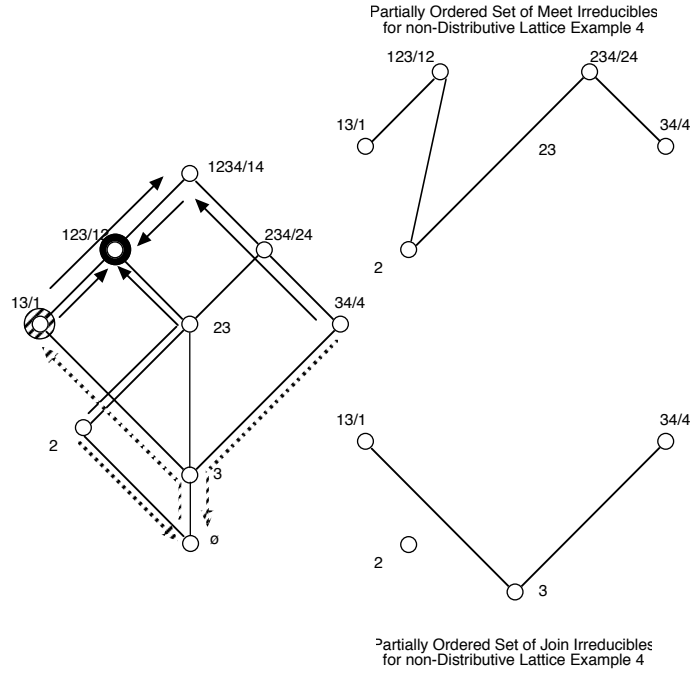


Figure 11: Non-distributive lattice for example 4

Thus, despite the fact that the choice function is path independent, the final choice can depend on the sequence of expansions and contractions of the sets from which the choice is made. This is true even though both of the initial sets and the final set from which choice is made are the same. The only difference is the sequence or order in which the additions and deletions are made. And, of course, the fact the largest sets from which particular choices could have been made varies.  $\diamond$

One intuition about how the additional power required to implement the PI choice function is used is to consider that more “effort” might be necessary to keep track of how the final choice must depend on the path taken.

Most obviously, the fact that distributive lattices have the same number of join irreducibles as meet irreducibles means that the two semilattices can be viewed as having roughly the same power. On the other hand, for the non-rational PI choice functions whose associated lattices are not distributive, the number of meet irreducibles can be greater than the number of join irreducibles. In distributive lattice example 4 above, the join irreducibles are  $12/2$ ,  $1$ ,  $3$  and  $34/4$ , while the meet irreducibles are  $12/2$ ,  $123/23$ ,  $134/14$  and  $34/4$ . Note that each collection of irreducibles has cardinality 4. For the non-distributive lattice from example 4, the collection of 4 join irreducibles  $13/1$ ,  $2$ ,  $3$  and  $34/4$  while there are 5 meet irreducibles  $13/1$ ,  $2$ ,  $123/12$ ,  $234/24$  and  $34/4$ . To emphasize the difference between the meet and join irreducibles in moving from the distributive lattice to the non-distributive lattice, the POSs of irreducibles are presented to the right of the lattices.

The important feature in the above discussion might be that while the dual of a distributive lattice is always a distributive LLD, the dual of a non-distributive LLD is never an LLD (Johnson and Dean [23]). Thus, in the case of distributive lattices the join semilattice and meet semilattice can interact well while in the case of the non-distributive lattices the two semilattices do not interact well.

## 4 Computational Complexity

In contrast to the algebraic complexity which reflects the power required to implement a choice rule, computational complexity concerns the resource requirement to construct, determine or identify a solution to a specific problem. For Turing machines, the goal is to identify fundamental aspects of the problem that are independent of the specific machine on which an algorithm is performed. Classically, this goal for a computational complexity measure is formalized by the Blum axioms (Blum [6]). In those axioms, the goal is to define an aspect of the problem that is independent of whether the problem is being solved on a “big” machine or a “small” machine. Here the computational complexity measure is based on the requirements of computing a particular choice function independent of the specific algorithm.

For PI choice functions the “work” that must be done is to identify the collection of sets from which the same choice will be made. As seen earlier, for PI choice functions these collections of sets must be intervals in the Boolean algebra of the domain. Determining these intervals is the focus of the computational complexity measure used here. Specifically, the computational complexity of a particular choice implementing semiautomaton is the number of prime intervals defining the equivalence class intervals in the Boolean algebra that are mapped into LLD action semigroup associated with the choice function being implemented.

Although this computational complexity measure is independent of the specific machine and, indeed, independent of the algorithm, it is informative to examine the operations of one particular algorithm in order to see what items must be identified in the process of constructing choice lattices. The algorithm used for this exposition is the Johnson and Dean “contraction” algorithm (Johnson and Dean [18] [19] [21]).<sup>20</sup> The mechanics of the Johnson and Dean contraction algorithm are offered below.

**Proposition 4.1 (Contraction of an Interval)** *Let  $C$  be a PI choice function on a finite set  $V$ . Let  $B$  be a meet irreducible element in the choice lattice that is not equal to  $C(V)$  or  $\emptyset$ . Let  $A$  be the unique element covering  $B$ . Suppose that  $A = B \cup \{x\}$ . Let the function  $C^*$  be defined on  $2^V$  as:*

$$\begin{aligned} C^*(S) &= C(S), \text{ if } C(S) \neq A \\ C^*(S) &= B, \text{ if } C(S) = A. \end{aligned}$$

*The function  $C^*$  is a path independent choice function on  $2^V$ .*

---

<sup>20</sup>In part, this algorithm is used for the exposition because it was the first algorithm to be identified that could compute all choice lattices on a finite set of alternatives. The same number of prime intervals would have to be identified if some other algorithm, say the Johnson and Dean “designer” algorithm (Johnson and Dean [22]).

As the following theorem assures, every PI choice function on a set  $V$  can be constructed by a series of these contractions.

**Theorem 4.1** <sup>21</sup> *Every PI choice function on a finite set  $V$  can be constructed by a sequence of contractions beginning with the identity choice function on  $V$ .*

By repeated applications of the contraction operation, every PI choice function on  $V$  can be constructed. To exposit the algorithm a little notation is useful. First, define a relation on the action semigroups as follows: If  $M$  is the result of a single contraction of the LLD lattice action semigroup  $L$ , write  $L \gg M$ . Let the transitive closure of  $(\gg)$  be denoted by  $(\geq)$  and note that this relation partially orders the set of all LLD action semigroups for PI choice functions defined on a set  $V$ . For  $|V| = n$ , the top of the POS is the Boolean algebra  $2^V$  and at the bottom are the linear orders on  $n$  elements. In between are all the possible LLD action semigroups for PI choice functions on  $V$ . The POS of LLD lattice action semigroups on  $V$  is graded by the number of elements in the lattices  $L$ .<sup>22</sup>

For the domain  $V$ ,  $|V| = n$ , the starting point for the algorithm is the Boolean algebra  $2^V$ . This lattice has  $2^n$  elements. In  $2^V$ , each of the  $n$  elements immediately below the top element is meet irreducible and appropriate for contraction. Further, no other elements are appropriate for contraction. Each the  $n$  lattices resulting from the contraction of  $2^V$  will result in a lattice with  $2^n - 1$  elements and they will all be isomorphic to each other. Now suppose that all LLD action semigroup lattices with  $q$  elements  $L_1, L_2, \dots, L_P$  have been constructed. The next step is to construct all of the lattices with  $q - 1$  elements. This is accomplished by examining each of the  $P$  lattices  $L_h$  with  $k$  elements and identifying each of the elements appropriate for contraction and constructing all the LLD lattices  $M$  such that  $L_h \gg M$  by effecting every possible contraction. Because some lattices can be constructed by more than one sequence of contractions, this process will produce multiple copies of some lattices but these can be identified and eliminated.

Before formalizing the measure of computational complexity, it is useful to consider an example on three elements that, while not large, does reveal the relevant mathematical systems. Conveniently, up to isomorphism, there are only five different lattices that arise and they cover all four of the classes of lattices that can arise from a PI choice function.

**Example 5:** Construction of the five discriminating choice lattices on three elements is diagramed in figure 10. Application of the contraction algorithm is direct. In most cases, so is identifying the computational complexity. The exceptions are the sequences of contractions resulting in the two chains of Boolean algebras. Special note will be made of features of those contractions and how that relates to the computational complexity of the choice function. Consider PI choice functions defined on  $V = \{1, 2, 3\}$ . The algorithm begins with the Boolean algebra on three elements presented at the top left of figure 10. The first discriminating lattice, labeled L7 at the top right of the diagram, is constructed by contracting one of the three intervals 123/12, 123/13, 123/23. Each of these contractions will result in a lattice isomorphic to the lattice depicted on the top right of the diagram. In that diagram, the interval 123/13 has been contracted. Note that *one* prime interval has been contracted. The second contraction is depicted on the next level where two

---

<sup>21</sup>Johnson and Dean [18] [19]

<sup>22</sup>A more formal treatment of the algorithm for constructing LLD lattices is available in Johnson and Dean [21].

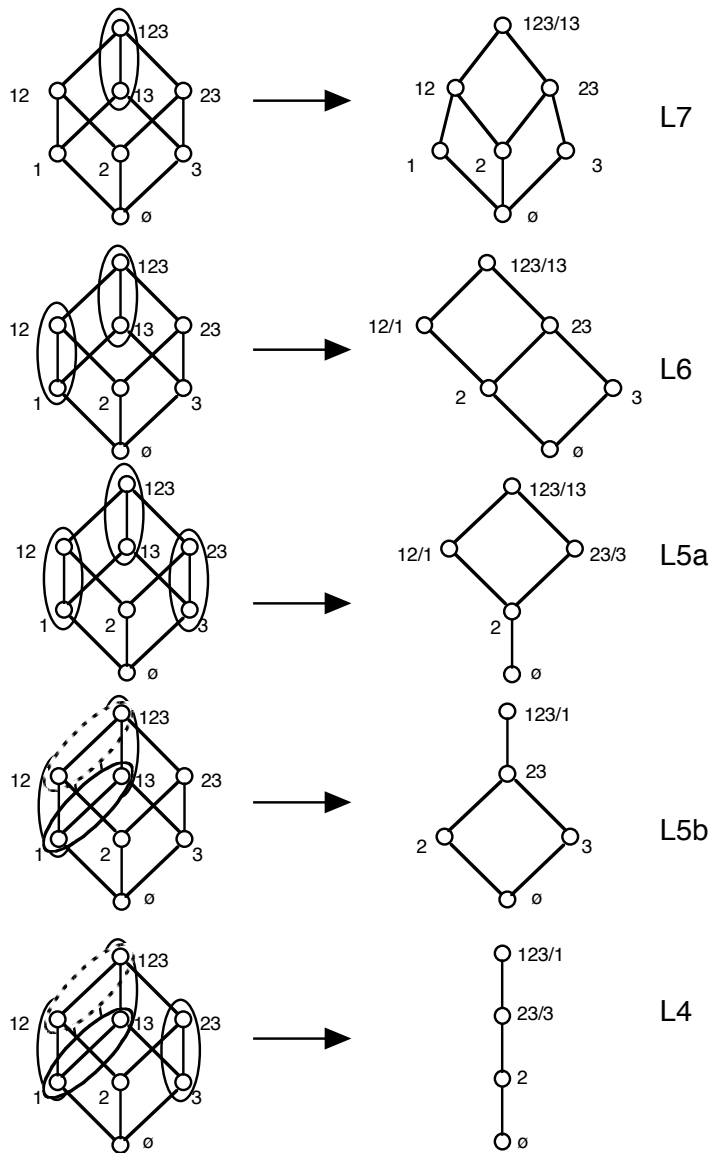


Figure 12: Construction of Choice Functions on Three alternatives.

intervals have been contracted. In this case there were two possible intervals that could have been contracted 12/1 or 23/3. Either contraction will result in a lattice that is isomorphic to the six element distributive lattice second from the top right labeled L6. In this case the interval 12/1 has been contracted. At this stage we have *two* prime intervals that have been contracted. In L6, there are again two intervals that can be contracted. One of them, 23/3, is easy to see because it is just like the earlier contractions. Contracting this interval leads to the lattice on the middle right, labeled L5a, that has a two element Boolean algebra on top of a singleton. In this lattice *three* prime intervals have been contracted. The other interval in L6 that can be contracted is an interval that involves two previously contracted intervals; this interval consists of the lattice point labeled 12/1 and the lattice point labeled 123/13. The result of this contraction is lattice with a two-element Boolean algebra on the bottom, labeled L5b. Looking to the left of this lattice, it can be seen that, although only three contraction operations have been made, a total of *four* prime intervals have been contracted. This is because the interval being contracted consisted of previously contracted intervals. Even though only three contraction operations have been made, the fourth prime interval has been contracted because of the interaction of the two contractions made previously. The induced contraction is depicted by the dashed loop. The final contraction is performed on L5b and results in the chain labeled L4 and this lattice has *five* prime intervals that have been contracted. If instead of working with L5b we had stayed with L5a where the Boolean algebra is on top, then there are two intervals that can be contracted and both of them involve previously contracted intervals. Contracting either of them results in a lattice isomorphic to the chain in L4 and that chain must have *five* prime intervals that have been contracted.  $\diamond$

A few comments on the example. First, the first discriminating lattice to occur is L7, a non-distributive LLD lattice. Independent of the size of the Boolean algebra used as the initial point of the algorithm, the first contraction will always result in a non-distributive LLD. This means that a representative of the most powerful class of lattices can be constructed with the fewest prime interval contractions — one. The next class of lattices obtained (by contracting two prime intervals in this example) is the distributive lattice, L6. Note that this distributive lattice is not a chain of Boolean algebras (a specific subclass of distributive lattices). Again, this result is generic. Lattices L5a and L5b are both chains of Boolean algebras where L5a requires three prime intervals to be contracted while L5b requires four prime intervals to be contracted. Lattice L4 is a chain in which five prime intervals have been contracted. Qualitatively, this relationship between the class of lattice and the minimum number of prime intervals that have to be contracted in order to construct a member of the class holds, independent of the size of  $V$ . Finally, observe that the computational complexity is not simply a proxy for the number of elements that have been deleted from the Boolean algebra of the domain. This can be seen by noting that both L5a and L5b have three fewer elements than their domains but L5a has only three prime intervals that have been contracted while L5b has four prime intervals contracted.<sup>23</sup>

Structurally, the reason the lattices arise in the sequence identified in the example above is because of the POS of join irreducibles, The lattice obtained by identifying a single prime interval for contraction the join irreducibles are 1, 2 and 3. For this lattice, there are no restrictions among join irreducibles. They are each not comparable just as they are in the Boolean algebra from which

---

<sup>23</sup>This example on three alternatives might lead some to think that it always will be the case that *every* non-distributive lattice will require fewer prime intervals to be contracted than any distributive lattice or that *every* distributive lattice will require fewer prime interval contractions than any chain of Boolean algebras. This is not the case. To see a richer example in which, for example, some non-distributive LLD lattices require more prime interval contractions than some distributive LLD lattices see Johnson and Dean [20].

the lattice was constructed. For L6, the join irreducibles are 12/1, 2 and 3 and these irreducibles there is a restriction; 12/1 covers 2. This feature is generic. For a chain of Boolean algebras, there must be at least two restrictions among the join irreducibles. In the case of the example on three elements, both L5a and L5b have two restrictions in their join irreducibles; for of L5a, 12/1 and 23/3 cover 2 while for L5b, 123/1 covers 2 and 3. Again this feature is generic.

There is a sense in which this example on three alternatives oversimplifies the situation however. For this specialized example, it turns out that the number of elements in the algebra is coincident with the class of lattice. Specifically, the only chains of Boolean algebras have exactly five elements and there are no distributive lattices on five or fewer elements. This could lead some to the misunderstanding that the number of states and the mathematical power, or algebraic complexity, of the semiautomaton are synonymous. This is not the case and this situation does not occur on larger domains. In particular, three of the four action semigroups presented in example 3 have nine elements and each of these three nine-element semigroups represents a different class of lattice; one LLD, one distributive lattice and one chain of Boolean algebras. Two of these three lattices were used in example 4 to demonstrate explicitly power differences in the action semigroups. This point is made more starkly when the full range of possible choice functions on four alternatives is considered. There, while the order in which the first member of each class arises is the same as on three alternatives, it also is the case that there are non-distributive LLD lattices that are smaller than some distributive lattices or, even than some of the chains of Boolean algebras.<sup>24</sup>

The computational complexity measure now can be formalized.

**Definition 4.1** *Let  $C$  be a path independent choice function defined on  $V$  and let  $J$  be the associated idempotent action semigroup. The computational complexity of  $J$ ,  $k(J)$ , is the number of prime intervals in the Boolean algebra  $2^V$  that must be identified to obtain  $J$*

Here we see that the computational complexity of a particular choice function is measured by the number of prime intervals that must be contracted in the Boolean algebra in order to construct the action semigroup for the choice implementing semiautomaton. This measure simply reflects the effort required to identify the collections of sets from which the same choice will be made.

In economic applications a major focus is on the computational complexity of the classes of choice functions and their associated semiautomata rather than the complexity of a specific semiautomaton. Indeed, a class computational complexity measure can be identified. This measure is based on the minimum number of prime intervals that must be identified to construct a representative of the class.

**Definition 4.2** *For path independent choice functions defined on  $V$  with cardinality  $n$  satisfying a consistency axiom  $A$  and action semigroups  $J$  with  $n$  join irreducibles belonging to the class of LLD lattices  $B$  Let the computational complexity  $K$  of a class  $B$  of LLD lattices be defined as follows;*

$$K(B) = (t | t = \min_{J \in B}(k(J))).$$

---

<sup>24</sup>Again reference Johnson and Dean [20]. Included, at the end of the Atlas, also is a brief description of the technique for recovering the choice function from any of the lattices.

This sequence of definitions identifies: (i) a computational complexity measure for a semi-automaton implementing a particular choice function based on the number of prime intervals that must be identified in order to construct the action semigroup, (ii) for a class of LLD action semigroup lattices, the computational complexity of the class is the minimum number of prime intervals that must be identified in order to construct a member of the class.

**Proposition 4.2** *Let  $V$  be a collection of  $n \geq 3$  join irreducibles and let  $2^V$  be the Boolean algebra on  $V$ . For discriminating choice functions defined on  $V$ , the computational complexities of the following classes of action semigroups  $P$ ,  $D$ ,  $W$ , and  $S$  are ordered as follows:*

$$K(P) < K(D) < K(W) < K(S).$$

*Proof:* See Appendix 1

## 5 Conclusions

While differing from previous approaches to addressing the structure of economically rational choice automata (for example, see Futia [12] or Gottinger [14]), the results presented here characterize the structure of choice-implementing semiautomata when constrained to satisfy standard economic consistency axioms. The approach is to combine previous algebraic results on choice functions of Plott [35], Johnson [15] [16] and Johnson and Dean [18] [19] [21] with work on classic algebraic automata theory results from Eilenberg [10] [11] and Holcombe [24]. Notably, the characterization results are tight in that each class of PI choice function is associated with a specific class of action semigroup.

For these choice-implementing semiautomata two different complexities are identified. The first, deriving directly from the characterization results is algebraic complexity, which reflects the mathematical power required of the semiautomaton in order to correctly implement the choice rule being modeled. When ranked by algebraic complexity, the broadest class of choice functions is identified as requiring the highest power in order to be correctly implemented. As the class of choice functions becomes increasingly restricted, the power required to correctly implement the choice rule is reduced. When viewed as choice implementing semiautomata, one intuition is that as the class of choice functions becomes more restrictive, the environments in which the semiautomata operate becomes “simpler” .

In contrast, the computational complexity that is determined by the effort required to make the action semigroups of the choice-implementing semiautomaton, is demonstrated to be lowest for the broadest class of choice functions and increasingly higher for the more restrictive classes. The class of choice function with the highest computational complexity is the class of choice functions rationalized by linear orders.

Perhaps most intriguingly, as identified in propositions 3.3 and 4.2, the two complexities are dual with algebraic complexity being highest when the computational complexity is lowest.

## 6 Appendix I

*Proof of Proposition 3.1:* Part 3.1.1 follow directly from the fact that  $J$  is a subsemigroup of Plott's semigroup  $\langle 2^V; \bullet \rangle$  and thus satisfies the associativity condition. The faithfulness condition is satisfied because  $J = \langle I(V); \bullet \rangle$  is a monoid (Johnson [16]) in which the empty set  $\emptyset$  is the identity element. Thus because  $\emptyset \in 2^V$ ,  $s \neq s'$  implies  $qs \neq qs'$  whenever  $q = \emptyset$ .

Part 3.1.2 follows from the fact that  $T = (2^V; J)$  is a transformation semigroup in which  $2^V$  is the underlying set and, as noted above,  $J = \langle I(V) : \bullet \rangle$  is the action semigroup. That  $T = (2^V; J)$  is a complete transformation semigroup follows from the completeness of the choice function.  $\square$

*Proof of Proposition 3.2:* Notice that the action semigroups are the range of the mapping C. Thus the characterization results of Johnson and Dean [19] provide the necessary characterizations. Part 3.2.1 follows by application of theorems 3 and 4. Part 3.2.2 follows by application of theorems 9 and 10. Part 3.2.3 follows by application of theorem 11 parts (iv) and (v). Part 3.2.4 follows by application of theorem 12 parts (i) and (ii).  $\square$

*Proof of Proposition 3.3:* This ordering follows directly from the nested nature of the classes of lattices. Simple examples can demonstrate that the containments are strict. In fact, from example 3, observe the the PI choice function diagramed as figure 5d is an LLD lattice that is not distributive, the RPI choice function diagramed as figure 5c is a distributive lattice that is not a chain of Boolean algebras and the WARP choice function diagramed as figure 5b is not a chain.  $\square$

*Proof of Proposition 4.2:* The easiest to see are  $K(P)$  and  $K(S)$ . Clearly, for any  $V$ ,  $|V| \geq 3$ ,  $K(P) = 1$  because a non-distributive LLD always can be constructed with a single contraction. This can be seen by observing that the result of a single contraction of  $2^V$  is an LLD lattice with  $2(n-1)$  join irreducibles. Because the contraction operation produces only LLD lattices and the number of join irreducibles is not equal to  $n$ , the LLD lattice is not distributive. Finally, because only one contraction has been made, there can only be one prime interval that has been identified.

To see that  $K(S)$  is the largest computational complexity observe that the chain on  $n$  elements is the smallest LLD on  $n$  join irreducibles and thus has the largest number of contracted prime intervals Thus clearly  $K(P) < K(S)$ .

Further, it should be clear that  $K(D)$  and  $K(W)$  are between  $K(P)$  and  $K(S)$ .

For,  $K(D)$  can be found by reference to Corollary 2 of Johnson and Dean [22] that assures that the distributive LLD requiring the fewest prime intervals to be contracted will be the one with only one comparable pair in its partially ordered set of join irreducibles. This will be the LLD that is the direct product of a chain of order 3 and a Boolean algebra isomorphic to  $2^{n-2}$ .

Finally for  $K(W)$  observe that any chain of Boolean algebras will require at least two comparable pairs in the partially ordered set of join irreducibles and, thus, must have more prime intervals contracted than for a distributive LLD. Thus,  $K(D) < K(W)$ .

Thus the computational complexities are ordered as  $K(P) < K(D) < K(W) < K(S)$ .  $\square$



## References

- [1] Abreu, D. and A. Rubinstein, (1988): “The structure of Nash equilibrium in repeated games with finite automata”, *Econometrica*, 1259-1281.
- [2] Arrow, K. J. (1959): “Rational choice functions and orderings”, *Econometrica*, **26**, 121-127.
- [3] Banks, J. S., and R. K. Sundaram (1990): “Repeated games, finite automata and complexity”, *Games and Economic Behavior*, **2**, 97-117.
- [4] Ben-Porath, E. (1990): “The complexity of computing a best response automaton in repeated games with mixed strategies”, *Games and Economic Behavior*, **2**, 1-12.
- [5] Birkhoff, G. (1967): *Lattice Theory*, (3rd ed.), Providence, American Mathematical Society.
- [6] Blum, M. (1967): “A machine-independent theory of the complexity of recursive functions”, *Journal of the Association for Computing Machinery*, **14**, 322-336.
- [7] Clifford, A. H. and G. B. Preston (1967): *The Algebraic Theory of Semigroups Vol. 1*, (2nd ed.), Providence American Mathematical Society.
- [8] Clifford, A. H. and G. B. Preston (1967): *The Algebraic Theory of Semigroups Vol. 2*, Providence American Mathematical Society.
- [9] Dilworth, R. P. (1950): “A decomposition theorem for partially ordered sets”, *Annals of Mathematics*, **51**, 161-166.
- [10] Eilenberg, S. (1974): *Automata, Languages and Machines, Vol. A*, New York, Academic Press.
- [11] Eilenberg, S. (1976): *Automata, Languages and Machines, Vol. B*, New York, Academic Press.
- [12] Futia, C. (1977): “The complexity of economic decision rules”, *Journal of Mathematical Economics*, **4**, 289-299.
- [13] Gilboa, I. (1988): “The complexity of computing best response automata in repeated games”, *Journal of Economic Theory*, **45**, 342-352
- [14] Gottinger, H. W. (1978): “Complexity in social decision rules”, in *Decision Theory and Social Ethics*, (H. W. Gottinger and W. Leinfellner eds.), Dordrecht, D. Reidel, 251-269.
- [15] Johnson, M. R. (1990): “Information associativity and choice requirements”, *Journal of Economic Theory*, **52**, 440-452.
- [16] Johnson, M. R. (1995a): “Ideal structures of path independent choice functions”, *Journal of Economic Theory*, **52**, 440-452.
- [17] Johnson M. R. (1995b) “Algebraic complexity of strategy implementing automata”, presented at Econometric Society World Congress, Tokyo, Japan 1995 and NSF/NBER Decentralization Conference, Tuscaloosa, Alabama, 1996.
- [18] Johnson, M. R. and R. A. Dean (1996) “An algebraic characterization of path independent choice functions”, presented at the Third International Meeting of the Society for Social Choice and Welfare, Maastricht, The Netherlands, 1996. (available at <http://homepage.mac.com/markjohnson>)

- [19] Johnson, M. R. and R. A. Dean (2001a): “Locally complete path independent choice functions and their lattices”, *Mathematical Social Sciences*, **42**, 53-87.
- [20] Johnson, M. R. and R. A. Dean (2001b): “The Construction of all Lower Locally Distributive Lattices on 4 Elements”, (available as “Atlas” at <http://homepage.mac.com/markrjohnson>)
- [21] Johnson, M. R. and R. A. Dean (2002): “Construction of finite lower locally distributive lattices”, presented at American Mathematics Society Meetings, San Diego, California. (available as “algorithm” at <http://homepage.mac.com/markrjohnson>)
- [22] Johnson, M. R. and R. A. Dean (2003): “Designer Path Independent Choice Functions” presented at Lattices, Universal Algebra and Applications, Lisbon, Portugal, 2003. (available as “DesignPCIF” at <http://homepage.mac.com/markrjohnson>)
- [23] Johnson, M. R. and R. A. Dean (2004): “Duality and distributivity in LLD lattices” mimeo. (available as “DualityandLLD” at <http://homepage.mac.com/markrjohnson>)
- [24] Holcombe, W. M. L. (1982): *Algebraic Automata Theory*, Cambridge, Cambridge University Press.
- [25] Kalai, E. (1990): “Bounded rationality and strategic complexity in repeated games”, in *Game Theory and Applications*, (T. Ichiishi, A. Neyman, Y. Tauman eds.) San Diego, Academic Press, 131-157.
- [26] Kalai, E. and W. Stanford (1988): “Finite rationality and interpersonal complexity in repeated games”, *Econometrica*, **56**, 397-410.
- [27] Kelly, J. S. (1978): *Arrow Impossibility Theorems*, New York, NY, Academic Press.
- [28] Knuth, D. E. (1973): *The Art of Computer Programming: Vol. 3/Sorting and Searching*, Reading, MA, Addison-Wesley Publishing company.
- [29] Koshevoy, G. A. (1999): “Choice functions and abstract convex geometries”, *Mathematical social Sciences*, **38**, 35-44.
- [30] Lipman, B. L. and S. Srivastava (1990): “Informational requirements and strategic complexity in repeated games”, *Games and Economic Behavior*, **2**, 273-290.
- [31] Myhill, J. (1957): “Finite automata and the representation of events”, WADD TR-57-624, 112-137. Wright Patterson AFB, Ohio.
- [32] Nerode, A. (1958): “Linear automaton Transformation, Proc. AMS, **9**, 541-544.
- [33] Neyman, A. (1985): “Bounded complexity justifies cooperation in the finitely repeated prisoner’s dilemma”, *Economics Letters*, **19**, 227-229.
- [34] Papadimitriou, C. H. (1992): “On players with a bounded number of states”. *Games and Economic Behavior*, **4**, 122-131.
- [35] Plott, C. R. (1973): “Path independence, rationality and social choice”, *Econometrica*, **41**, 1075-1091.
- [36] Rubinstein, A. (1986): “Finite automata play the repeated prisoner’s dilemma”, *Journal of Economic Theory*, **39**, 83-96.

- [37] Sen, A. K. (1970): *Collective Choice and Social Welfare*, San Francisco, Holden-Day.
- [38] Suzumura, K. (1983): *Rational choice, collective decisions and social welfare*, Cambridge, Cambridge University Press.