American Economic Review 101 (April 2011): 724–748
http://www.aeaweb.org/articles.php?doi=10.1257/aer.101.2.724

Procedural Analysis of Choice Rules with Applications to Bounded Rationality

By Yuval Salant*

I study how limited abilities to process information affect choice behavior. I model the decision-making process by an automaton, and measure the complexity of a specific choice rule by the minimal number of states an automaton implementing the rule uses to process information. I establish that any choice rule that is less complicated than utility maximization displays framing effects. I then prove that choice rules that result from an optimal trade-off between maximizing utility and minimizing complexity are history-dependent satisficing procedures that display primacy and recency effects. (*JEL* D01, D03, D11, D83)

In 1981, an internal study by American Airlines found that travel agents booked the first flight that appeared on their computer screen 53 percent of the time, and a flight that appeared somewhere on the first screen almost 92 percent of the time. This provided an incentive for American to manipulate the order in which flights were listed on American's "Sabre" computer reservation system, which dominated the market for electronic flight booking. Allegations by smaller airlines that American—and United Airlines which owned the second largest computer reservation system "Apollo"—engaged in such manipulations led the US federal government to regulate screen display in computer reservation systems.[1]

This example, in which the order of the alternatives affects decision making, is by no means unique. In elections, for example, being listed first on the ballot increases the likelihood of winning office, mainly at the expense of candidates listed in the middle of the ballot.[2] In competitions, judges tend to award higher

* Kellogg School of Management, Northwestern University, 2001 Sheridan Rd, Evanston, IL (e-mail: y-salant@northwestern.edu). I am indebted to Bob Wilson for his devoted guidance, constant support, and most valuable suggestions. I am grateful to Ariel Rubinstein and Jeremy Bulow for the encouragement, productive discussions, and important comments. I thank Gil Kalai, Ron Siegel, and Andy Skrzypacz for most insightful feedback in various stages of this project. I also thank Drew Fudenberg, Jon Levin, Michael Ostrovsky, Roy Radner, Ilya Segal, and seminar participants at UC Berkeley, Brown University, Caltech, Harvard University, Hebrew University, University of Iowa, LSE, MIT, Northwestern University, NYU, Stanford University, Tel-Aviv University, UCL, Washington University in St. Louis, and Yale University for helpful comments. This research is supported in part by the Leonard W. and Shirley R. Ely Fellowship of the Stanford Institute for Economic Policy Research.

[1] See US Department of Justice (1983), US General Accounting Office (1986), and Duncan G. Copeland, Richard O. Mason, and James L. McKenney (1995).

[2] Joanne M. Miller and Jon A. Krosnick (1998) and Krosnick, Miller, and Michael P. Tichy (2004) find statistically significant and sometimes substantively large effects of being listed first on the vote shares of major party candidates in US federal elections. Jonathan G. S. Koppell and Jennifer A. Steen (2004) and Daniel E. Ho and Kosuke Imai (2008) find similar effects for major party candidates in primary elections in the United States. Amy King and Andrew Leigh (2008) replicate these findings in partisan Australian federal elections. Marc Meredith and Salant (2009) find that in local elections in California, the proportion of election winners from the first ballot position is larger than expected and the proportion of winners from the median ballot position is smaller than expected, absent order effects.

scores to contestants who appear later in the competition.[3] And in designing a menu, restaurant managers are advised to position their highest profit-margin items first, second,and last on the menu.[4] In these and other contexts, investigating the sources and nature of order effects may provide a deeper understanding of their economic, strategic, and policy consequences.

One of the sources of order effects is likely to be cognitive and procedural limitations.[5] Choosing the best available flight, for example, seems cognitively demanding because it may involve comparing each flight to every other flight. Travel agents may find it simpler to choose the first satisfactory flight that appears on their screen. Similarly, recalling the performance of the last contestant in a competition seems simpler than recalling the performances of those who appeared before him, which may lead to a tendency to choose the last contestant.

The goal of this paper is to investigate how certain cognitive and procedural limitations relate to order effects and choice behavior. The main takeaway from the analysis is that any choice rule that is procedurally simpler than utility maximization displays order effects, and that primacy and recency effects as well as other common behavioral effects emerge when an expected utility maximizer optimally economizes on procedural costs.

In the procedural model I investigate, the decision maker can be in one of several *states*. A state may reflect the state of mind of the decision maker. For example, a satisficer, who chooses the first satisfactory alternative he encounters, is "unsatisficed" until he encounters a satisfactory alternative and becomes "satisficed." A state may also capture information the decision maker considers relevant for making a choice. For example, when maximizing a utility function, a state may record the best alternative encountered so far.

States are used in processing information and making choices. The decision maker encounters the alternatives in a form of a list (as in the examples above), and processes them one after the other as follows. He starts in some predetermined state. When considering the next alternative in the list, he decides (as a function of his current state and the current alternative) whether to stop and make a choice or continue. If the decision maker continues, he updates his state as necessary and then considers the next alternative. When reaching the last alternative, the decision maker must make a choice, if he has not yet done so. For example, a satisficer may be thought of as staying in the "unsatisficed" state until he encounters a satisfactory alternative, in which case he stops and chooses it.

States are costly in the sense that the more states a decision maker uses in making choices, the more cognitive resources he expends and the finer the information he utilizes. The *state complexity* of a given choice behavior is the minimal number of states needed to implement that behavior.

[3] Order effects have been found in contests such as the World Figure Skating Competition (Wändi Bruine de Bruin 2005), the International Synchronized Swimming Competition (Vietta E. Wilson 1977), the Eurovision Song Contest (Bruine de Bruin 2005), and the Queen Elisabeth Contest for violin and piano (Herbert Glejser and Bruno Heyndels 2001).

[4] See Jack E. Miller (1980).

[5] In a seminal work, Herbert A. Simon (1955) argued that cognitive and procedural considerations may push decision makers toward various forms of "boundedly" rational behavior, and suggested satisificing as an alternative to utility maximization. The bounded rationality literature develops this idea in different contexts.

State complexity was used to measure the complexity of strategies in repeated games and the complexity of decision rules in the context of individual inference making.[6] It reflects the cognitive costs associated with refining the information used in making decisions. It is also related to the notion of categorization, as I show in Section III. Intuitively, the decision maker may be thought of as classifying past information into one of several categories, each of which corresponds to a state, and then processing future information based on this classification.

Clearly, the complexity of making choices involves additional procedural aspects beyond those captured by state complexity. For example, it may be costly for a decision maker to understand which alternatives are feasible, or to search for the next alternative. In analyzing the effect of state complexity on choice behavior, I make only a first step in understanding the effects of cognitive and procedural limitations on individual choice behavior.

I focus on comparing the state complexity of utility maximization, or *rational choice*, to that of other behaviors. A rational decision maker has in mind a strict preference relation $\succ$ over the alternatives that he maximizes when making choices.[7] Rational choice is frequently used in economics to model the behavior of economic agents and is robust to order effects.

The way a rational decision maker processes information depends on the identity of the best alternative he encountered so far. For every two alternatives x and y, the set of alternatives that are $\succ$-superior to x is different from the set of alternatives that are $\succ$-superior to y, and therefore a rational decision maker needs two different states to account for how future information will be processed conditional on seeing x and conditional on seeing y. This implies that the state complexity of rational choice nearly equals the number of feasible alternatives.[8] Satisficing is much simpler in terms of state complexity since the only relevant information in processing the next available alternative is whether a satisfactory alternative has already appeared.

The first main result of the paper is that utility maximization is procedurally simplest among all choice rules that are robust to order effects.[9] That is, rational choice is *uniquely* simplest among all choice rules that are order-independent, i.e, rules that choose the same alternative from every two lists that are permutations of one another. In addition, rational choice is simplest among all choice rules that are order-independent for lists with just two alternatives, but possibly order-dependent for larger lists. Put another way, any choice rule that is simpler than rational choice is order-dependent. This result provides a possible procedural explanation for why we

[6] Abraham Neyman (1985), Ariel Rubinstein (1986), Dilip Abreu and Rubinstein (1988), and others study state complexity in the context of repeated games. James Dow (1991) and Andrea Wilson (2002) study state complexity in the context of inference making.

[7] A strict preference relation is a complete, asymmetric, and transitive binary relation.

[8] No state is required for processing information conditional on seeing the $\succ$-maximal alternative, because in that case the decision maker can stop and choose the $\succ$-maximal alternative.

[9] Several papers have pointed out that rational choice is "simple." Donald E. Campbell (1978) and Taradas Bandyopadhyay (1988) show that intuitive procedural properties characterize choice correspondences that can be represented as the maximization of a binary relation. Rubinstein (1996) suggests that the frequent appearance of order relations in natural language can partly be explained by the fact that order relations are easy to describe. Gil Kalai (2003) shows that rational behavior is easy to learn in the Probably-Approximately-Correct learning model. Michael Mandler (forthcoming) shows that in a model of choosing by a checklist of criteria, rational agents make choices faster than nonrational agents.

see order effects in real-life decision making: by using the order, decision makers are able to simplify the complexity of making choices.

As mentioned above, the state complexity of rational choice nearly equals the number of feasible alternatives. Hence, when the number of alternatives is large, rational choice becomes cognitively demanding. For example, when choosing among flights the number of possible choices may be very large and processing every two flights differently may be complicated. In such a case, a decision maker may replace utility maximization with a choice rule that is "close" to utility maximization but uses fewer states. One possibility is to satisfice or perhaps fully rank-order only a few flights and compare all other flights to these "pivotal" flights. I investigate this *choice design* problem under the assumption that the objective is to find a choice rule that maximizes expected utility subject to having a few states.

The second main result of the paper is that choice rules that solve this choice design problem are history-dependent satisficing procedures. In a history-dependant satisficing procedure, the decision maker satisfices using several aspiration thresholds that adjust according to the identity of elements that appeared so far in the list. Unlike in optimal search with no recall or with perfect recall, even when a history-dependent satisficer evaluates all the elements in a list, he may end up choosing an element that is neither the utility maximal element (as in search with perfect recall) nor the last element in the list (as in search with no recall).

This result hints at the type of order and other framing effects that may emerge in practice. An optimal history-dependent satisficer tends to choose alternatives from the beginning of the list, so that moving an element, except maybe the last, toward the beginning of the list improves the likelihood it is chosen. In addition to this primacy effect, an optimal history-dependent satisficer also displays a recency effect in the sense that moving an element that is not chosen to the last position in the list may make it the chosen element. Moreover, an optimal history-dependent satisficer may choose, from a shorter list, an alternative whose utility is higher than that of the chosen alternative from a longer list that subsumes the shorter one, in the spirit of the choice overload phenomenon.[10] Finally, in the presence of a default alternative, the decision maker may ignore all the alternatives in some utility range above the default, except the last alternative in the list, in the spirit of the status quo bias. Thus, particular kinds of framing effects, which are considered "biases" in some contexts, are actually optimal when taking procedural costs into account.[11]

I. Describing Choice Behavior

In many real life situations, decision makers choose from lists. Lists are often generated by an external mechanism that arranges the alternatives in some order. This is what happens in the flights, elections, and competitions examples. Lists may also be generated internally by the decision maker. As the decision maker thinks about various courses of action, he may evaluate the different alternatives in some order.

[10] See Sheena S. Iyengar and Mark R. Lepper (2000).

[11] Wilson (2002) makes a similar argument in the context of inference making with limited memory.

The order of the alternatives in a list may affect choice behavior. For example, a decision maker may pay more attention to the first few alternatives in the list and therefore tend to choose alternatives from the beginning of the list. Travel agents displayed such a primacy effect when selecting flights from computer reservation systems. A decision maker may also recall more vividly the last few alternatives in the list and therefore tend to choose an alternative from the end of the list. Judges often display such a recency effect when choosing a winner in a competition.

A choice function from lists (see Rubinstein and Salant 2006) describes choice behavior that is potentially affected by order. Given a finite set $\mathcal{X}$ of N elements, a *list* is a finite sequence of elements from $\mathcal{X}$. A given element may not appear, appear once, or appear multiple times in a given list. A *choice function from lists* assigns to every nonempty list an element from the list, interpreted as the chosen element.

In Herbert Simon's (1955) *satisficing* procedure, for example, the decision maker classifies each element in $\mathcal{X}$ as either satisfactory or not, and chooses from every list the first satisfactory element. If there is no satisfactory element in the list, he chooses the last element. A satisficer displays a primacy effect (moving a satisfactory element toward the beginning of the list increases the likelihood it is chosen) and a recency effect (moving a nonsatisfactory element to the last position in the list increases the likelihood it is chosen).

Choice functions from lists can also describe more subtle order effects. For example, in the *maximizing with a bias* choice procedure, the decision maker has in mind a utility function u and a "bias" function b, both from $\mathcal{X}$ to R_+. He evaluates the elements of a list in order, and begins by designating the first element as a "favorite." When reaching the i'th element, a_i, the decision maker replaces the current favorite y with a_i if $u(a_i) > u(y) + b(y)$, i.e., he gives a bonus to the current favorite in the spirit of the endowment effect (see Daniel Kahneman, Jack L. Knetsch, and Richard H. Thaler 1991). When the list ends, the decision maker chooses the current favorite.

A choice function from lists can also describe behavior that depends on the number of times a given alternative appears in a list. An extreme example is a decision maker who observes advertisements for the different alternatives, and then chooses the alternative for which he saw the largest number of advertisements.

The model of choice from lists nests the standard model of choice in which behavior is not affected by order or by repetition. The benchmark behavior in the standard model is *rational choice* in which the decision maker has in mind a strict preference relation (i.e., a complete, asymmetric, and transitive binary relation) $\succ$ over $\mathcal{X}$, and he chooses the $\succ$-maximal element from every list.

Many other choice procedures are not affected by order or repetition. In R. Duncan Luce and Howard Raiffa's (1957) *frog legs* procedure, for example, the decision maker maximizes one preference relation if a certain alternative (in the context of a restaurant menu, frog legs) appears on the list, and a different preference relation otherwise. Such a decision maker is affected by the content of the choice problem, not by the way in which it is framed. Other examples of frame-independent yet content-dependent choice procedures include Amartya Sen's (1993) *tea party* procedure in which the decision maker chooses the second-best alternative according to his preference relation in order to appear polite; and Itamar Simonson and Amos Tversky's (1992) *compromise* procedure in which the decision maker chooses a median-quality median-priced item as a compromise between the conflicting price and quality considerations.

II. The Decision Making Process

The cognitive complexity of making choices varies across different choice rules. Satisficing, for example, seems less complicated than rational choice. In satisficing, as the decision maker evaluates the alternatives in a list, processing the current alternative depends only on whether a satisfactory alternative already appeared, whereas in rational choice processing an alternative depends on the identity of the best alternative encountered so far. Choosing the second-best alternative, in turn, seems more complicated than rational choice. In choosing the second-best, the way a decision maker processes an alternative depends on the identity of both the best and the second-best alternatives encountered so far.

The first step in comparing the complexity of different choice rules is to specify the cognitive mechanism by which a decision maker processes information and makes choices. I focus on a model in which the decision maker can be in one of several *states*. A state may summarize any information the decision maker considers relevant for making a choice. For example, in the case of rational choice, the relevant information may be the best alternative encountered so far. A state may also reflect the state of mind of the decision maker. For example, in satisficing, the decision maker's state is "unsatisficed" until he encounters a satisfactory alternative.

For a given list, the decision maker processes the alternatives in the list one at a time by moving between his different states. He starts in some predetermined state. When considering the next alternative in the list, he decides (as a function of his state and the current alternative) whether to stop and make a choice or continue. If the decision maker continues, he updates his state as necessary and then considers the next alternative. When the list ends, a decision maker who has not yet decided makes a choice.

I use an *automaton* to model this cognitive mechanism.[12] The building block of an automaton is a set of states, which may be thought of as the states the decision maker uses in processing information. An automaton reads the elements of a list in order, and processes information by moving between its states according to the alternatives it encounters. When the automaton decides to stop or the list ends, the automaton outputs a chosen element.

A. *An Automaton*

The four components of an automaton are (i) a finite set $\mathcal{Q}$ of states; (ii) an *initial* state $q_0 \in \mathcal{Q}$ in which the automaton starts operating; (iii) a transition function $g : \mathcal{Q} \times \mathcal{X} \to \mathcal{Q} \cup \{Stop\}$–if the automaton is in state q and it encounters the element x, it moves to state $g(q,x)$ or *Stops*; and (iv) an output function $f : \mathcal{Q} \times \mathcal{X} \to \mathcal{X}$ –if $g(q,x) = Stop$ or x is the last element in the list, the automaton produces the output $f(q,x)$.

[12] The automaton model is one of the basic tools developed in computer science to investigate computational complexity (see John E. Hopcroft and Jeffrey D. Ullman 1979). Neyman (1985), Rubinstein (1986), Abreu and Rubinstein (1988), and others adapt the automaton model to study procedural aspects in repeated games. Dow (1991) and Wilson (2002) study procedural models of inference-making, which may be thought of as variations of the automaton model.

An automaton operates as follows. For every list $\mathbf{L} = (a_1, \ldots, a_k)$, the automaton starts in the initial state q_0 and reads the elements of the list one after the other. It processes information by moving between states: when the automaton is in state q (which is initially q_0) and it reads the element a_i, it moves to state $g(q, a_i)$ or stops. If the automaton decides to stop or a_i is the last alternative in the list, the automaton chooses the element $f(q, a_i)$. The chosen element $f(q, a_i)$ must appear in the list $\mathbf{L}$. That is:

Implementation. An automaton *implements* a choice function from lists C if it outputs the element $C(\mathbf{L})$ from every list $\mathbf{L}$. An automaton implementing C is *efficient* if there exists no automaton with fewer states that implements C.

This specification of an automaton differs from the one traditionally used in economic theory. First, an automaton can stop before the list ends. This reflects the ability of the decision maker to stop and make a choice at any time. Second, an automaton can condition its output on both its current state and the alternative it currently processes. This enables distinguishing between present information, i.e., the alternative currently processed, and past information, i.e., the alternatives the decision maker already processed. Past information is not readily accessible and thus must be incorporated in a state, while present information is easily available for making a choice without using a state for that purpose.[13]

B. *Examples*

I now discuss examples of choice rules and automata implementing them. For simplicity, I consider an outcome space with four alternatives, $\mathcal{X} = \{1, 2, 3, 4\}$.

Satisficing. Consider a decision maker who classifies the alternatives 3 and 4 as satisfactory and the alternatives 1 and 2 as nonsatisfactory. He chooses from every list the first satisfactory element, or the last element if the list contains no satisfactory elements.

The one-state automaton in Figure 1 implements this satisficing procedure. The circle in the figure is the initial state q_0 of the automaton. The Stop sign represents the ability of the automaton to stop. Edges represent transitions: the automaton stays in state q_0 as long as it sees nonsatisfactory elements (i.e., 1 or 2); once it sees a satisfactory element (i.e., 3 or 4) it stops. The mapping $x \to x$ below state q_0 describes the output of the automaton in state q_0: if the automaton is in state q_0 and x is the last element in the list or $g(q_0, x) = \mathit{Stop}$, then the automaton outputs x.

Given a list, the automaton in Figure 1 reads the elements of the list in order. It stays in state q_0 until it encounters a satisfactory element, in which case it stops and outputs that element. If it did not stop before reaching the end of the list, then the list contains no satisfactory elements and the automaton outputs the last element in the list. Thus the automaton in Figure 1 implements the satisficing procedure above.

[13] In the computer science literature, an automaton in which the output function depends on both the state and the current input is called a Mealy machine. See George H. Mealy (1955).

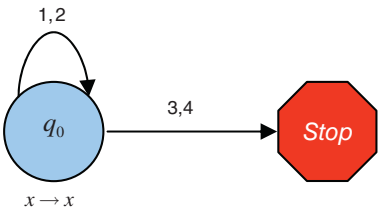

FIGURE 1. SATISFICING

Note that without enabling an automaton to condition its output on the element it currently processes, implementing satisficing would require four states because every alternative among 1, 2, 3, and 4 is a possible output.

Maximizing with a Bias. Consider a decision maker who maximizes with a bias. He assigns the value $u(1) = 0$ to the alternative 1, the value $u(x) = x$ to $x = 2,3,4$, and a bonus of $b = 1.5$ to his favorite alternative. The decision maker designates the first alternative in the list as a favorite. When encountering the element x, he designates x as the new favorite if $u(x) > u(y) + b$, where y is the current favorite. When the list ends, the decision maker chooses the current favorite.

The three-state automaton in Figure 2 implements a maximizing with a bias procedure based on these primitives. The left-most state in the figure q_0 is the initial state that describes how the automaton processes information in the beginning of the list. States q_1 and q_2 describe how the automaton processes information after seeing the element 2 and after seeing the element 3. For example, the automaton ignores the element 3 in state q_1 (i.e., $g(q_1,3) = q_1$ and $f(q_1,3) = 2$) because in state q_1 the decision maker designates the element 2 as a favorite and hence considers 2 superior to 3.

Note that the automaton in Figure 2 is not efficient. When the decision maker encounters the alternative 3 in state q_0, he considers it superior to every other alternative. He can therefore stop and choose it. Hence, by omitting state q_2 and defining $g(q_0,3) = Stop$, we obtain an automaton with fewer states that implements the same procedure. This construction is illustrated in Figure 3.

Contrast. Consider a decision maker who is affected by contrasts. He classifies every alternative as either "conventional" or "nonconventional." From every list, he chooses the first conventional element that appears after two consecutive nonconventional elements. If there is no such element, he chooses the last element in the list. The three-state automaton in Figure 4 implements a contrast procedure in which the elements 2 and 3 are conventional, and 1 and 4 are nonconventional. In the figure, states may be thought of as reflecting how determined the decision maker is to make a conventional choice.

C. *State Complexity*

The complexity of a choice rule is the minimal number of states needed to implement it.

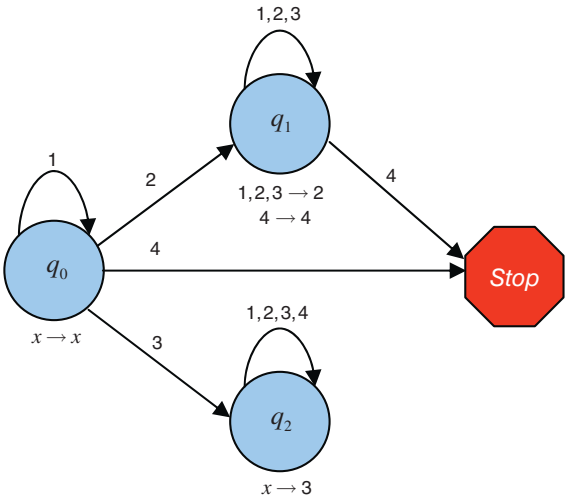

FIGURE 2. MAXIMIZING WITH A BIAS (1)

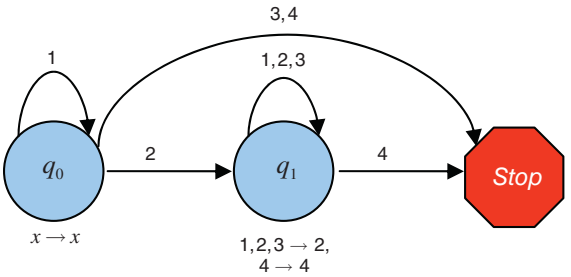

FIGURE 3. MAXIMIZING WITH A BIAS (2)

State Complexity. The *state complexity* of a choice function from lists is the number of states in an efficient automaton implementing the choice function.

State complexity reflects the number of different ways in which a decision maker processes information. Consider, for example, the maximizing with a bias procedure. In this procedure, the decision maker processes information in two different ways, depending on whether the element 2 appears before the element 3 or not. If 2 appears before 3, the decision maker ignores 3 because $u(3) < u(2) + b$. Otherwise, the decision maker chooses 3 because $u(3) + b$ is higher than the utility of any other element. Hence, the decision maker can classify the list of alternatives encountered so far into one of two *categories* depending on whether the list includes the element 2 or not, and then process future information based on this classification. These two categories correspond to states q_1 and q_0 in Figure 3.

The contrast example also illustrates the connection between states and classifying information into categories. In this example, a decision maker who has not yet decided can classify the list of alternatives he encountered so far into one of three categories depending on whether (i) it ends with a conventional element; (ii) it ends with a nonconventional element preceded by a conventional one; and (iii) it ends with

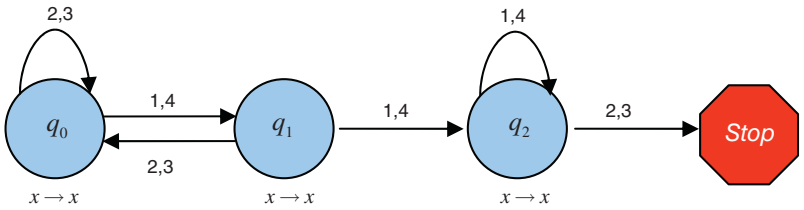

FIGURE 4. NONCONVENTIONAL ELEMENTS HIGHLIGHT THE ATTRACTIVENESS OF CONVENTIONAL ONES

two nonconventional elements. These three categories correspond to states q_0, q_1, and q_2 in Figure 4.

Thus, the states of an efficient automaton may be thought of as categories, each containing information that the decision maker considers relevant for making choices. The decision maker classifies the list of alternatives he encountered so far into one of these categories, and then processes future information based on this classification.

Classifying information into categories is considered by psychologists to be an important cognitive process. In a seminal work, Gordon W. Allport (1954) argues that "the human mind must think with the aid of categories" and outlines the basic principles of categorization. Eleanor Rosch (1978) asserts that categorization provides "maximum information with the least cognitive effort," and Carolyn B. Mervis and Rosch (1981) define that a "category exists whenever two or more distinguishable objects or events are treated equivalently."[14]

In the context of choice from lists, a category is defined as follows. Let C be a choice function. A list $\mathbf{L}$ is *C-undecided* if $\mathbf{L}$ is empty or if there is a list $\mathbf{L}'$ such that $C(\mathbf{L},\mathbf{L}') \neq C(\mathbf{L})$, where $(\mathbf{L},\mathbf{L}')$ denotes a list that consists the elements of $\mathbf{L}$ followed by the elements of $\mathbf{L}'$. Intuitively, a list is C-undecided if after seeing that list, the decision maker is still uncertain about his choice. Two lists $\mathbf{L}_1$ and $\mathbf{L}_2$ are *C-equivalent* if $C(\mathbf{L}_1,\mathbf{L}) = C(\mathbf{L}_2,\mathbf{L})$ for every nonempty list $\mathbf{L}$. Intuitively, two lists are C-equivalent if, after the decision maker sees one of them, future choices are the same as after he sees the other. For example, in satisficing, a list is undecided if and only if it does not contain a satisfactory element, and every two undecided lists are equivalent because, for every continuation, the first satisfactory element in the continuation is chosen.

Category. A *category* is an equivalence class of the binary relation $\sim_C$ defined by $\mathbf{L}_1 \sim_C \mathbf{L}_2$ if $\mathbf{L}_1$ and $\mathbf{L}_2$ are C-undecided and C-equivalent lists.

Thus, a category contains all the lists for which future information processing is required (i.e., the lists are undecided) and is identical (i.e., the lists are equivalent).

[14] Allport (1954) also argues that categorization is an important cognitive source of prejudice and stereotyping. Social psychologists have developed this idea by investigating how categorization affects social interactions and, in particular, how it leads to prejudice (see John F. Dovidio, Peter Glick, and Laurie A. Rudman 2005, for a survey). Recent applications of categorization to economics include Roland Fryer and Matthew O. Jackson (2008), who investigate how experiences are optimally sorted into categories and how categorization affects social decision making, and Paola Manzini and Marco Mariotti (forthcoming), who investigate a boundedly rational choice procedure in which agents categorize alternatives before choosing.

To see the connection between state complexity and classifying lists into categories, suppose that a choice function C classifies undecided lists into K categories. We can think of each category as a state and construct an automaton implementing C with K states as follows. We first designate the state that is associated with the category containing the empty list as the initial state. We then link two states q_1 and q_2 by a transition $g(q_1,a) = q_2$ if the category corresponding to state q_1 contains some list $\mathbf{L}$ and the category corresponding to q_2 contains the list $(\mathbf{L},a)$.[15] Finally, we define the automaton's output in state q when seeing the alternative a by $f(q,a) = C(\mathbf{L},a)$, where $\mathbf{L}$ is any list that belongs to the category corresponding to state q. Because all the lists in a given category are equivalent, this automaton is well defined and it is easy to prove by induction on the length of a list that it implements C. Thus the state complexity of C is at most K.

Consider now an efficient automaton implementing C. All the lists after the processing of which the automaton reaches the same state are C-equivalent because the automaton's future actions depend only on its state and future information. Hence, they belong to the same category, implying that the state complexity of C is weakly larger than K. We therefore obtain that:

PROPOSITION 1: *The state complexity of a choice function C is equal to the number of categories into which C classifies lists.*

Proposition 1 is a modification to the context of individual decision making of the Myhill-Nerode Theorem (see Hopcroft and Ullman 1979) from the computer science literature, and a result by Ehud Kalai and William Stanford (1988) from the repeated games literature. It is proved in the Appendix.

III. State Complexity and Choice Behavior

I now investigate how differences in state complexity relate to regularities in choice behavior. I first show that the simplest choice rules according to state complexity are satisficing procedures, and that rational choice rules are more complicated than satisficing. As mentioned above, satisficing procedures display primacy and recency effects, while rational choice is robust to order effects. I then establish that *any* choice rule that is simpler than rational choice displays order effects.

A. *Satisficing and Rational Choice*

Any satisficing procedure can be implemented by a one-state automaton. Indeed, given a satisficing procedure, we can construct an automaton with one state q_0 implementing the procedure by defining the transition function to be $g(q_0,x) = Stop$ if x is a satisfactory element, and $g(q_0,x) = q_0$ otherwise, and by defining the output function to be $f(q_0,x) = x$. That is, the automaton stays in the initial state until it encounters a satisfactory element, in which case it stops and chooses it. If the

[15] If the category corresponding to state q_1 contains some list $\mathbf{L}$ such that the list $(\mathbf{L},a)$ is not undecided, then $g(q_1,a) = Stop$.

automaton did not stop before reaching the end of the list, it outputs the last element in the list. Figure 1 illustrates this construction.

The other direction is also true: if a choice function can be implemented by a one-state automaton, then it can be represented as a satisficing procedure. To see this, consider an automaton with one state q_0 that implements a choice function C. Because the automaton has only one state, it cannot use any past information (except for the fact that it has not stopped yet) in making choices. In particular, its output in state q_0, when seeing an element x, must be x (otherwise, the automaton outputs an incorrect choice from some one-element list). Labeling an element x as satisfactory if the automaton stops when seeing it, and as nonsatisfactory otherwise, we obtain that C is consistent with choosing the first satisfactory element from every list, or the last element in the list if no satisfactory element appears in the list. Thus:

PROPOSITION 2: *A choice function has minimal state complexity if and only if it can be represented as a satisficing procedure.*

Rational choice is more complicated than satisficing. Intuitively, in rational choice, information processing depends on the identity of the best element seen so far, and hence the complexity of rational choice nearly equals the number of feasible alternatives.

PROPOSITION 3: *The procedural complexity of rational choice is* $N - 1$, *where* $N = |\mathcal{X}|$ *is the number of feasible alternatives.*

To see that fewer than $N - 1$ states do not suffice for implementing rational choice, let $x_{\max}$ and $x_{\min}$ denote the utility maximal element and the utility minimal element in the outcome space $\mathcal{X}$. Suppose the decision maker encounters an alternative $x \neq x_{\max}$ in the beginning of a list. Because $x_{\max}$ may appear down the list, the decision maker cannot stop when seeing x, and because x is a possible choice, the decision maker cannot move to the same state he would have moved to were $y \neq x$ the first element in the list (since in this case the decision maker will not know which element of x and y to choose if $x_{\min}$ would be the second and last element). Thus, the decision maker needs at least a number of states that is equal to the number of elements in the outcome space excluding $x_{\max}$, i.e., $N - 1$ states.

To see that $N - 1$ states suffice for implementing rational choice, note that when seeing $x_{\max}$, the decision maker can stop and choose it. When seeing any other element x in the beginning of the list, the decision maker can move to a state that "records" that x was seen.[16] In this state, the decision maker ignores alternatives that are inferior to x, and when encountering an alternative y that is superior to x, the decision maker moves to a state that records y. The decision maker continues processing information in this fashion until $x_{\max}$ appears, in which case he stops and chooses $x_{\max}$, or the list ends, in which case he chooses the utility-higher alternative among the one recorded in the current state and the last alternative. Figure 5 illustrates this construction for the case of maximizing the preference relation $4 \succ 3 \succ 2 \succ 1$. In the figure, state q_i

[16] Think about the initial state as recording that $x_{\min}$ was seen.

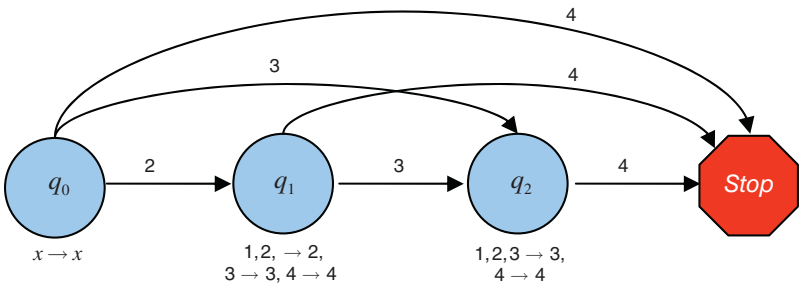

FIGURE 5. RATIONAL CHOICE

corresponds to how information is processed conditional on the element $i + 1$ being the $\succ$-maximal element seen so far (the figure does not depict self-loops).

By Proposition 3, a decision maker can reduce the difficulty of maximizing by shrinking the number of possible choices. Consider, for example, a decision maker who has in mind a *default* alternative. The default alternative may be an alternative the decision maker chose in the past, an alternative that was recommended to him, or simply the option of not making a choice. In making choices, the decision maker either keeps the default or chooses from the list of available alternatives. If the default alternative is superior to some alternatives in the outcome space $\mathcal{X}$, then a rational decision maker can ignore all the alternatives that are inferior to the default in making choices. In this case, he maximizes over a smaller outcome space, and by Proposition 3 the complexity of making choices is smaller. If the decision maker attaches a utility bonus to the default in addition to its intrinsic value (as in the status-quo bias), then the set of elements that are superior to the default shrinks further, and making choices becomes even simpler.

A decision maker may also reduce the difficulty of maximizing by affecting the ordering of the alternatives. For example, in thinking about which product to purchase, the decision maker may consider the alternatives in increasing price order. If the resulting ordering is correlated with the decision maker's preference relation $\succ$, maximizing $\succ$ may be simpler. For example, if $\succ$ is single-peaked with respect to price and the decision maker organizes products according to price, then once the decision maker "passes his peak" he chooses the $\succ$-maximal alternative among the $\succ$-best alternatives before the peak and the first one after the peak. That is, no states are required for processing elements that appear after the peak. Of course, if presorting generates an ordering that is identical to $\succ$, as may be the case in the list of results returned by a search engine, making choices becomes trivial.

B. *Rational Choice and Order Effects*

One difference between rational choice and satisficing is that rational choice is robust to order effects while satisficing is not. I now show that in comparing rational choice to simpler behaviors this is a general phenomenon: any choice rule that is procedurally simpler than rational choice displays order effects. Put another way, from the procedural perspective, rational choice is the simplest behavior that is robust to order effects.

Order-independent choice. A choice function C is *order-independent* if for every list $(a_1, a_2, \ldots, a_k)$ and every permutation σ of $\{1, \ldots k\}$, it holds that $C(a_1, a_2, \ldots, a_k) = C(a_{\sigma(1)}, a_{\sigma(2)}, \ldots, a_{\sigma(k)})$. Thus, a choice function is order-independent if changing the ordering of the alternatives does not affect choice behavior. If a choice function C satisfies this condition only for $k = 2$, then C is *order-independent for pairs*.

Order-independent choice functions can be classified according to whether they are rationalizable or not. A choice function C is *rationalizable* if there exists a preference relation $\succ$ such that for every list $\mathbf{L}$, $C(\mathbf{L})$ is the $\succ$-maximal element in $\mathbf{L}$. Otherwise, C is not rationalizable.

THEOREM 1: *If a choice function is order-independent for pairs then it is at least as complicated as rational choice. If a choice function is order-independent and is not rationalizable, then it is strictly more complicated than rational choice.*

Theorem 1 implies that any choice function that is simpler in terms of state complexity than rational choice inevitably displays order effects. If the state complexity of a choice function is smaller than that of rational choice, then there are two alternatives, a and b, such that the decision maker either chooses the first alternative from both lists, (a,b) and (b,a), or the second alternative from both lists. If the state complexity of a choice function C is equal to that of rational choice and C is not rationalizable, then changing the order of the alternatives in some list (not necessarily a list of two alternatives) alters choice behavior. This provides a possible procedural explanation for order effects: by using the order, the decision maker can simplify the complexity of making choices.

Theorem 1 also implies that every standard choice function that is not affected by order and cannot be described by the maximization of a preference relation is more complicated than maximizing a preference relation. Examples of such choice functions include Luce and Raiffa's frog legs procedure, Sen's tea party procedure, and Simonson and Tversky's compromise procedure. In these, and in any other standard choice function that is not rationalizable, making choices depends on information *beyond* the alternative that the decision maker would choose if the list ended now. This implies a higher state complexity than in rational choice in which information processing depends *only* on the alternative that would be chosen if the list ended now.

Theorem 1 extends to certain forms of repetition effects. A choice function C is *repetition-independent* if $C(\mathbf{L}) = C(\mathbf{L}, x) = C(x, \mathbf{L})$ for every list $\mathbf{L}$ and for every element x that appears in $\mathbf{L}$. That is, a choice function is repetition-independent if adding an element that already appears in the list to the beginning or to the end of the list does not alter choice. If this condition holds only for two-element lists, then C is *repetition-independent for pairs*.

Repetition-independence implies order-independence. Indeed, if C is repetition-independent for pairs, then $C(a,b) = C(a,b,a) = C(b,a)$ for every two elements a and b, and thus C is order-independent for pairs. Similarly, if C is repetition-independent, then it is order-independent.[17] Thus:

[17] Otherwise, if C is order-dependent, then there exist lists $\mathbf{L}_1$ and $\mathbf{L}_2$ that are permutations of one another such that $C(\mathbf{L}_1) \neq C(\mathbf{L}_2)$. By repetition independence, however, we obtain that $C(\mathbf{L}_1) = C(\mathbf{L}_1, \mathbf{L}_2) = C(\mathbf{L}_2)$.

COROLLARY 1: *If a choice function is repetition-independent for pairs, then it is at least as complicated as rational choice. If a choice function is repetition-independent and is not rationalizable, then it is strictly more complicated than rational choice.*

I now prove Theorem 1, relegating the technical details to the Appendix.

PROOF OF THEOREM 1.

To prove the first part of the theorem, consider an automaton implementing a choice function C that is order-independent for pairs. If the automaton stops after seeing an element x in the beginning of a list, then it cannot stop after seeing a different element y in the beginning of a list. Otherwise, the automaton chooses the first element from the lists (x,y) and (y,x), violating order-independence for pairs. Similarly, the automaton cannot move to the same state q after seeing an element a in the beginning of a list and after seeing an element $b \neq a$ in the beginning of the list. Otherwise, if $C(a,b) = a$, then the automaton moves to state q after seeing a, and then outputs $f(q,b) = a$. But this implies that the automaton also chooses a from the list (b,b) and thus does not implement C.

To summarize, in order to keep track of the information required to implement C, an automaton needs to process every element (excluding at most one element) using a different state. The automaton therefore has a number of states that is equal to or larger than the number of elements in the outcome space (excluding at most one element), implying that C is at least as complicated as rational choice.

To prove the second part of the theorem, let C be an order-independent choice function. By the first part of the theorem, the state complexity of C is at least $N-1$. Thus, to conclude the proof, it is enough to show that if the state complexity of C is exactly $N-1$, then C is rationalizable.

The following two Lemmas that are proved in the Appendix assert (i) that C has a "best" element that is chosen whenever it is available and a "worst" element that is never chosen whenever other elements are available, and (ii) that C is robust to repetition effects.

LEMMA 1: *If a choice function C is order-independent and has complexity $N-1$, then there are two elements $x_{\max}$ and $x_{\min}$ such that $C(x_{\max},\mathbf{L}) = x_{\max}$ for every list $\mathbf{L}$ and $C(x_{\min},\mathbf{L}) = C(\mathbf{L})$ for every list $\mathbf{L}$ that contains elements other than $x_{\min}$.*

LEMMA 2: *If C is order-independent and has complexity $N-1$, then $C(a,\mathbf{L}) = C(\mathbf{L})$ for every list $\mathbf{L}$ and for every element a that appears in $\mathbf{L}$.*

Consider an automaton with $N-1$ states implementing C. By Lemma 1, when seeing an element $x \neq x_{\max}$ in the beginning of a list, the automaton cannot stop because the element $x_{\max}$ may appear down the list. In addition, the automaton cannot move to the same state after seeing x in the beginning of a list and after seeing a different element y in the beginning of a list because the next and last element may be the worst element $x_{\min}$, in which case the automaton will not know whether to output x or y. We can thus identify any state q with a unique element $x \in \mathcal{X}\backslash\{x_{max}\}$ that triggers the transition to state q when it appears in the beginning of the list.

By Lemma 2, C is defined over sets, i.e., neither order nor repetition affects choice behavior. Therefore, to prove that C is rationalizable, it is enough to show that C satisfies the Independence of Irrelevant Alternatives (IIA) property. The IIA property says that omitting an alternative that is not chosen from a choice problem does not alter choice. In the context of choice from sets, it is well known that a choice function satisfies the IIA property if and only if the function is rationalizable.

Assume, to the contrary, that C violates the IIA property. Then, there exists a set of alternatives $\mathcal{A}$ and an element $x \notin \mathcal{A}$ such that $C(\mathbf{L}_{\mathcal{A}},x) \in \mathcal{A}$ but $C(\mathbf{L}_{\mathcal{A}},x) \neq C(\mathbf{L}_{\mathcal{A}})$, where $\mathbf{L}_{\mathcal{A}}$ denotes some listing of the elements of the set $\mathcal{A}$. That is, although x is not chosen from $(\mathbf{L}_{\mathcal{A}},x)$, choice behavior is affected by omitting x from $(\mathbf{L}_{\mathcal{A}},x)$.

The list $\mathbf{L}_{\mathcal{A}}$ does not include the element $x_{\max}$; otherwise by Lemma 1, the element $x_{\max}$ would be chosen from both $\mathbf{L}_{\mathcal{A}}$ and $(\mathbf{L}_{\mathcal{A}},x)$. Hence, as the automaton processes the elements of the list $\mathbf{L}_{\mathcal{A}}$, it does not stop because $x_{\max}$ may appear down the list. Moreover, after processing the elements of $\mathbf{L}_{\mathcal{A}}$, the automaton reaches the state q associated with the element $C(\mathbf{L}_{\mathcal{A}})$ (were it moved to any other state and $x_{\min}$ was the next and last element, the automaton would output an incorrect choice).

Thus, because of the lack of states, the automaton reaches the same state after processing the element $C(\mathbf{L}_{\mathcal{A}})$ and the list $\mathbf{L}_{\mathcal{A}}$. Therefore, the automaton outputs the same alternative y from the lists $(C(\mathbf{L}_{\mathcal{A}}),x)$ and $(\mathbf{L}_{\mathcal{A}},x)$. This means that the automaton fails to implement C. Indeed, if $y \neq C(\mathbf{L}_{\mathcal{A}},x)$, then the automaton outputs an incorrect choice from $(\mathbf{L}_{\mathcal{A}},x)$, and if $y = C(\mathbf{L}_{\mathcal{A}},x)$, then the automaton outputs an infeasible choice from $(C(\mathbf{L}_{\mathcal{A}}),x)$ because we picked the set $\mathcal{A}$ and the element x so that (i) the element $C(\mathbf{L}_{\mathcal{A}})$ differs from y (because $y = C(\mathbf{L}_{\mathcal{A}},x)$ and $C(\mathbf{L}_{\mathcal{A}}) \neq C(\mathbf{L}_{\mathcal{A}},x)$), and (ii) the element x differs from y (because $y = C(\mathbf{L}_{\mathcal{A}},x) \in \mathcal{A}$ and $x \notin \mathcal{A}$). We therefore obtain that C satisfies IIA and hence it is rationalizable.

IV. Choice Design

Rational choice requires using different states to process different alternatives. This may be cognitively demanding, especially when the number of feasible alternatives is large. For example, when choosing among flights, the number of possible choices may be very large and processing every flight differently may be complicated. In such cases, the decision maker may replace utility maximization with a choice procedure that is "close" to maximizing utility but uses fewer states.

Theorem 1 establishes that any choice rule that uses fewer states than rational choice displays order effects. Theorem 1 is silent, however, on the nature of these order effects. In this section, I explore what order and other framing effects emerge when a decision maker optimally approximates rational choice using a simpler choice rule.

The first step in investigating this question is to specify how well a given choice rule performs with respect to utility maximization. This depends on which lists the decision maker is expected to choose from, which depends in turn on the process that generates lists. In some cases, the choice designer may not have a clear perception of this process. He may then be interested in a choice rule that performs well in the worst possible case. In other cases, the designer may know the details of the process that generates lists. He may then use a measure of performance that depends on the specific process, such as maximizing expected utility. Both

maximizing expected utility and performing well in the worst case result in choice rules that display similar framing effects. I will focus on maximizing expected utility, and comment on performing well in the worst case when stating the main result of this section.

In what follows, lists are assumed to be generated by a stationary process. The first element in the list is drawn according to a probability measure P over the outcome space $\mathcal{X}$, where $P(x) > 0$ for every $x \in \mathcal{X}$. With probability $\mu > 0$, the process ends. With probability $1 - \mu$, the process continues and an additional element is drawn according to P. The process continues in this fashion until the list ends.

Given a choice function C and a utility function u, let $u(C(\mathbf{L}))$ denote the utility of the element C chooses from the list $\mathbf{L}$, and let $E(u(C(\mathbf{L})))$ denote the expected utility C obtains, where the expectation is taken with respect to the process above. The objective of the following choice design problem is to maximize expected utility subject to having only K states to process information:

$$\max_{C\,:\,comp(C)=K} E(u(C(\mathbf{L}))),$$

where $comp(C)$ denotes the state complexity of C.[18]

When there is only one state, the decision maker cannot use any past information in making choices. Hence, when encountering an alternative he either stops and chooses it or ignores it, depending on whether the utility of that alternative is higher than the value of continuing. If the utility of the current alternative exceeds the continuation value, then the alternative is satisfactory and the decision maker stops and chooses it. Otherwise, the alternative is not satisfactory and the decision maker continues to the next alternative. Thus, the decision maker satisfices with an aspiration level that corresponds to the continuation value.

When $K > 1$ states are available, the decision maker can use $K - 1$ pieces of information in making choices (in the initial state, he cannot use any past information). In the optimal solution, these pieces of information correspond to $K - 1$ different alternatives because the process that generates lists is stationary. The decision maker maximizes for these $K - 1$ "pivotal" alternatives. Processing the remaining alternatives is done as follows. The decision maker sets an aspiration level in the initial state and $K - 1$ additional aspiration levels that are associated with the $K - 1$ pivotal alternatives. When seeing an alternative x that is not pivotal, the decision maker satisfices with the current aspiration level: he chooses x if x exceeds the current aspiration level, and ignores x otherwise. The current aspiration level is the initial aspiration level as long as the decision maker did not encounter a pivotal element, and the aspiration level associated with the best encountered pivotal element afterward.

More formally, a *K-phase satisficing* procedure is characterized by an initial aspiration level t_0 and $K - 1$ higher aspiration levels $t_1 < \ldots < t_{K-1}$ that correspond to $K - 1$ pivotal alternatives $a_1, \ldots, a_{K-1}$, such that $u(a_1) < \ldots < u(a_{K-1})$. For every list, the decision maker starts in phase 0 and satisfices with the aspiration threshold

[18] It is straightforward to extend the analysis to situations in which each state is costly and the designer also chooses the number of states.

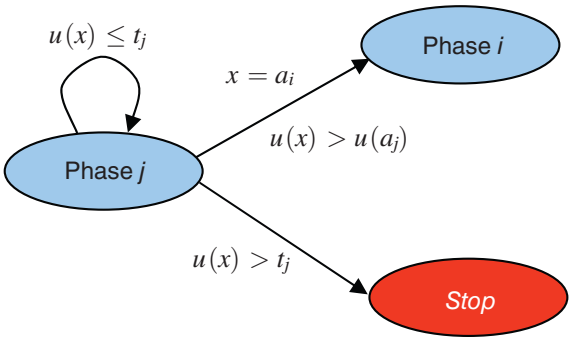

FIGURE 6. HISTORY-DEPENDENT SATISFICING

t_0 until he encounters an element a_j and moves to phase $j > 0$. In phase j, the next element x in the list is processed as follows (see Figure 6 for an illustration):

(i) If x is the last element in the list, the decision maker chooses the u-maximal element among a_j and x.

(ii) If $x = a_i$ and $u(x) > u(a_j)$, the decision maker switches to phase i.

(iii) Otherwise, the decision maker satisfices with the threshold t_j : if $u(x) > t_j$ he chooses x and stops, and if $u(x) \leq t_j$ he ignores x and stays in phase j.

THEOREM 2: *Any choice function with state complexity K that maximizes expected utility can be represented by a K-phase satisficing procedure.*[19]

By Theorem 2, the optimal way of approximating rational choice with K states is to maximize over $K - 1$ pivotal alternatives and to satisfice (conditional on the best encountered pivotal element) for the remaining alternatives. Operationally, this K-phase satisficing procedure is simpler than utility maximization in the sense that the decision maker rank-orders only a few alternatives. The remaining alternatives are not fully evaluated, but rather are compared only to the pivotal elements and the corresponding thresholds. Hence, this procedure uses a coarser classification of information than rational choice.

A K-phase satisficer displays several behavioral effects. First, he displays a *primacy* effect: by moving an element (except maybe the last) toward the beginning of the list, the likelihood that the decision maker will choose that element increases. This is so because aspiration levels increase along the list and because the decision

[19] Theorem 2 extends to minimizing the worst-case performance as follows. Given a list $\mathbf{L}$, let $u(\mathbf{L})$ be the utility of the u-maximal element in $\mathbf{L}$, and let $u(C(\mathbf{L}))$ be the utility of the element that C chooses from $\mathbf{L}$. The regret of C with respect to u, $regret_u(C) = \max_{\mathbf{L}\in\mathcal{L}}\{u(\mathbf{L}) - u(C(\mathbf{L}))\}$, is the maximal utility loss associated with making choices according to C rather than maximizing u, where $\mathcal{L}$ is the set of all nonempty lists. When the choice designer seeks to minimize the regret associated with making choices, i.e., solve $\min_{C\,:\,Comp(C)=K} regret_u(C)$, it can be shown that subject to two efficiency refinements, the generically unique choice function that solves this problem is a K-phase satisficing procedure.

maker chooses the first element above the current aspiration level. Second, the decision maker displays a mild *recency effect*: by moving an element x that is not chosen to the last position in the list, it may be chosen. This could happen when the utility of x is smaller than the current aspiration level but larger than the utility of the corresponding aspiration element. In such a case, the decision maker ignores x in the middle of the list because the continuation value is high, but chooses x if it appears last.

Moreover, for a K-phase satisficer, having more choices is not necessarily better: adding an element x to a list $\mathbf{L}$ may result in a utility-inferior choice, either because x is satisfactory but the utility of x is smaller than that of the element previously chosen from $\mathbf{L}$, or because x is a pivotal element that increases the aspiration level enough so that a previously chosen alternative is no longer chosen. This may be interpreted as an example of the choice overload hypothesis (see Iyengar and Lepper 2000).

Finally, when the decision maker is endowed with a default alternative, he may display a tendency to favor the default over alternatives with higher utility. Indeed, in the presence of a default alternative, the decision maker's aspiration level in the initial phase will be higher than the utility of the default because the decision maker has the option of choosing the default. Thus, the decision maker will ignore elements with utility higher than that of the default but lower than the initial aspiration level, except maybe the last alternative in the list.

V. Concluding Remarks

This paper investigated one procedural aspect of making choices. I considered automata implementing choice rules, and measured the procedural complexity of a given choice rule by the minimal number of states required for implementing it. State complexity reflects the amount of information processing involved in making choices.

In rational choice, information processing depends on the identity of the best alternative considered so far, so the complexity of rational choice nearly equals the number of feasible alternatives. When the number of feasible alternatives is large, rational choice may therefore be cognitively demanding. In this case, a decision maker may replace maximization with a simpler choice rule to economize on procedural costs.

The paper establishes that in order to simplify the complexity of making choices, the decision maker has to take advantage of the order in which the alternatives appear. Doing so optimally results in more concrete order effects, including primacy and recency effects, and in additional behavioral regularities, including choice overload and default tendency. These effects emerge as optimal when a rational decision maker economizes on procedural costs.

In evaluating these results, one should keep in mind that there are cognitive and procedural aspects that are not captured by state complexity. In a basic search model, for example, the source of procedural complexity is the "marginal" cost associated with searching for the next element in a list rather than the "fixed" cost of implementing the choice rule. Analyzing how state complexity interacts with this and other procedural aspects is clearly important for gaining a deeper understanding of various behavioral regularities. This is a task for future research.

APPENDIX

A. *Proof of Proposition 1*

Fix a choice function C such that $\sim_C$ has K equivalence classes. It is enough to show that the state complexity of C is K. To do so, I first show that any automaton implementing C has $\geq K$ states. I then construct an automaton with K state implementing C.

To see that any automaton implementing C has $\geq K$ states, assume to the contrary that there exists an automaton $\mathcal{M}$ with fewer than K states that implements C. Because the number of $\sim_C$-equivalence classes is K, there are K undecided lists, $\mathbf{L_1},\ldots,\mathbf{L_K}$, such that every two of them are not equivalent. Because these lists are undecided, $\mathcal{M}$ does not stop while or after processing any of them. Because $\mathcal{M}$ has fewer than K states, there are two lists, $\mathbf{L_i}$ and $\mathbf{L_j}$, such that $\mathcal{M}$ reaches the same state after processing any of them. But then $\mathcal{M}$ chooses the same element from $(\mathbf{L_i},\mathbf{L})$ and $(\mathbf{L_j},\mathbf{L})$ for every continuation $\mathbf{L}$, in contradiction to $\mathbf{L_i}$ and $\mathbf{L_j}$ not being equivalent.

I now construct an automaton $\mathcal{M}$ with K states that implements C. Denote the equivalence class of an undecided list $\mathbf{L}$ by $[\mathbf{L}]_{\sim_C}$. The states of $\mathcal{M}$ are the K equivalence classes of $\sim_C$. The initial state is the equivalence class of the empty list. The transition function maps a state q and an element $a \in \mathcal{X}$ to another state as follows: take any list $\mathbf{L}$ in the equivalence class q and move to state $[(\mathbf{L},a)]_{\sim_C}$ if the concatenated list $(\mathbf{L},a)$ is undecided; if $(\mathbf{L},a)$ is decided (i.e., $(\mathbf{L},a)$ is *not* undecided), Stop. Let $f(q,a) = C(\mathbf{L},a)$ for some list $\mathbf{L}$ in q.

The functions f and g are well-defined functions. Indeed, assume $\mathbf{L}_1 \sim_C \mathbf{L}_2$. Then $C(\mathbf{L}_1,a) = C(\mathbf{L}_2,a)$ for every element a and thus f is well defined. In addition, because $\mathbf{L}_1$ and $\mathbf{L}_2$ are undecided and $C(\mathbf{L}_1,\mathbf{L}) = C(\mathbf{L}_2,\mathbf{L})$ for every nonempty list $\mathbf{L}$, the list $(\mathbf{L}_1,a)$ is undecided if and only if $(\mathbf{L}_2,a)$ is undecided. If both are undecided, then $(\mathbf{L}_1,a) \sim_C (\mathbf{L}_2,a)$. Thus g is well defined.

It remains to show that $\mathcal{M}$ implements C. I do so by induction on the length of the list. For every one-element list $\mathbf{L} = (a)$, the automaton outputs $f(q_0,(a)) = C(\mathbf{L},a)$ for every list $\mathbf{L}$ in the equivalence class q_0. By taking $\mathbf{L}$ to be the empty list, we obtain that $\mathcal{M}$ outputs $C(a)$ as required. Consider a list $\mathbf{L} = (\mathbf{L}',a)$ where $\mathbf{L}'$ is nonempty. If $\mathbf{L}'$ is undecided, then by construction $\mathcal{M}$ reaches state $[\mathbf{L}']_{\sim_C}$ after reading $\mathbf{L}'$. Since a is the last element in the list, $\mathcal{M}$ outputs $f([\mathbf{L}']_{\sim_C},a)$. By the definition of $f, f([\mathbf{L}']_{\sim_C},a) = C(\mathbf{L}',a) = C(\mathbf{L})$. If $\mathbf{L}'$ is decided, denote by $\mathbf{L}''$ the shortest beginning of $\mathbf{L}'$ that is decided. By construction, the element $\mathcal{M}$ chooses from $\mathbf{L}$ is the element $\mathcal{M}$ chooses from $\mathbf{L}''$. By the induction assumption, the element $\mathcal{M}$ chooses from $\mathbf{L}''$ is $C(\mathbf{L}'')$, and by the fact that $\mathbf{L}''$ is decided $C(\mathbf{L}'') = C(\mathbf{L})$. Thus, $\mathcal{M}$ outputs $C(\mathbf{L})$ as required.

B. *Proof of Lemmas Stated in the Proof of Theorem 1*

PROOF OF LEMMA 1:

Consider an automaton with $N-1$ states implementing C. If there is no element $x_{\max}$ such that $C(x_{\max},\mathbf{L}) = x_{\max}$ for every list $\mathbf{L}$, then the automaton never stops in the initial state, i.e., $g(q_0,x) \neq \textit{Stop}$ for every $x \in \mathcal{X}$. Since the automaton has $N-1$ states and the number of feasible alternatives is N, there exist two elements

x and y such that the automaton moves to the same state q after seeing any of them in the beginning of the list, i.e., $g(q_0,x) = g(q_0,y) = q$. Assume without loss of generality that $C(x,y) = C(y,x) = x$. Then, when seeing y in state q, the automaton needs to output x because it is possible that the element that caused the transition to state q is x and thus the automaton needs to choose $C(x,y)$. But this implies that the automaton outputs the element x from the list (y,y) and hence it does not implement C. Thus, there exists an element $x_{\max}$ as required. The proof of the second part of the Lemma is similar.

PROOF OF LEMMA 2:

Consider an automaton with $N-1$ states implementing C. By Lemma 1 and the explanation that follows Lemma 1 in the main text, we can identify any state q of the automaton with a unique element $x \in \mathcal{X}\backslash\{x_{max}\}$ that triggers the transition to state q when x appears in the beginning of the list.

Let $\mathbf{L}$ be some list and a an element that appears in $\mathbf{L}$. If $a \in \{x_{\max}, x_{\min}\}$, then $C(a,\mathbf{L}) = C(\mathbf{L})$ by Lemma 1. Consider an element $a \notin \{x_{\max}, x_{\min}\}$. Denote by $\mathbf{L}'$ a sublist of $\mathbf{L}$ in which one instance of a is omitted. Because C is order-independent, $C(\mathbf{L}) = C(a,\mathbf{L}')$ and $C(a,\mathbf{L}) = C(a,a,\mathbf{L}')$. Thus, to prove that $C(a,\mathbf{L}) = C(\mathbf{L})$, it is enough to prove that $C(a,\mathbf{L}') = C(a,a,\mathbf{L}')$, and to prove the latter equality, it is enough to prove that $g(q_a,a) = q_a$, where q_a is the state to which the automaton moves after seeing a in the beginning of the list. Assume this is not the case. Then, $g(q_a,a) = q_b$, where q_b is the state to which the automaton moves after seeing b in the beginning of the list ($g(q_a,a) \neq Stop$ because $x_{\max}$ may appear down the list). For the automaton to make a correct choice from the list $(b,x_{\min})$, it must be that $f(q_b,x_{\min}) = b$, but then the automaton outputs b from the list $(a,a,x_{\min})$, which is an incorrect choice. Thus, $g(q_a,a) = q_a$ as required.

C. *Proof of Theorem 2*

Because there is a finite number of choice functions with state complexity K, there exists a choice function C that solves $\max_{C\,:\,comp(C)=K} E(u(C(\mathbf{L})))$. Let $\mathcal{M}$ be an automaton with K states implementing C, and denote its initial state by q_0 and its remaining states (if $K > 1$) by $q_1,\ldots,q_{K-1}$. Let $V_i = E(u(C(\mathbf{L}))|q_i)$ be the continuation value associated with state q_i. Denote by $x_{\min}$ the u-minimal element in the outcome space $\mathcal{X}$. The following Lemma characterizes the output function of $\mathcal{M}$.

LEMMA 3: *In every state q_i there exists an element $a_i \in \mathcal{X}$ such that $f(q_i,x) = \operatorname{argmax}\{u(a_i),u(x)\}$ for every $x \in \mathcal{X}$. Moreover $a_i = x_{\min}$ for $i = 0$, and $a_i \neq x_{\min}$ otherwise.*

PROOF:

Let $\mathcal{A}_i = \{w\,|\,f(q_i,x) = w \text{ for some } x \neq w\}$. The set $\mathcal{A}_i$ contains at most one element. To see this, assume to the contrary that $\mathcal{A}_i$ contains two different elements w_1 and w_2 such that $u(w_1) > u(w_2)$. Then $\mathcal{M}$ always processes both alternatives w_1 and w_2 before reaching state q_i. Redefine the output function in state q_i to output w_1 whenever it outputted w_2 to obtain a new automaton that achieves a higher expected utility than $\mathcal{M}$. Thus $\mathcal{A}_i$ is either empty (e.g., in the initial state) or contains a single

element. Define $a_i = x_{\min}$ if $\mathcal{A}_i$ is empty and otherwise define a_i to be the unique alternative in $\mathcal{A}_i$. We have that $f(q_i,x) = \text{argmax}\{u(a_i),u(x)\}$ for every x because otherwise we can redefine $f(q_i,x) = \text{argmax}\{u(a_i),u(x)\}$ to obtain an improvement over $\mathcal{M}$.

To complete the proof, assume to the contrary that $a_i = x_{\min}$ for $i \neq 0$. Then the output function in states q_0 and q_i is identical. In particular, there is no element w (expect maybe $x_{\min}$) that the automaton always processes before reaching state q_i. (If there were such an element, we could redefine the output function to obtain a new automaton that performs better than $\mathcal{M}$.)

This implies that $V_i = V_0$. To see this, note that if $V_i > V_0$, then we can obtain an improvement over $\mathcal{M}$ by redefining the initial state to be q_i. If $V_0 > V_i$, then we can redefine the transition function in state q_i to mimic the transition function in state q_0 to obtain an improvement over $\mathcal{M}$.

But if $V_0 = V_i$, then we can also construct a automaton $\mathcal{M}'$ that performs better than $\mathcal{M}$ by replacing state q_i with a new state q' as follows. First, we shift all the incoming transitions to state q_i to enter state q_0 instead (i.e., if $g(q,x) = q_i$ in $\mathcal{M}$ then we define $g(q,x) = q_0$ in $\mathcal{M}'$). All other transitions and outputs in $\mathcal{M}'$ are identical to $\mathcal{M}$. Because there are no incoming transitions to state q_i in $\mathcal{M}'$, we omit q_i in $\mathcal{M}'$. Thus $\mathcal{M}'$ obtains the same expected utility as $\mathcal{M}$ with one less state. We now add a state q' to $\mathcal{M}'$ that improves its performance as follows. Because $\mathcal{M}'$ has $K-1 < N-2$ states and there are N possible alternatives, there is an element $w \notin \{x_{\min},x_{\max}\}$ such that either (i) $g(q_0,w) = \textit{Stop}$; or (ii) $g(q_0,w) = g(q_0,x) = q_m$ for some element x. Suppose (i) holds. Define $g(q_0,w) = q', g(q',x) = \textit{Stop}$ and $f(q',x) = \text{argmax}\{u(w),u(x)\}$. That is, q' either outputs w or a u-higher element, and thus the expected utility conditional on seeing w in the initial state is now larger. Suppose (ii) holds. Then the output function in state q_m is identical to the output function in q_0, and by the argument above, $V_m = V_0$. Define $g(q_0,w) = q', g(q',x) = g(q_0,x)$ and $f(q',x) = \text{argmax}\{u(w),u(x)\}$ for every element x. By definition, the continuation value in state q' is larger than $V_m = V_0$ because q' imitates q_0, except for sometimes outputting an element with a larger utility. This implies that the expected utility conditional on seeing w in the initial state is now larger, and thus we obtain an improvement over $\mathcal{M}$.

Lemmas 4 and 5 characterize the continuation values in the different states.

LEMMA 4: *If q_i and q_j are two states such that $u(a_i) > u(a_j)$, then $V_i > V_j$.*

PROOF:

Suppose not. Pick q_j to be a state associated with u-smallest element a_j for which this happens (if there is more than one such state, pick q_j to be the one with the highest continuation value). Since the outputs in state q_i are sometimes identical and sometimes u-larger than the outputs in state q_j, the continuation value in state q_j is weakly larger than in state q_i only if there is a transition $g(q_j,x) \neq q_j$ for some element x that results in a u-higher expected value than the transition $g(q_i,x)$ in state q_i. But this is impossible because we can always replicate this transition in state q_i and thus obtain a higher continuation value in state q_i. Indeed, if $g(q_j,x) = \textit{Stop}$, then we can define $g(q_i,x) = \textit{Stop}$ as well. Otherwise the transition $g(q_j,x)$ is to state q_m such

that $u(a_m) > u(a_j)$. (Because we picked a_j so that all continuation values in states associated will u-lower elements are lower than V_j, we must have $u(a_m) > u(a_j)$.) In this case, $x = a_m$. (It cannot be that a_m appears in all lists that reach state q_j because then we would output a_m rather than a_j in state q_j.) We can thus replicate this transition in state q_i as well.

LEMMA 5: *For every two states q_i and q_j, it holds that $V_i \neq V_j$.*

PROOF:

Suppose $V_i = V_j$. Then $a_i = a_j$ by Lemma 4, and the output function in states q_i and q_j is identical. Suppose the automaton always processes an element $y \neq a_i$ before reaching state q_i. Then, $u(y) < u(a_i)$ or else we would output y in state q_i instead of a_i. In addition, there is no sequence of transitions connecting state q_i to a state q_m such that $f(q_m, x) = \text{argmax}\{u(y), u(x)\}$ because this would imply a transition from some state with a higher continuation value to another state with a lower continuation value (following Lemma 4), contradicting the optimality of $\mathcal{M}$. Thus, the only relevant history in state q_i is a_i. Similarly, the only relevant history in state q_j is $a_j = a_i$. Because the relevant history in both states is a_i, we can use arguments similar to Lemma 3 and replace one of the states with a new state q' such that the modified automaton achieves a higher expected utility.

By Lemma 5, we have that $V_i \neq V_j$ for every $0 < i \neq j \leq K-1$. This implies that $a_i \neq a_j$ because otherwise we would have $V_i = V_j$. Thus, by Lemmas 4 and 5, we can reorder the states such that $V_0 < V_1 \ldots < V_{K-1}$ and $u(a_0) < u(a_1) \ldots < u(a_{K-1})$. For every i we also have that $V_i > u(a_i)$ (or else we can define $g(q_i, x) = \textit{Stop}$ and obtain an improvement over $\mathcal{M}$).

To conclude the proof, I characterize the transition function of $\mathcal{M}$. Consider a transition $g(q_i, a_j)$ for $j > i$. Because $V_j > V_i$ and $V_j > u(a_j) > u(a_i)$, we cannot have that $g(q_i, a_j) \in \{q_i, \textit{Stop}\}$. Thus, $g(q_0, a_j) = q_j$. For $i \neq 0$, we cannot have that (i) $g(q_i, a_j) = q_0$ because $V_j > V_0$; or that (ii) $g(q_i, a_j) = q_m$ for $m > 0, m \neq i, j$, because this implies that $\mathcal{M}$ generates an infeasible choice from the list $(a_i, a_j, x_{\min})$. Thus, $g(q_i, a_j) = q_j$ for $j > i$. By similar arguments, if $j \leq i$ then $g(q_i, a_j) = q_i$.

Consider a transition $g(q_i, x)$ for $x \neq a_j$. We cannot have that $g(q_i, x) = q_m$ with $m > i$ because $\mathcal{M}$ then outputs an infeasible choice from $(a_i, x, x_{\min})$. We cannot have that $g(q_i, x) = q_m$ with $m < i$ because $V_i > V_m$. Thus, $g(q_i, x) \in \{q_i, \textit{Stop}\}$. By the optimality of $\mathcal{M}$, $g(q_i, x) = \textit{Stop}$ if $u(x) > V_i$, and $g(q_i, x) = q_i$ otherwise.

REFERENCES

Abreu, Dilip, and Ariel Rubinstein. 1988. "The Structure of Nash Equilibrium in Repeated Games with Finite Automata." *Econometrica*, 56(6): 1259–81.

Allport, Gordon W. 1954. *The Nature of Prejudice*. Reading, MA: Addison-Wesley.

Bandyopadhyay, Taradas. 1988. "Revealed Preference Theory, Ordering and the Axiom of Sequential Path Independence." *Review of Economic Studies*, 55(2): 343–51.

Bruine de Bruin, Wändi. 2005. "Save the Last Dance for Me: Unwanted Serial Position Effects in Jury Evaluations." *Acta Psychologica*, 118(3): 245–60.

Campbell, Donald E. 1978. "Realization of Choice Functions." *Econometrica*, 46(1): 171–80.

Copeland, Duncan G., Richard O. Mason, and James L. McKenney. 1995. "Sabre: The Development of Information-Based Competence and Execution of Information-Based Competition." *IEEE Annals of the History of Computing*, 17(3): 30–57.

Dovidio, John F., Peter Glick, and Laurie A. Rudman, ed. 2005. *On the Nature of Prejudice: Fifty Years After Allport*. Malden, MA: Blackwell Publishing.

Dow, James. 1991. "Search Decisions with Limited Memory." *Review of Economic Studies*, 58(1): 1–14.

Fryer, Roland, and Matthew O. Jackson. 2008. "A Categorical Model of Cognition and Biased Decision Making." *The B.E. Journal of Theoretical Economics*, 8(1). http://www.bepress.com/bejte/vol8/iss1/art6/.

Glejser, Herbert, and Bruno Heyndels. 2001. "Efficiency and Inefficiency in the Ranking in Competitions: The Case of the Queen Elisabeth Music Contest." *Journal of Cultural Economics*, 25(2): 109–29.

Ho, Daniel E., and Kosuke Imai. 2008. "Estimating Causal Effects of Ballot Order from a Randomized Natural Experiment: The California Alphabet Lottery, 1978–2002." *Public Opinion Quarterly*, 72(2): 216–40.

Hopcroft, John E., and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory: Languages and Computation.* Reading, MA: Addison-Wesley.

Iyengar, Sheena S., and Mark R. Lepper. 2000. "When Choice Is Demotivating: Can One Desire Too Much of a Good Thing?" *Journal of Personality and Social Psychology*, 79(6): 995–1006.

Kahneman, Daniel, Jack L. Knetsch, and Richard H. Thaler. 1991. "Anomalies: The Endowment Effect, Loss Aversion, and Status Quo Bias." *Journal of Economic Perspectives*, 5(1): 193–206.

Kalai, Ehud, and William Stanford. 1988. "Finite Rationality and Interpersonal Complexity in Repeated Games." *Econometrica*, 56(2): 397–410.

Kalai, Gil. 2003. "Learnability and Rationality of Choice." *Journal of Economic Theory*, 113(1): 104–17.

King, Amy, and Andrew Leigh. 2009. "Are Ballot Order Effects Heterogeneous?" *Social Science Quarterly*, 90(1): 71–87.

Koppell, Jonathan GS, and Jennifer A. Steen. 2004. "The Effects of Ballot Position on Election Outcomes." *Journal of Politics*, 66(1): 267–81.

Krosnick, Jon A., Joanne M. Miller, and Michael P. Tichy. 2004. "An Unrecognized Need for Ballot Reform: The Effects of Candidate Name Order on Election Outcomes." In *Rethinking the Vote: The Politics and Prospects of American Election Reform*, ed. Ann N. Crigler, Marion R. Just and Edward J. McCaffery, 51–73. New York: Oxford University Press.

Luce, R. Duncan, and Howard Raiffa. 1957. *Games and Decisions: Introduction and Critical Survey*. New York: Wiley.

Mandler, Michael. Forthcoming. "Rationality and the Speed of Decision-Making." *In Proceedings of the 12th Conference on Theoretical Aspects of Rationality and Knowledge*.

Manzini, Paola, and Marco Mariotti. Forthcoming. "Categorize then Choose: Boundedly Rational Choice Welfare." *Journal of the European Economic Association*.

Mealy, George H. 1955. "A Method for Synthesizing Sequential Circuits." *Bell Systems Technical Journal*, 34(5): 1045–79.

Meredith, Marc, and Yuval Salant. 2007. "The Causes and Consequences of Ballot Order-Effects." SIEPR Discussion Paper 06-029.

Mervis, Carolyn B., and Eleanor Rosch. 1981. "Categorization of Natural Objects." *Annual Review of Psychology*, 32: 89–115.

Miller, Jack E. 1980. *Menu Pricing and Strategy*. Boston, MA: CBI Publishing.

Miller, Joanne M., and Jon A. Krosnick. 1998. "The Impact of Candidate Name Order on Election Outcomes." *Public Opinion Quarterly*, 62(3): 291–330.

Neyman, Abraham. 1985. "Bounded Complexity Justifies Cooperation in the Finitely Repeated Prisoner's Dilemma." *Economics Letters*, 19(3): 227–29.

Rosch, Eleanor. 1978. "Principles of Categorization." In *Cognition and Categorization*, ed. Eleanor Rosch and Barbara B. Lloyd, 27–48 Hillsdale, NJ: Erlbaum.

Rubinstein, Ariel. 1986. "Finite Automata Play Repeated Prisoner's Dilemma." *Journal of Economic Theory*, 39(1): 83–96.

Rubinstein, Ariel. 1996. "Why Are Certain Properties of Binary Relations Relatively More Common in Natural Language?" *Econometrica*, 64(2): 343–55.

Rubinstein, Ariel, and Yuval Salant. 2006. "A Model of Choice from Lists." *Theoretical Economics*, 1(1): 3–17.

Sen, Amartya. 1993. "Internal Consistency of Choice." *Econometrica*, 61(3): 495–521.

Simon, Herbert A. 1955. "A Behavioral Model of Rational Choice." *Quarterly Journal of Economics*, 69(1): 99–118.

Simonson, Itamar, and Amos Tversky. 1992. "Choice in Context: Tradeoff Contrast and Extremeness Aversion." *Journal of Marketing Research*, 29(3): 281–95.

US General Accounting Office. 1986. *Airline Competition: Impact of Computerized Reservations Systems.* Resources, Community, and Economic Development Division. Washington, DC, May.

US Department of Justice. 1983. *Advanced Notice of Proposed Rulemaking – Airline Computer Reservation Systems*. EDR - 466, Docket No. 41686 (before the Civil Aeronautics Board).

Wilson, Andrea. 2002. "Bounded Memory and Biases in Information Processing." *NAJ Economics*, 5. http://www.najecon.org/naj/cache/234936000000000070.pdf.

Wilson, Vietta E. 1977. "Objectivity and Effect of Order of Appearance in Judging of Synchronized Swimming Meets." *Perceptual and Motor Skills*, 44(1): 295–98.