WORKING PAPER SERIES

What automaton model captures decision making? A call for finding a behavioral taxonomy of complexity

Stephan Schosser/Bodo Vogt

Working Paper No. 10/2015

OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

FACULTY OF ECONOMICS
AND MANAGEMENT

Impressum (§ 5 TMG)

Herausgeber:
Otto-von-Guericke-Universität Magdeburg
Fakultät für Wirtschaftswissenschaft
Der Dekan

Verantwortlich für diese Ausgabe:
Stephan Schosser/Bodo Vogt
Otto-von-Guericke-Universität Magdeburg
Fakultät für Wirtschaftswissenschaft
Postfach 4120
39016 Magdeburg
Germany

http://www.fww.ovgu.de/femm

Bezug über den Herausgeber
ISSN 1615-4274

What automaton model captures decision making? A call for finding a behavioral taxonomy of complexity

Stephan Schosser
Stephan.Schosser@ovgu.de

Bodo Vogt
Bodo.Vogt@ovgu.de

July 22, 2015

Abstract

When investigating bounded rationality, economists favor finite-state automatons – for example the Mealy machine – and state complexity as a model for human decision making over other concepts. Finite-state automatons are a machine model, which are especially suited for (repetitions of) decision problems with limited strategy sets. In this paper, we argue that finite-state automatons do not suffice to capture human decision making when it comes to problems with infinite strategy sets, such as choice rules. To proof our arguments, we apply the concept of Turing machines to choice rules and show that rational choice has minimal complexity if choices are rationalizable, while complexity of rational choice dramatically increases if choices are no longer rationalizable. We conclude that modeling human behavior using space and time complexity best captures human behavior and suggest to introduce a behavioral taxonomy of complexity describing adequate boundaries for human capabilities.

1 Introduction

Bounded rationality, namely players resorting to non-equilibrium paths due to mental limitations, can be explained by different automaton models. To do so, strategies are implemented using an automaton. A strategy is favored if the computational complexity of the corresponding automaton is low. Although this approach is intriguing, two central issues are not resolved: (1) Although most automaton models were developed to capture the human brain, neither of them does. (2) Not every decision problem is solvable by every automaton.

Early work on the computational complexity of decision problems focused on applying finite-state automatons to (finitely) repeated prisoner's dilemma games (Neyman, 1985; Rubinstein, 1986). Later, this work was extended to repetitions of two person normal form (Neyman, 1998) and extensive form (Piccione & Rubinstein, 1993; Chatterjee & Sabourian, 2000) games. In a recent paper (Salant, 2011), choice rules are analyzed with a focus on computational complexity.

Especially choice rules are different from the other decision problems: Repetitions of extensive form and normal form games, allow for a limited set of potential outcomes and strategies which are ex ante known. Here complexity is a result of the number of actions in previous periods a player has to memorize. For choice rules neither strategies, i.e. the choice to take, nor outcomes, i.e. the utility value of the choice, are limited and ex ante known. Here, according to (Simon, 1955, 1956), three aspects reduce the complexity of decision making: (1) Separation of means and ends: Best alternatives are found only if the goals and the measures taken to reach those goals are easily distinguishable. (2) (Low) size of the planning horizon: The smaller the set of alternatives to consider, the simpler the identification of the best alternative at hand. (3) Fixed aspiration levels: Changing expectations concerning the best alternative increases the complexity of the decision.

In this paper, we argue that finite-state automatons, such as Mealy machines (Mealy, 1955), are well suited for repeated games with limited sets of strategies, but fail to capture choice rules: Choice functions from lists (Rubinstein & Salant, 2006) clearly separate the means (the choices) and the ends (the payoff of the given choices) (1). The separation of means and ends cannot influence the complexity of finding an adequate choice from a list. At the same time, the size of the planning horizon (2), namely the number of elements in the list, is one aspect that influences complexity (Salant, 2011). However, in contrast to Salant (2011), we argue aspiration levels are another (3): Salant represents aspiration levels with a fixed satisfactory threshold a^*. Notice that fixed satisfactory thresholds are crucial when representing choice behavior using a Mealy machine: Mealy machines capture input and intermediate results using states. Each state represents a maximum of one possible satisfactory threshold. As each satisfactory threshold can result in only one of all infinitely many values, a corresponding Mealy machine has to consist of infinitely many states. Because, per definition, Mealy machines have only finitely many states (Mealy, 1955), Mealy machines cannot implement such behavior. Salant (2011) circumvents this problem by introducing a K-phase satisficer, i.e. a machine supporting K different, predefined satisfactory thresholds. The K-phase satisficer compares each choice from the

list to these K ex ante defined satisfactory thresholds and halts when it has found a satisfiable one.

We give an overview over automaton models which seem to be adequate for modeling human decision making. By using a simple counting example – a task that human beings are able to solve but a Mealy machine is not – we discuss the computational limitations of the Mealy machine. We also show that a Turing machine captures the idea that humans have a long-term memory and a short-term memory, which is not captured by the Mealy machine.

In a next step we discuss the concept of complexity for these machine models. Complexity consists of state complexity, time complexity and space complexity. On the one hand, state complexity measures the length of the strategy description, i.e. the quality of the strategy designer. On the other hand, space and time complexity measure the effort of realizing the behavior induced by the strategy. Therefore, in complexity discussions, state complexity is often not regarded as central, but space complexity and time complexity are. If we compare models of human decision making and the machine models, state complexity corresponds to limitations in long-term memory and space complexity to limitations in short-term memory. The fact that short-term rather than long-term memory limitations are regarded as important in human decision making supports our argument that space complexity is a central complexity concept.

In a last step, we show that Turing machines and space complexity capture the problem of (variable) aspiration levels as described by Simon (1956) in contrast to a Mealy machines and state complexity. We compare two central families of choice functions from lists, namely rational choice[1] and satisficing. By applying machine models most popular among computer scientists but different from the Mealy machine, namely the Turing machine (Turing, 1936), and complexity concepts other than state complexity, namely space and time complexity (Hartmanis & Stearns, 1965), we show the central results of Salant (2011) hold, even in the presence of variable aspiration levels, i.e. if aspiration levels are unknown ex ante. However, we show that introducing complexity limits much lower than the limits of existing complexity classes, is necessary to justify satisficing behavior.

We see our work as a motivation to combine two separate approaches from computer science and economics. While to former spent effort in sepa-

[1]In the literature rational choice behavior is often called optimizing or maximizing behavior. In the remainder of this paper, we focus on the term rational choice to simplify presentation.

rating tractable from intractable games with a focus on computerized agents (Tennenholtz, 2004) choosing strategies on behalf of humans, the latter modeled bounded rationality with respect to state complexity. However, neither of both approaches predicts human behavior. State complexity does not represent any mental limitations, and even several tractable problems are far to complex to be solved by human decision makers. Only a joint approach to precisely derive the limitations of human decision making and game theoretic extensions incorporating these limitations can help to model bounded rationality.

2 Overview over Automaton Models

Most theoretical work on bounded rationality considering (procedural) complexity of strategies has focused on either Moore machines (e.g., Abreu & Rubinstein, 1988; Neyman, 1985) or, more recently, on Mealy machines (e.g., Salant, 2011, see Gilboa & Zemel, 1989; Koller & Megiddo, 1992 for exceptions). Both Moore and Mealy machines are deterministic finite automatons. Aside from deterministic finite automatons, several other automaton models exist which differ in the descriptive power of the program (or language) they support. In this section we first discuss the motivation for introducing automaton models different from deterministic finite automatons following the hierarchy introduced by Chomsky (1956). When we discuss the advantages and disadvantages of different automaton models, we use the problem of counting which seems to be simple for a human being to realize.

All existing automaton models rely on the same basic concept. When processing information, the automaton starts in an initial state. Then the automaton receives an input in the form of a string. It processes this input sign by sign. Depending on the last sign read, it either changes its state or remains in its current state. The set of rules describing the transitions from one state to another depending on the input sign are called the program of the automaton (Hopcroft, 2007).

Chomsky (1956) classified the automaton models according to the program language they can process. The simplest automaton models, i.e. Chomsky Type-3, are deterministic finite automatons, like Mealy or Moore machines. They support evaluations of the input with respect to regular repetitions in the input string. For example, with respect to lists of different choices, they identify the first occurrence of an element with value 3 or the first element with value 3 after six elements with value 2.

A central disadvantage of deterministic finite automatons is that they

cannot compute solutions for minimal extensions of regular repetitions. Think of a list with repeated elements with value 3 and value 2. An automaton should identify the most frequent value. No finite deterministic automaton can solve this problem according to the pumping lemma (Jaffe, 1978): Consider a list starting with n elements of value 3 followed by $n+1$ elements of value 2. Formally, the list has length $2n+1$ and is represented by the following sequence $3^n 2^{n+1}$. A deterministic finite automaton to identify the most frequent value needs at least $n+1$ states to count the occurrence of value 3 and value 2. In state q_1 it remembers having seen one value 3, in state q_2 it remembers having seen two values 3, in state q_n it remembers having seen n values 3. If in state q_i a value 2 is read from the input, the automaton moves to q_{i-1}. If after reading all input elements the automaton is in a state different from q_0 more values 3 than 2 occurred. As soon as the input consists of one additional 3 in the beginning of the list and one additional 2 at the end of the list, the automaton needs another state resulting in $n+2$. However, one can always think of a list having one more 3 and one more 2, resulting in an automaton having infinitely many states to support comparisons of the number of occurrences. As a deterministic finite automaton may not have infinitely many states (Hopcroft, 2007), this easy decision rule cannot be realized using a deterministic finite automaton.

The main reason for the limitations of deterministic finite automatons is their lack of memory. The problem described could easily be solved using an automaton of Chomsky Type-2 with one (memory) stack, i.e. a pushdown automaton (Oettinger, 1961). In a stack one can save up to infinitely many elements and access only the last one added. Whenever a pushdown automaton for counting reads a 3 from the input, it adds it to the stack. If the automaton observes a 2, it removes the last added 3 from the stack. If the automaton reads a 2 from the input and the stack is empty, the input consists of more 2s than 3s. Although pushdown automatons support counting, they still have their limitations. Think of additional input elements with values different from 3 and 2. To find the most frequent one the automaton has to memorize the number of elements observed for each specific element. Hence, it would have to simultaneously access at least 2 of $n-1$ stacks, where n is the number of different values.

To increase the capability of the automaton, we have to allow the automaton to access more than one memory element. A corresponding automaton of Chomsky Type-1 is the linear-bounded non-deterministic Turing machine (Myhill, 1960). A linear-bounded non-deterministic Turing machine supports $n \cdot k$ memory slots, with n being the length of the input and $0 < k < \infty$ being a constant. These memory slots can be used to count the

Table 1: Supported comparisons per Chomsky Type

Chomsky Type	Automaton model	Access to memory	Unsupported Comparisons
3	Deterministic Finite Automaton (Mealy and Moore machine)	0 elements	Frequency of two different elements in lists Frequency of three different elements in list All elements in a list with each other
2	Non-deterministic Pushdown Automaton	1 element	Frequency of three different elements in list All elements in a list with each other
1	Linear-bounded non-deterministic Turing machine	$n \cdot k$ elements[2]	All elements in a list with each other
0	Deterministic Turing machine	Infinitely many elements	- (Computationally equivalent to computers)

occurrence of each element and identify the most frequent one.

Linear-bounded non-deterministic Turing machines still lack some of the functionality of modern computers. I.e., think of a machine that wants to compare all elements in a list with every other element, and create a preference order in which one element is better than another element if it lost fewer comparisons with worse elements. An automaton implementing such behavior would have to memorize the result of each individual comparison and determine an order depending on these results. Hence, it would need to memorize the result of $n \cdot n$ comparisons. As always, a n exists so that $n \cdot n > n \cdot k$ holds: this is not applicable to a linear-bounded non-deterministic Turing machine. Deterministic Turing machines without memory limits (Turing, 1936) can solve this problem. They have Type-0 according to the Chomsky hierarchy (Hopcroft, 1969). Table 1 summarizes our discussion.

3 Turing machines

We have given a short overview of different automaton models. The simplest ones, i.e. finite-state automatons, cannot compare the number of occurrences of two different values in a list of input elements. More sophisticated automatons, such as the non-deterministic pushdown automaton, still lack the ability to compare the occurrences of more than two different values. We have shown that choice rules with variable aspiration levels cannot be captured by finite-state automatons, but by Turing machines. We introduced two different variants of Turing machines, namely linear-bounded non-deterministic Turing machines and deterministic Turing machines, both of which support counting. Because deterministic Turing machines are as powerful as current computers, computer scientists today focus their com-

[2] n is the number of elements of the input or the length of the input. k is a constant with $k < \infty$.

plexity considerations on this automaton model. In the remainder of this paper, we analyze choices from lists using multi-tape Turing machines, a machine type computationally equivalent to deterministic Turing machines (Papadimitriou & Yannakakis, 1994). Therefore, we first formally introduce the multi-tape Turing machine in the remainder of this section formally, before we give an overview of the complexity concepts.

A (multi-tape) Turing machine processes data using n tapes consisting of infinitely many cells[3]: the input tape and $n-1$ output tapes. Initially the input tape, a read-only tape, holds the input, a finite-length string. All cells of the $n-1$ output tapes hold the symbol B. The input tape head reads the data on the input tape. The input tape head is initially positioned at the left-most position of the input. When processing a program the Turing machine moves along all n tapes. Every move consists of three steps: (1) The machine changes its state, where the new state might be the old state. (2) The Turing machine writes a symbol to any combination of output tapes at the position of the corresponding tape heads. This symbol replaces the symbol currently at the specified position and might be the same symbol as the one that was there before. (3) All tape heads move either one position to the left or one position to the right or remain in the current position.

Definition 1 *A Turing machine is a 6-tuple $M = (Q, \Gamma, \delta, q_0, B, F)$, where Q is the finite set of states, Γ is the set of tape symbols, $\delta : Q \times \Gamma^n \rightarrow Q \times [(\Gamma \times \{L, R, N\})]^n$ is a transition function, $q_0 \in Q$ is the start state, $B \in \Gamma$ is the blank symbol and $F \subseteq Q$ is the set of accepting states.*

In contrast to deterministic finite automatons, three complexity concepts – state complexity, space complexity and time complexity – are applicable to a Turing machine. The definition of state complexity ($state_M$) is equivalent to the definition of state complexity of a deterministic finite automaton. In addition, space complexity captures the number of positions on output tapes used during the processing of a program, while time complexity is the number of head moves used.

Definition 2 *We capture the complexity of a Turing machine M with three different concepts:*

a) $state_M$: number of states M consists of

[3]Notice, that we resort to multi-tape Turing machines as we believe them to be closest to the human brain: human decision makers can access different data chunks they memorized without having to move on their memory tape.

b) $time_M(w)$: number of head moves M uses when processing input w

c) $space_M(w)$: number of tape cells M uses when processing input w

Note that, while the number of states does not depend on the input, both the number of head moves and the number of tape cells can vary with the input. One typically measures both time and space complexity in a worst-case scenario. That is, one measures the maximum head moves or tape cells used when receiving the worst input for the algorithm of length n.

Definition 3 *(e.g., Goldreich, 2008): For a Turing machine M complexity is defined as*

a) $STATE_M$: $state_M$

b) $TIME_M(n)$: $max\{time_M(w)||w| = n\}$

c) $SPACE_M(n)$: $max\{space_M(w)||w| = n\}$

4 Chomsky Hierarchy and Automaton Models

Let us now apply the results from considering automaton models with different Chomsky Types to economic decision making (see Table 2). Notice, as the capabilities increase between the different Chomsky types every automaton model with a lower Chomsky Type can solve all problems a automaton with a higher type can solve. E.g., if an automaton of Chomsky Type-2 can solve a certain problem, all automata with Chomsyk Type-0 and Type-1 can do so, too.

Pure strategies in repeated versions of normal form and extensive form games can be easily implemented by automatons of any Chomsky type: The basic idea is that they assign a certain response to a (finite) history of previous actions. Hence, the corresponding automaton has to have (a maximum of) one state per possible history to consider and realize the response to this history. Given that the game is not played for infinitely many periods or players 'forget" part of the history after some time the history to remember is always finite and memorizable.

In contrast to equilibria in (repeated) games, for choices from lists, as the ones considered by Salant (Salant, 2011), simple automata with Chomsky Type-3 suffice to make a choice, if and only if outcomes are ex ante known and decision maker knows their specific order. We believe that this is not realistic. Think of a website presenting shoes characterized by different

Table 2: Chomsky Type and decision problems

Chomsky Type	Decision problem
3	Equilibria in pure strategies in (repeated) games
	Choice from list if outcomes are known ex ante and finite
2	Choice from list with memorizing position of best alternative
	Choice from list with variable aspiration levels
1	Market decisions based on decisions of the competition
0	Any decision possible

prices, colors and shapes. A decision maker will not ex ante know all possible shoes. Hence, for him the set of shoes is infinite and an automaton with Chomsky Type-3 cannot implement the corresponding choice rule. Notice, this even holds for K-phase satisficing: To specify the K thresholds the decision maker has to know the distribution of his utility values for the shoes, an unrealistic assumption. If he does not know the distribution, he could start with an automaton accepting only shoes with to high utility values and would have to restart after traversing all shoes without any above his lowest acceptance threshold. As the automaton of Chomsky Type-3 cannot memorize anything, after this traversing the distribution of utility values of shoes would still be unknown and the new automaton would have to be parameterized without any knowledge from the first traversal.

To make choices if outcomes are ex ante unknown Chomsky Type-2 suffices. Here, the player can memorize the utility value of the best choice seen to date using his memory and compare all subsequent choices to this so far best alternative. Given a utility function, an automaton of Type-2 can implement any choice from a list.

An automaton of Chomsky Type-1 or higher is necessary if properties of all alternatives to need be kept in mind. I.e., think of a market decision where one company wants to choose a price based on the prices and product qualities of the competition. Given that price and product qualities the output per competitor could be calculated and the own decision made.

Finally Type-0, further extends the capabilities. A corresponding automaton would be necessary, if the influence of prices and product quality of one company on another would be taken into account and should be measured.

5 Introducing Aspiration Levels

In the last section, we showed that for realistic choice decisions, i.e. choices with variable aspiration levels, automatons with at least Chomsky Type-2 are necessary. We now argue that applying Turing machines allow for better

capturing the human brain structure, before we describe a Turing machine to make choices from lists.

If we compare models of human decision making and empirical findings with the Turing machine time and space complexity better capture the difficulties in human decision making than state complexity. A simple comparison would look as follows. The states of a Turing machine correspond to human long-term memory. Human short-term memory is comparable to the space a Turing machine requires on the output tape when processing information. Finally, the time it takes for a human to process certain information is captured by the time a Turing machine requires to process information. Recent research by neurologists assigns a different importance to all concepts: Experiments with monkeys show that the more difficult a decision problem is, the slower the rate of increase in neural activity (Heekeren, 2008). Hence, the more difficult the problem, the more slowly a decision is made: Time complexity does matter for human information processing. While initial work on human short-term memory claims that humans are able to memorize up to 7 chunks[4] of information (Miller, 1956), more recent work sees the limit at about 4 chunks of information (Cowan, 2001): Space complexity does matter for human information processing. According to recent results of neurologists the size of the neural network is not really limited. Each human brain consists of 21.5 billion neurons (Pakkenberg & Gunderson, 1997). As this is an indicator for the capabilities of the human long-term memory, we believe that state complexity does not matter in human decision making.

5.1 Mealy machine

We now discuss different machines implementing choice rules. We first define choice rules before introducing corresponding machines. "A list is a [...] finite sequence of elements X," given that X is a finite set of N elements (Rubinstein & Salant, 2006). A rational "decision maker has a strict preference relation (i.e., complete, asymmetric and transitive) $\succ$ over X and chooses the $\succ$-best element from every list. (Salant, 2011)" In contrast, a

[4] A "chunk of information is rather fuzzy and culture dependent: A chunk of information is one data item a human being can memorize in isolation. Think for example of the string "NYTWSJNFLNBA. Although the string is a combination of 12 characters, it can be seen as 4 chunks: "NYT (New York Times), "WSJ (Wall Street Journal), "NFL (National Football League), "NBA (National Basketball Association), at least for the average American. For a Chinese farmer, all these abbreviations can have no meaning, which results in his being presented with 12 symbols not in the characters he is familiar with. He would have to memorize each letter in isolation, or 12 chunks.

satisficing "decision maker has a strict preference relation $\succ$ over X and a satisfactory threshold $a^* \in X$. He chooses the first element in the list that is not inferior to a^*; if there is no such element he chooses the last element in the list."

As an example, we use an outcome space with four alternatives, $X = \{1, 2, 3, 4\}$, introduced by Salant (2011)[5]. Salant suggests an automaton without support for variable aspiration levels (see Figure 1 (a)). Per definition, satisfactory choices are 3 and 4, while 1 and 2 are unsatisfactory. The automaton consists of one state. As long as 1 or 2 are read from the input, the automaton stays in the unsatisfied state q_0 and reads the next element from the list. As soon as the automaton reads 3 or 4, the automaton halts and writes the found number as output. In this subsection, we prove that the simplicity of the automaton is mainly due to the lack of support for variable aspirations levels.

Proposition 1 *The state complexity of Mealy-automaton supporting variable aspiration levels is at least* $N - 1$.

Proof 1 *When deriving aspiration levels an automaton ex ante does not know the acceptable aspiration level. Hence, an automaton could accept every one of the* $N-1$ *values between the minimal possible and the maximal possible value as aspiration level. A corresponding automaton has to consist of at least* $N-1$ *states, as for each aspiration level a separate state needs to be part of the machine which determines whether to stop searching, to remain in the aspiration level or to switch to another aspiration level. Figure 1 (b) shows a simple example of an automaton with support for variable aspiration levels. The machine starts in state* q_0 *which only accepts the maximum (i.e.,* 4*) as a solution. After reading the maximum the machine switches to the final state and halts. If it reads* 1 *or* 2*, it switches to a state* q_1 *which accepts* 2 *as solution, while it switches to a state* q_2*, which accepts* 3 *otherwise. In subsequent stages,* q_1 *or* q_2*, the automaton switches to the final state and halts if either the maximum or the aspiration level is reached, while it keeps processing the input if the aspiration level is not reached.*

[5]Note that the state in which the machine halts, represented by the symbol "Stop, according to Mealy (1955) also is a state, called the final state. To adhere as close to the terminology of Salant (2011) as possible, we do not count final states when deriving state complexity. In Mealy machines the output is typically specified on the edges. Salant visualizes output by transition functions for outgoing edges below states. To simplify comparison, in this paper follow the notation of Salants notation.

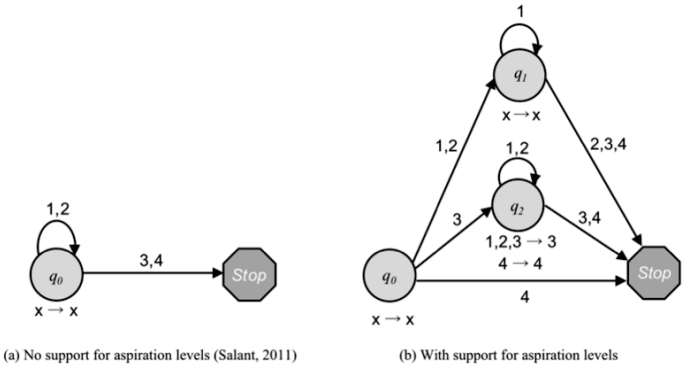

Figure 1: Different Mealy machines

Theorem 1 *A Mealy-automaton cannot support variable aspiration levels given the set of elements X is not known or infinitely large.*

Proof 2 *The theorem directly follows from Proposition 1. If the set of elements X is not known an infinite number of potential aspiration levels has to be considered. Hence $|X|$ is infinite. If $|X| = N - 1$ is finite, a corresponding Mealy-automaton would have infinitely many states which contradicts the definition of the automaton.*

5.2 Turing machine

Having shown, that Mealy machines are not capable of solving the problem of making a choice from a list given variable aspiration levels, we now apply Turing machines both to satisficing and rationalizing behavior and show that the theorems introduced by Salant 2011 still hold. Therefore, we prove the two theorems introduced by Salant (2011) for our setting, i.e. for a Turing machine, and derive explicit results for the complexity of the Turing machine.

Proposition 2 *For a Turing machine implementing order-independent and rationalizable choice functions, the following complexity results hold*

a) $STATE_M$*:* 1

b) $TIME_M(n)$*:* n

c) $SPACE_M(n)$*:* 1

Proof 3 *A Turing machine implementing rationalizable choice functions has to consist of one initial state. In this initial state, it reads the input and compares it with the first position of the output tape. If the input is larger than the element on the output tape or the element on the output tape is the blank symbol B, the Turing machine writes the input to the output type; otherwise it writes the element of the output to the output tape again. The Turing machine stays in the same state and moves the head on the input tape one position to the left. The Turing machine repeats this until (after n steps) it reaches the end of the input. The element on the output tape is the solution of the maximizing problem. While processing the input, the automaton never changes the state resulting in $STATE_M = 1$, it only writes one cell of the output tape resulting in $SPACE_M(n) = 1$. The Turing machine changes its state once for every sign of the input resulting in $TIME_M(n) = n$.*

Proposition 3 *For a Turing machine implementing order-independent and choice functions, which are not rationalizable, the automaton has the following complexity.*

a) $STATE_M$*:* 2

b) $TIME_M(n)$*:* $\frac{(n-1)(n-2)}{2}$

c) $SPACE_M(n)$*:* n

Proof 4 *A Turing machine implementing choice functions that are not rationalizable memorizes not just the best solutions, but rather all solutions that are not dominated by any other solution. In the worst case no element dominates any other element. Therefore, the Turing machine has to memorize all elements in the list on the output tape resulting in $SPACE_M(n) = n$. In the worst case, the automaton has to compare each element to all proceeding elements for this check. Hence, the time complexity of this step is $TIME_M(n) = \sum_{i=1}^{n} n - 1 = \frac{(n-1)(n-2)}{2}$. A corresponding Turing machine consists of at least one state for reading the input and one state for traversing through the output tapes and comparing the currently read element to all preceeding elements having a state complexity of $STATE_M = 2$.*

From Proposition 2 and Proposition 3 follows Theorem 2:

Theorem 2 *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.*

We conclude that Theorem 2 also holds, if complexity is modeled using a Turing machine. In particular, it holds for all complexity concepts, namely state, space and time complexity.

Theorem 3 *Any choice function with state complexity K that maximizes expected utility can be represented by a K-phase satisficing procedure.*

Proof 5 *Think of the deterministic finite automaton introduced by Salant (2011) with $STATE_M = K$. We turn this automaton into a Turing machine by ex ante defining K different fixed aspiration levels. We represent each aspiration level in a separate state of the Turing machine. The Turing machine always writes the best alternative found to date onto the first cell of the output tape resulting in $SPACE_M(n) = 1$. In the worst-case scenario, when the automaton finds no satisfiable solution, it reads all input elements $TIME_M(n) = n$. Note that, by applying Turing machines in contrast to finite-state automatons, the complexity of a satisficing procedure does not increase if aspiration levels turn from fixed to variable: In a Turing machine, the aspiration level can be saved on one of the output tapes. And for every modified aspiration level, the old aspiration level is overwritten, ensuring that the space complexity does not increase due to variable aspiration levels. At the same time the Turing machine needs no additional state, since the automaton always compares the current value to the value on the output tape. Finally, time complexity will not change as the automaton including all states and state transitions remains the same.*

6 Discussion

We argued that finite state automatons are not suitable to describe choice rules allowing for variable aspiration levels. By applying automaton models with lower Chomsky-Type, i.e., Turing machines, we showed that rational choice and satisficing are equally complex. While the former results, casts doubts on the quality of state complexity to analyze choice rules, the later result – at first sight – questions the use of time and space complexity. In this section, we sketch how to capture bounded rationality using a novel approach to space and time complexity

Computer science traditionally distinguishes between (1) unsolvable, (2) intractable and (3) tractable problems. Basically, problems are unsolvable if no algorithm to solve them exists, while intractable problems are problems that take to much time to be solved. Finally computers can solve tractable problems in a reasonable amount of time. To be more precise, problems

are tractable, if an algorithm exists which (in the worst case) solves the problem in polynomial time. I.e., the time complexity of the algorithm can be described by a function growing in n at the speed of a polynomial function or slower.

In this sense, space and time complexity can be used to classify problems to be either intractable or tractable. To mention just a few, Gilboa (1989) and Roughgarden (2009) showed that no tractable algorithm exists to derive Nash equilibria in which all players receive a payoff at a certain height. Koller (1992) found that for extensive form games with imperfect recall computing max-min behavior is not tractable. Ben-Porath (1990) proofed that finding the best response automaton is intractable, given uncertainty concerning the automaton other players are using. While these (and similar) results are important for agent design, i.e., they can help to understand whether autonomous agents acting on behalf of humans can find certain outcomes or not, they do not help to understand bounded rationality.

Think of the Turing machine implementing choice functions again. When using a Turing machine to implement order-independent and rationalizable choice functions from lists, the complexity of rational choice is minimal. This directly follows from Proposition 2. For every choice function (except for such trivial ones as choosing the first element) a worst case exists in which the automaton has to read all elements to find a solution resulting in $TIME_M(n) = n$. Hence, for rational choice an algorithm can be found which is tractable, while since Simon (Simon, 1955) economists accept that human decision makers satisfice due to bounded rationality.

Nevertheless, we believe that Turing machines (and space/time complexity) capture human behavior quite well: (1) Decision makers have to memorize observed behavior and/or other properties of the problem at hand. Hence, their memory limitations of approximately 4 chunks (Cowan, 2001) strongly influence what problem they can solve and what they cannot. (2) The time of each decision maker is limited hence, he will limit the time to process any problem at hand. Think of the choice from lists again. A typical decision maker will stop searching for better solutions after seeing $N << n$ elements of the list. I.e., the decision maker will, instead of rational choice, turn to some form of satisficing behavior which takes the best of the first N alternatives. Hence, it is time complexity which bounds his rationality. However, with much lower upper bounds than computers have.

To benefit from existing complexity considerations and the increasing number of researchers working on corresponding topics, it is crucial to better understand mental limitations (like space and time constraints) on the one hand and the algorithms – in the sense of the adaptive toolbox (Marewski

et al., 2009) – people implement on the other. With this understanding, we can derive the complexity of heuristics and show for which problems the heuristics are applicable and for which they are not by deriving the exact space and time complexity of a problem instead of deriving an upper bound for infrequent special cases. As Kearns (2012) suggested after analyzing human behavior in well-known algorithmic problems, we believe after formally investigating choice rules that bounded rationality in well-known game-theoretic applications calls for a behavioral taxonomy of computational difficulty. Only such a taxonomy can help us to understand bounded rationality.

7 References

References

Abreu, Dilip, and Rubinstein, Ariel. 1988. "The Structure of Nash Equilibrium in Repeated Games with Finite Automata" In Econometrica: Journal of the Econometric Society. 56(6), 1259-1281.

Axtell, Robert. 2005. "The Complexity of Exchange" In The Economic Journal. 115. 193-210.

Ben-Porath, Elchanan. 1990. "The Complexity of Computing a Best Response Automaton in Repeated Games with Mixed Strategies". In Games and Economic Behavior. 2 (1), 1-12.

Chatterjee, Kalyan, and Hamid Sabourian. 2000. "Multiperson Bargaining and Strategic Complexity." Econometrica 68 (6): 14911509.

Chomsky, Noam. 1956. "Three models for the description of language" In IRE Transactions on Information Theory. 2(3), 113-124.

Cowan, Nelson. 2001. "The magical number 4 in short-term memory: a reconsideration of mental storage capacity." In The Behavioral and brain sciences. 24(1), 87 114.

Gilboa, Itzhak, and Zemel, Eitan. 1989, "Nash and correlated equilibria: Some complexity considerations" In Games and Economic Behavior. 1(1), 80-93.

Gilboa, Itzhak. 1988. "The Complexity of Computing Automata in Repeated Best-Response Games" In Journal of Economic Theory. 45 (2), 342–352.

Goldreich, Oded. 2008, "Computational Complexity: A Conceptual Perspective," Cambridge University Press, Cambridge.

Hartmanis, Juris, and Stearns, Richard E. 1965. "On the computational complexity of algorithms" In Transactions of the American Mathematical Society. 117, 28530.

Heekeren, Hauke R, Sean Marrett, and Leslie G Ungerleider. 2008. "The neural systems that mediate human perceptual decision making." In Nature reviews. Neuroscience, 9(6): 46779.

Hopcroft, John E., Motwani, Rajeev, and Ullman, Jeffrey D. 2007. "Introduction to Automata Theory, Languages, and Computation," Addison Wesley, Boston/San Francisco/New York.

Hopcroft John E., Ullman Jeffrey D. 1969. "Formal languages and their relation to automata," Addison Wesley, Boston/San Francisco/New York.

Jaffe, Jeffrey. 1978. "A necessary and sufficient pumping lemma for regular languages" In ACM SIGACT News, 10(2), 48-49.

Kearns, Michael. 2012. "Experiments in social computation" In Communications of the ACM, 55 (10), 56-67.

Koller, Daphne, and Megiddo, Nimrod. 1992. "The complexity of two-person zero-sum games in extensive form" In Games and Economic Behavior, 4(4), 528-552.

Marewski, Julian N.; Gaissmaier, Wolfgang and Gigerenzer, Gerd. 2009. "Good judgments do not require complex cognition" In Cognitive Processing. 11 (2), 103-121.

Mealy, George H. 1955. "A method for synthesizing sequential circuits" In Bell System Technical Journal, 34(5), 1045-1079.

Miller, George A. 1956. "The magical number seven, plus or minus two: some limits on our capacity for processing information" In Psychological Review. 63, 81 97.

Moore, Edward F. 1956. "Gedanken experiments on Sequential Machines" In Automata Studies, Annals of Mathematical Studies, 129153, Princeton University Press.

Myhill, John. 1960. "Linearly Bounded Automata", WADD Technical Note 60-165, Wright-Patterson Air Force Base, Ohio.

Neyman Abraham. 1985. “Bounded complexity justifies cooperation in the finitely repeated prisoners dilemma” In Economics Letters. 19, 227-229.

Neyman, Abraham. 1998. “Finitely Repeated Games with Finite Automata” In Mathematics of Operations Research. 23 (3), 513552.

Oettinger, Anthony G. 1961. “Automatic syntactic analysis and the push-down store” In Proceedings of the Symposia in Applied Mathematics. 12.

Pakkenberg, Bente & Gundersen, Hans J. 1997. “Neocortical neuron number in humans: effect of sex and age” In The Journal of comparative neurology. 384(2), 31220.

Papadimitriou, Christos H., and Mihalis Yannakakis. 1994. “On Complexity as Bounded Rationality (Extended Abstract)” In, 726733. ACM.

Piccione, Michele, and Ariel Rubinstein. 1993. “Finite Automata Play a Repeated Extensive Game” In Journal of Economic Theory. 61,160168.

Roughgarden, Tim. 2009. “Computing equilibria: a computational complexity perspective” In Economic Theory. 42 (1), 193-236

Rubinstein, Ariel. 1986. “Finite Automata Play the Repeated Prisoner’s Dilemma” Journal of Economic Theory 39 (1) (June 1): 8396.

Rubinstein, Ariel, and Salant, Yuval. 2006. “A model of choice from lists” In Theoretical Economics. 1, 3-17.

Salant, Yuval. 2011. “Procedural analysis of choice rules with applications to bounded rationality” In American Economic Review. 101, 724-748.

Simon, Herbert A. 1955. “A behavioral model of rational choice” In The quarterly journal of economics. 69(1), 99-118.

Simon, Herbert A. 1956. “Rational choice and the structure of the environment” In Psychological Review. 63, 129-138.

Tennenholtz, Moshe. 2004. “Program Equilibrium” In Games and Economic Behavior. 49 (2), 363–373.

Turing, Alan. 1936. “On Computable Numbers, With an Application to the Entscheidungsproblem” In Proceedings of the London Mathematical Society. 42(2).

Velupillai, K Vela. 2010.“Foundations of Boundedly Rational Choice and Satisficing Decisions” In Advances in Decision Sciences. 116.

Otto von Guericke University Magdeburg
Faculty of Economics and Management
P.O. Box 4120 | 39016 Magdeburg | Germany

Tel.: +49 (0) 3 91 / 67-1 85 84
Fax: +49 (0) 3 91 / 67-1 21 20

www.fww.ovgu.de/femm

ISSN 1615-4274