What Makes a Rule Complex? *

Ryan Oprea†

July 11, 2020

Abstract

We study the complexity of rules by paying experimental subjects to implement a series of algorithms and then eliciting their willingness-to-pay to avoid implementing them again in the future. The design allows us to examine hypotheses from the theoretical "automata" literature about the characteristics of rules that generate complexity costs. We find substantial aversion to complexity and a number of regularities in the characteristics of rules that make them complex and costly for subjects. Experience with a rule, the way a rule is represented, and the context in which a rule is implemented (mentally versus physically) also influence complexity.

Keywords: Complexity, rules, procedural decision making, bounded rationality, behavioral economics, economics experiments

JEL codes: C91, D91 G0, K4

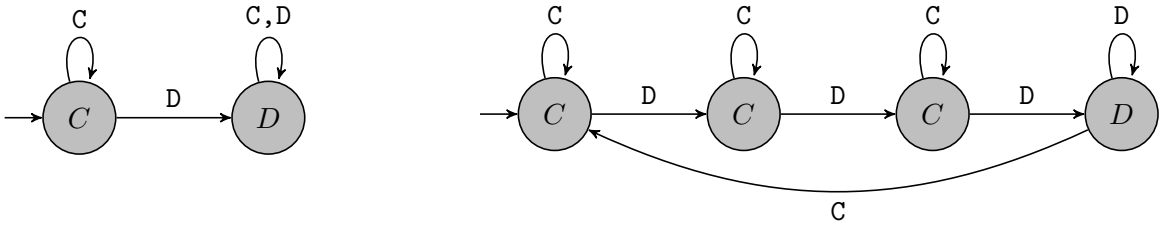
*I thank Youssef Benzharti, Peter Boessaerts, Andy Brownback, Mark Dean, Akshay Dixit, Guillaume Frechette, David Freeman, Tamsin German, Daniel Lokshtanov, Kirby Nielsen, John Rehbeck, Collin Raymond, Julian Romero, Yaroslav Rosokha, Hamid Sabourian, Adam Sanjurjo, Charles Sprenger, Emanuel Vespa and Sevgi Yuksel for their detailed comments and discussions. I also thank Lucas Coffman, Ben Enke, Yoram Halevy, PJ Healy, Daniel Martin and Dale Stahl for valuable conversations about this research. I am, also, grateful to seminar audiences at the University of Arkansas, the University of California, Davis, the University of Melbourne, Purdue University, the University of California, Santa Barbara, the University of Texas, Austin, the University of California, Berkeley, the University of California, San Diego and audiences at the Stanford Institute for Theoretical Economics, the 2019 Society for the Advancement of Economic Theory meetings in Ischia, the AFTS Workshop at University of Adelaide and the 2019 Economic Science Association North American Meetings in Los Angeles. This research was supported by the National Science Foundation under Grant SES-1949366.

†Economics Department, University of California, Santa Barbara. *Email: roprea@gmail.com*

1 Introduction

When people fail to behave as economists expect them to, they tend to err on the side of simplicity.[1] A long tradition in economics (e.g. Simon 1955) argues that this is, in part, driven by the fact that people dislike (or are incapable of) implementing complex rules and procedures.[2] In this paper we propose new experimental methods to identify what makes a rule complex to implement, and to measure the costs of this "procedural complexity." Doing this may allow us to better understand why people don't always choose to use optimal procedures (e.g. strategies, dynamic optimization policies, state-contingent agreements, belief updating rules, choice procedures) or perfectly comply with rules handed down by governments and organizations (e.g. regulations, laws, bureaucratic rules).

Our investigation is built around the idea that rules are in fact *algorithms*, implemented not by computers but by human actors and thinkers. The "automata literature " (surveyed in Kalai (1990), Chatterjee & Sabourian (2009)) formalizes this idea by mathematically describing rules as "automata," formal descriptions of algorithms from computer science. This literature advances several hypotheses (reviewed in Section 2) about what structural characteristics of automata make the rules they describe *complex*, operationalizing "complexity" as the *cost* of implementing a rule. We use these hypotheses – and the literature's operationalization of complexity as cost – to organize our investigation.



(a) Grim (*2S-1T*) (b) Tit-for-3-Tat (*countable*)

Figure 1: Automata representations of two rules (strategies in the repeated prisoner's dilemma). In parentheses are names for the equivalent automata used in the experimental design.

Figure 1 shows examples of automata from the repeated games literature, the first place this idea was applied in economics. Panel (a) shows the automaton description of a rule called *Grim* (the "grim trigger strategy"), a *strategy* for playing the repeated prisoner's dilemma.[3] The automaton consists of two *states* (shown as circles) and a set of *transitions* (shown as arcs). Letters inside the circles tell the player what to do in each state (e.g. C for "choose to cooperate", D for "choose to defect") and letters next to each arc tell the player when to follow the corresponding transition

[1] For instance, many key tendencies from behavioral economics seem to involve simplification: satisficing simplifies rational choice by radically reducing the number of rank comparisons made; myopia simplifies dynamic optimization by ignoring the impact of current acts on future outcomes; representativeness simplifies Bayes' rule by ignoring the influence of base rates on posteriors; narrow bracketing simplifies joint optimization by ignoring the relationships between choices; adaptive learning simplifies rational expectations by ignoring counterfactuals; the winner's curse and related errors simplify Bayesian Nash Equilibrium by ignoring contingencies or information in others' choices etc.

[2] This includes not only aversion to engaging in complex behavior (e.g. using complex strategies, following complex regulations) but also aversion to implementing the complex evaluative procedures needed to identify optimal choices (or decision rules) in *computationally complex* decision problems.

[3] For expositional ease, these automata include only transitions that depend on other players' actions – to completely describe a strategy, transitions must also depend on the player's own actions (see Kalai & Stanford 1988).

(e.g. when your counterpart chooses C, D). The rule instructs the player to begin by choosing C (cooperate), but then to transition to the second state and choose D (defect) as soon as the other player chooses D. The right state is *absorbing*, meaning no transitions lead away from it, so once here the player chooses D forever. Panel (b) of Figure 1 shows the automaton description of another strategy discussed in the literature called "Tit-for-3-tat," (Tf3T), a lenient version of the famous tit-for-tat strategy.[4] This rule instructs the player to cooperate until her counterpart has defected 3 times and then to switch to defect, forgiving her afterwards by starting the rule over (shown as an arc transitioning back to the first state) if she cooperates.

Intuitively Tf3T seems more complex to implement than Grim, but in what sense is this true? What is it about a rule like Tf3T that makes it "more complex" than a rule like Grim, and how is this complexity formally reflected in its algorithmic structure?

To answer questions like this, we conduct an experiment (described in Section 3) in which we assign subjects a sequence of abstract rules of the form e.g. "Choose x until you see a, after which switch to y" and show them a sequence of randomly generated events (letters in the alphabet).[5] The subject's task is simply to correctly *implement* the rule in response to the random events by typing a sequence of letters on her keyboard. After subjects have implemented a number of different rules, we elicit their *willingness to pay* to avoid having to implement each rule again in the future, allowing us to measure the subjective cost of implementing each rule. By varying characteristics of the automata describing these rules, we can measure what algorithmic characteristics cause a rule to be complex and costly.

For instance, one popular theory from the automata literature is *state complexity* (or s-complexity), the hypothesis that states generate complexity costs. States describe the degree to which a rule-follower must condition her behavior on the past and they also summarize the distinct ways a rule requires a decision maker to process and respond to information. In our example, Tf3T contains two more states than Grim and by comparing how much subjects are willing to pay to avoid Tf3T versus Grim we can measure the subjective cost of adding two states to a rule, allowing us to evaluate s-complexity. In the experiment we find that subjects are willing to pay *twice as much* to avoid Tf3T as Grim, providing some evidence of s-complexity. (Other natural complexity measures like implementation time are also generated by our data and we can perform the same exercise on them – for instance it takes subjects three times as long to implement Tf3T as Grim.)

Of course many algorithmic characteristics (not only states) differ between Tf3T and Grim and so the experiment is designed around a set of carefully chosen rules that independently vary automata characteristics to study their effects. For instance, Tf3T also contains many more *transitions* than Grim, and *transition complexity* (*t-complexity*, e.g. Banks & Sundaram (1990)) is the theory that this generates complexity in a rule. Transitions summarize the intensity with which a rule requires a decision maker to *monitor* and respond to the environment, summarizing a different aspect of a rule than states. Independently varying states and transitions and comparing behavior across our rules, we find that adding a transition indeed adds complexity costs, but that the effect is half as large as that of adding a state. Grim also has a simpler *architecture* than Tf3T because it contains an absorbing state and thus never requires the rule-follower to return to previously visited states. We find evidence that the architecture of rules shapes complexity too: ceteris paribus, rules with absorbing states are considerably less complex. Overall, our cost estimates suggest that rules like

[4]Normally Tf3T requires the player to wait for three contiguous rounds of defection before defecting. For ease of exposition we show a cumulative version that doesn't require the three rounds of defections to be contiguous.

[5]By studying abstract rules, we remove subjects' ability to make use of idiosyncratically familiar heuristics and analogies from daily life that may influence complexity in difficult-to-measure ways.

Tf3T are more complex (costly) than rules like Grim and we therefore might expect them to be used less frequently. As we discuss in Section 5, this is in fact true in experimental repeated prisoner's dilemmas.

Our experiment also allows us to study how people *represent* rules to themselves and how this influences complexity. For instance, the literature generally assumes efficient representation: people don't over-complicate plans of action by representing them as rules containing unnecessary states.We test this by giving subjects similar rules that do and do not contain redundant states and we find strong evidence against efficient representation: subjects do not treat rules that contain redundant states as any less complex. A related but more fundamental assumption is that people only represent behavior using rules describable by *finite state machines* (FSMs), the basic class of algorithms studied by the automata literature. Tf3T, for example, cannot be represented as an FSM with fewer than four states, but if represented as a slightly richer type of algorithm (a "pushdown automata") that can simply *count* the number of 'D' events (by holding and manipulating a simple object in working memory), it can be represented with only two states and thus be dramatically simplified. When we compare responses to rules like Tf3T to rules that cannot be represented as counting rules, we find that indeed Tf3T is far less complex (less costly, faster to implement), suggesting that subjects are not constrained to represent behavioral patterns as FSM-like rules. Our results thus suggest that richer descriptions of algorithms than are typically studied in the automata literature are likely important for fully understanding and characterizing procedural complexity.

Finally, our experimental task is abstract but we take some first steps towards understanding how *context* influences the complexity of rules. For instance, we show that familiarity with a rule has a powerful alleviative effect on complexity: subjects given repeated exposure to a rule learn to implement it much faster and grow less willing to pay to avoid the rule. Likewise, a treatment variation in our design reveals that it is significantly more complex (from both a time and cost perspective) for subjects to enact rules mentally (e.g. to think through a counterfactual or forecast another's behavior) than it is to implement a procedure via a sequence of acts (e.g. to enact a strategy in a repeated game).

The purpose of our experiment is not to "test automata models" or even to propose automata as models of human cognition. Rather, automata are formal ways of describing and taxonomizing *rules*, and the purpose of our experiment is to study what (if any) components of these formal descriptions (e.g. states, transitions) systematically translate into complexity costs for decision makers, given the rules they describe. To the degree algorithmic characteristics predict complexity costs, we may be able to use automaton descriptions of rules, procedures, strategies etc. (combined with estimates using methods like ours) to model, predict and explain behavior in applications (e.g. to understand what sorts of rules people are unlikely to use or follow).

Overall, our findings (described in detail in Section 4) can be summarized as follows:

1. Most subjects find implementing complex rules costly, but these costs are highly heterogeneous across subjects and linked to independent measures of cognitive ability.

2. Algorithmic characteristics of rules significantly influence their complexity. Rules that require more **states** are significantly more complex (more costly, take longer to implement). Adding a **transition** is also costly, but adds half as much cost.

3. "Architectural" characteristics of algorithms other than simple state and transition counts influence complexity too: rules that eventually terminate in **absorbing** states (rather than

endlessly doubling back to previously visited states) are signficantly less complex (equivalent to removing a state from the rule).

4. Subjects have difficulty efficiently representing rules to themselves by discarding **redundant states**. However, subjects are able to represent some rules to themselves in efficient ways not achievable using the simplest types of algorithms usually studied in the literature. For instance, subjects seem able to conserve complexity by tracking simple **sequences in working memory**, allowing them to remove costly states from rules.

5. Complexity costs fall significantly as subjects become experienced at (familiar with) a rule, but this "**procedural learning**" is fragile and not easily transferred to even superficial perturbations of the learned rule.

6. The context in which a rule is implemented can matter a great deal for complexity. Employing a rule **mentally** (e.g. to make inferences) is significantly more costly than employing a rule physically (e.g. to implement a strategy).

7. The features of rules that generate cost are closely related to the features that generate implementation time, another important complexity metric. By contrast, there is only a very weak relationship between the drivers of costs and of implementation mistakes.

Although our motivating examples are strategies, we emphasize that *any* decision procedure or rule has algorithmic structure and so the methods and measurements we study here have potential applications across economics. Perhaps most intriguingly, aversion to procedural complexity has been used to explain (via automata models) important findings in *behavioral economics* including satisficing, primacy and recency effects, choice overload and status quo bias (Salant 2011), stochastic choice (Kalai & Solan 2003), non-Bayesian inference (Chauvin 2020), biases in information processing (Wilson 2014), and failures of backwards induction (Neyman 1985).[6] Understanding the degree to which distaste for complexity underlies and ties together classic findings from behavioral economics is a particularly important application for our methods. Earlier, automata were used to understand how aversion to complexity might solve equilibrium selection problems in games (e.g. Rubinstein 1986, Abreu & Rubinstein 1988, Kalai & Stanford 1988), another promising application.[7] A third potentially valuable application is to use methods like ours to study the algorithmic complexity of regulations, bureaucracies, laws, tax codes etc. in order to understand why policies and institutions don't always work as intended and to guide the design of more effective alternatives.[8]

Though our focus is on understanding what makes *behavior* complex ("implementation complexity"), our methods may also be useful for understanding what makes some *decision problems* complex ("computational complexity"). This is because the *computational complexity* of a problem is typically operationalized (e.g. in computer science) as the *implementation complexity* of the most efficient evaluative procedure capable of solving it.[9] In some cases, optimal decision making may be complex because it requires decision makers to use elaborate solution procedures that are

[6] Recent experimental evidence suggests that subjects prefer to evaluate risk and lotteries procedurally (Halevy & Mayraz 2020, Nielsen & Rehbeck 2020) meaning procedural complexity may influence risky choice as well.

[7] Automata models of complexity have also been used in applied strategic contexts ranging from bargaining (e.g. Chatterjee & Sabourian 2000) and industrial organization (e.g. Fershtman & Kalai 1993, Sherstyuk 2011) to general equilibrium (e.g. Sabourian 2004, Gale & Sabourian 2005).

[8] See Abeler & Jäger (2015) for a related experiment on the complexity of tax incentives.

[9] The complexity of individual algorithms is measured in a branch of computer science called "analysis of algorithms" and is a direct analogue to what we call "implementation complexity." These complexity measures are ultimately used in "computational complexity theory" in computer science to define and evaluate computational complexity.

complex to implement (costly, time-intensive, difficult) in a similar way and for similar reasons that other rules (such as those in our experiment) are.[10,11] For example, Salant (2011) shows in an automata model that the computational problem of selecting an optimal option from a list is implementation-complex in exactly the sense we measure here. Several recent experimental papers (e.g. Murawski & Bossaerts 2016) have studied how the computational complexity of optimization problems impacts subjects' performance, and studying the degree to which implementation complexity measures like ours can be used to understand the effects of computational complexity on choice seems an important task for future research.[12]

Finally, we operationalize complexity *behaviorally* as a measure of the subjective burden a rule places on a decision maker. In doing this we follow several adjacent literatures. Computer science similarly defines complexity as a measure of the resources a machine must expend in order to implement an algorithm – usually time ("time complexity") or working memory ("space complexity")). By focusing on the subjective costs of rules ("cost complexity"), we also follow growing literatures in both cognitive psychology[13] and economics[14] that interpret behavior in terms of cognitive costs. A related interdisciplinary psychology literature studying the complexity of (e.g. workplace) tasks (e.g. Campbell 1988), makes a useful distinction between "objective complexity" (defined purely by characteristics of a task) and "experienced complexity" (determined also by the abilities, tastes and

[10]See Bossaerts & Murawski (2017) for a discussion of computational complexity and human behavior, emphasizing that many even basic decision problems in economics can be intractably computationally complex. Echenique et al. (2011) point out that, even though individual optimization problems are computationally complex, satisfying rationality in the revealed preference sense is not. Thus, our results on aversion to implementation complexity do not necessarily preclude e.g. GARP consistency. Interestingly, although subjects in experiments often have trouble with optimization in experiments, experimental revealed preference tests often find strikingly high rates of GARP-consistency.

[11]Of course implementation complexity is unlikely to exhaust what we mean when we say a decision problem is "difficult." Many decisions are difficult because the most attractive and immediate models of the problem or procedures for solving them are simply wrong. For instance, in the famous Monty Hall problem or the Wason selection task, the computations required to choose optimally are trivial but most people make fundamental mistakes because the intuitively appealing rule is faulty. Understanding how the salience, attractiveness or obviousness of decision procedures competes with raw implementation costs to determine the selection of procedures is an important topic for future research.

[12]Doing this may require future researchers to measure complexity costs using richer types of algorithms than simple automata. For instance computer scientists typically rely on Turing machines to benchmark computational complexity and experiments studying willingness to pay to avoid Turing machines may therefore be valuable for understanding what makes problems computationally complex for humans.

[13]There is a growing literature in cognitive science developing the idea that using mental resources in complex tasks generates direct costs (e.g. Kool & Botvinick 2018, Shenhav, Musslick, Lieder, Kool, Griffiths, Cohen & Botvinick 2017). A common interpretation in this literature is that complex problems are costly because they require the decision maker to exert "cognitive control" over mental resources that are otherwise deployed reflexively in "automatic" default mental processes. Shutting down these automatic processes (and keeping them at bay) in order to focus cognitive resources on a complex problem that requires dedicated and sustained monopolization of computational resources is costly. Some evidence suggests that this is a top-down function of the executive network in the brain (related especially to the dorsal anterior cingulate cortex). Under this lens, a natural interpretation of our finding that complexity costs fall with learning (Section 4.4) is that familiarity allows subjects to follow rules more automatically, requiring them to exert less cognitive control.

[14]There is a recent theoretical literature in economics studying the cost of using complex reasoning procedures (e.g. Ergin & Sarver 2010, Ortoleva 2013, Alaoui & Penta 2018) as well as a large and growing literature on the costs of attending to information (e.g. Sims 2003, Caplin 2016, Dean & Neligh 2019). Experimental work has been broadly supportive of this latter line of research, producing evidence in favor of the general modeling strategy (Dean & Neligh 2019) and methods for measuring these attentional costs directly (Caplin, Csaba, Leahy & Nov 2019, Dewam & Neligh 2020). Our experiment follows the automata literature in treating features of algorithms as primitives over which subjects have preferences, but it is quite possible that elaborations on frameworks like rational inattention could be applied to the problem of implementing an algorithm, rooting algorithmic preferences in attentional costs.

circumstances of the agent).[15] With this distinction in mind, our experiment is designed to articulate an empirical mapping between "objective complexity" characteristics (e.g. "state complexity") and "experienced complexity" responses like costs. We will sometimes refer to our cost measures directly as "complexity" (evoking experienced complexity) and sometimes as "the cost of complexity" or "complexity costs" (emphasizing cost as a response to the objective complexity characteristics of the rule).

2 Automata and Complexity

Consider a decision maker (DM) who must make a series of responses ($r_t \in R$) to a series of events ($e_t \in E$) at a sequence of dates $t = 1...T$. We can think of *events* as true historical events (e.g. others' actions in a game as in Rubinstein (1986)) or as pieces of information considered one at a time in some order (as in e.g. Wilson (2014)). Likewise, we can think of *responses* as either actions made at each date (e.g. actions in a game) or as beliefs developed cumulatively or decisions made provisionally as facts are progressively considered (as in e.g. Salant (2011)).

The DM follows a *rule* that, at each date, maps the history of events so far into a current response. In applications, this could be a rule the DM chose on her own (e.g. which strategy to use in a game or what procedure to use to guide choice) or a rule provided by her environment (e.g. a series of steps and contingencies required to properly file taxes).

The "automata literature" builds on the insight that a rule can be described algorithmically as a *finite state machine* (or *finite automaton*), a basic formalization of algorithms used in computer science (e.g. Hopcroft & Ullman (1979)).[16] Formally, a *finite state machine* is a four-tuple (S, s^0, f, τ) where S is a set of *states*, s^0 is the initial state, $f : S \to R$ is an output function, specifying a response in each state, and $\tau : S \times E \to S$ is a *transition* function that specifies a next state as a function of the current state and the current event. The number of states, s_i, in state machine m is $|S_m|$, the number of total transitions from state s_i is $N(s_i)$ and the number of total transitions in machine m, summed over states, is $N(m) \equiv \sum_{s_i \in S_m} N(s_i)$. Figure 1 represents two automata visually in the standard way, as directed graphs with (i) circles representing states, (ii) arcs representing total transitions, (iii) letters inside of the circles representing responses from a set $R = \{C, D\}$ and (iv) letters next to transitions represent events from a set $E = \{C, D\}$. A free standing arrow pointing at the left-hand state indicates the initial state, s^0.

The automata literature poses several hypotheses about what characteristics (e.g. states, transitions etc.) of automata generate *complexity* where "complexity" is interpreted as the cost of *implementing* a rule – a way of ranking the DM's preferences over implementing different rules. Thus, rule A is more complex than rule B for a DM if the DM would prefer to implement rule B, ceteris paribus. We will follow suit by focusing attention throughout the paper on the cost of complexity

[15] This literature proposes (usually informal) taxonomies of the objective characteristics of tasks that summarize their complexity (i.e. proposes definitions of objective complexity) and also discusses how subjective characteristics (e.g. learning, skill, workplace environment) influence experienced complexity. See Campbell (1988) for an influential overview and Liu & Li (2012) for a recent survey.

[16] Automata can be used to describe a broad range of rules. For instance Kalai & Stanford (1988) show that finite automata can be used to formally describe any finite strategy in a repeated game.

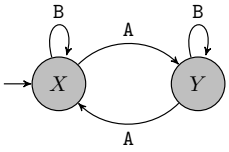
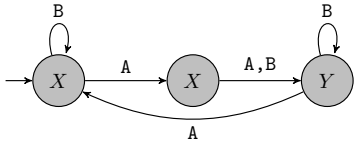
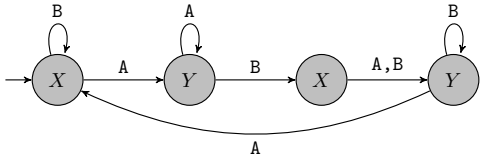
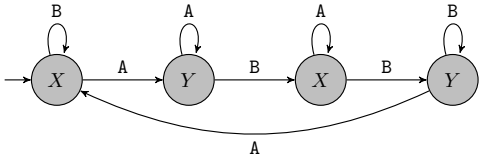
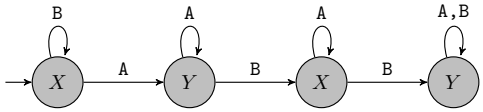
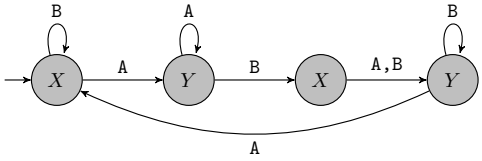
Hypothesis	Simple	Complex
s-complexity *# of states to track and remember.* (Hypothesis 1)	 *2S-2T*	 *3S-2T*
t-complexity *# of events to monitor, respond to.* (Hypothesis 2)	 *4S-3T*	 *4S-4T*
absorption *Do tracking and monitoring requirements end?* (Hypothesis 3)	 *4S-3Ta*	 *4S-3T*

Table 1: Hypotheses on the *structure* of rules. *Notes: On the left we include an example of a simple rule and on the right a complex rule under the Hypothesis listed on the far left.*

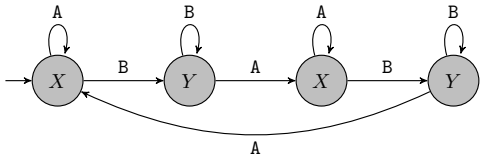
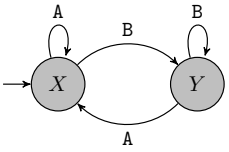
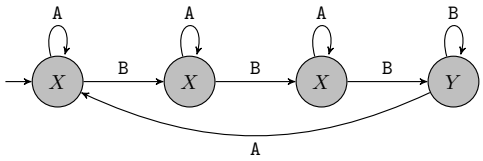
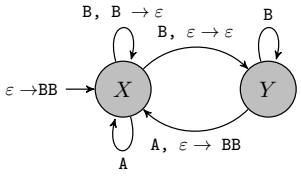
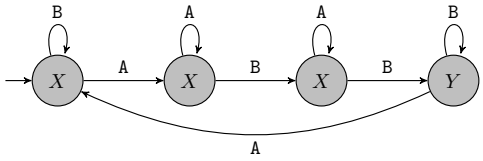
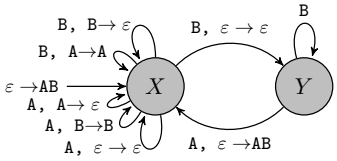
Hypothesis	Given Form	Reduced Form
State Reduction *Can the rule be represented as a finite state machine with fewer states?* (Hypothesis 4)	 *reducible* (Finite State Machine)	 *reduced* (Finite State Machine)
Stack Representation *Can state-tracking be replaced with a simpler working memory stack?* (Hypothesis 5)	 *countable* (Finite State Machine)	 *countable* (Pushdown Automoton)
	 *sequenceable* (Finite State Machine)	 *sequenceable* (Pushdown Automoton)

Table 2: Hypotheses on the *representation* of rules. *Notes: On the left we include an example of the finite state machine that closest resembles the statement of the rule. One the right we include the simpler way it can be represented under the Hypothesis on the far left.*

or "cost complexity."[17,18] However, we will also consider other natural metrics of complexity such as implementation time ("rules are more complex if they take longer to implement") or mistakes rates ("rules are more complex if they are harder to correctly implement").

2.1 Structural Hypotheses: States, Transitions and Absorption

We begin with a set of hypotheses that relate to the *structure* of the finite state machine (FSM) representation of the rule. These theories are listed in Table 1 which includes an illustrative example of a relatively simple (on the left) and relatively complex (on the right) rule under each theory.

The earliest (and by far most popular) hypothesis in the automata literature is *state complexity* (or *s-complexity*) (e.g. Rubinstein 1986, Abreu & Rubinstein 1988). States describe the distinct ways that a rule requires the decision maker to (i) act (choose a response) and (ii) process information (react to events) and s-complexity is the claim that partitioning behavior in this way is costly: the DM prefers rules requiring fewer states to those requiring more.

Hypothesis 1. ***(s-complexity)*** *A rule A is more complex than a rule B if A has more states than B ($|S_A| > |S_B|$).*

Table 1 provides an example of s-complexity: the right hand rule (3S-2T) is more s-complex than the left hand rule (2S-2T) (it has an additional circle in its graph and thus an additional state).

Banks & Sundaram (1990) hypothesize that another central feature of a rule – transitions – drive complexity. Adding a transition to a state (adding additional arcs in the graph of a rule) requires the DM to more precisely monitor and differentiate between events at that state in order to successfully implement the rule. *Transition complexity* (or *t-complexity*) hypothesizes that this increase in monitoring demands is costly. In order to focus on monitoring and to avoid confounding our transitions measure with s-complexity[19] we operationalize t-complexity using "net transitions" (the number of transitions in excess of the number of states) as described in Banks & Sundaram (1990), and we will refer to these simply as "transitions" throughout the paper:[20]

Hypothesis 2. ***(t-complexity)*** *A rule A is more complex than a rule B if A has more transitions in excess of states than B ($N(A) - |S_A| > N(B) - |S_B|$).*

[17]Often this literature uses the term "complexity" and "implementation cost" interchangably. Computer science defines the complexity of an algorithm in a closely related way: an algorithm A is more complex than B if it uses more *resources* than B. Computer science typically focuses on two forms of complexity, each linked to a different resource required to implement an algorithm: *time complexity* is the amount of time required by a computational device to implement an algorithm while *space complexity* is the amount of working memory the computational device needs to successfully do so. In humans a third, resource-driven measure of complexity suggests itself that we might call *hedonic complexity* – the disutility experienced directly by implementing the rule (e.g. because doing difficult things is unpleasant). An interpretation of the cost complexity studied in the automata literature is that it is the sum of these, with hedonic complexity adding disutility directly, time complexity generating cost via the opportunity cost of time and space complexity raising the hazards of *cognitive overload*, leading to costly implementation mistakes.

[18]Although cost is a natural operationalization of complexity in settings like ours where it is relatively clear that the main thing varying is mental effort, it may raise some conceptual difficulties in other contexts. For instance costs of behavior in some settings may be driven by fear or aesthetic distaste and it may make less sense to refer to such behaviors as "complex."

[19]Each time a reachable state is added to a machine, at least one additional transition must be mechanically added. Because the goal of the experiment is to differentiate between the effects of elements like states and transitions it is important to use a measure of transitions that does not include this redundancy with our measurement of state-complexity.

[20]This is Measure 3 in Banks & Sundaram (1990).

The t-complexity row of Table 1 provides an example: the right hand rule (4S-4T) is no more s-complex than the left hand rule (4S-3T) but is more t-complex because it has one additional arc in its the graph.

Finally, rules can differ in broader "architectural" characteristics, such as whether they contain *absorbing states* – states with no transitions leading away from them. Absorbing rules are potentially less costly and difficult to implement than transient rules (those with no absorbing states) because in the former the DM eventually reaches a position in which she no longer has to track states (s-complexity effectively goes to zero) or monitor the environment (t-complexity goes to zero). Although it is intuitive that "architectural" features of rules such as transience and absorption might influence complexity, the literature does not explicitly model absorbing rules as less complex.

Hypothesis 3. ***(transience and absorption)*** *Ceteris paribus, rule A is more complex than rule B if B contains an absorbing state while A does not.*

The absorption row of Table 1 gives an example: the rule on the left (4S-3Ta) has identical states and transitions to the rule on the right (4S-3T), but while the rule on the left has an absorbing state (at the rightmost state of the graph), the rule on the right does not. Under the hypothesis that rules are less complex if absorbing, the rule on the right is more complex.

Two other related ideas from the literature bear mentioning. First, Lipman & Srivastava (1990) point out that automata differ in their *robustness* – in the degree to which mistakes in perceiving events lead to serious mistakes in following the rule's prescription. We consider this and its bearing on complexity costs in the context of the data in section 4.5. Second, Chatterjee & Sabourian (2000) discuss a refinement of s-complexity that they call response complexity or r-complexity, which applies to settings that differ somewhat from ours (settings in which extensive form games must be played in each state). A rough translation to our setting is that a rule is more r-complex if it requires more changes in behavior (in "responses") when states change. This is connected to issues discussed in Section 2.2.2 below.

2.2 Representational Hypotheses: Redundancy and Memory

A rule is a way of *representing* a plan of action and, in general, there will be many (indeed infinitely many) ways a decision maker can algorithmically represent any plan of action to herself as a rule. Any account of procedural complexity therefore must include an account of how decision makers represent rules to themselves – in particular an account of how *efficiently* they represent rules to themselves. The automata literature has developed under two assumptions about representation that we will use to articulate two *representational* hypotheses in the next two subsections.

2.2.1 Eliminating Redundant States

First, the automata literature generally assumes that decision makers use representations of rules that economize on states. A rule's complexity is therefore governed not by the algorithmic features of the rule itself but by the *lowest-state* equivalent rule that delivers an identical sequence of actions over all possible sequences of events. This is an efficiency notion for complexity, postulating that a DM will not waste cognitive effort in following the representation of a rule given to her (or conceived by her) if there is an available equivalent representation that has fewer states. Behaviorally, this is

equivalent to a hypothesis that a DM will find a heuristically simpler representation if it is available, re-representing a rule to herself in a less structurally complex way whenever possible.

Hypothesis 4. ***(State Reduction)*** *Suppose rule B can be represented as a finite state machine with (i) no more states and transitions and (ii) either strictly fewer states or strictly fewer transitions than rule A. Then rule A is more complex than rule B.*

The State Reduction row of Table 2 provides an example. The rule on the left shows the rule *reducible* which is (under Hypotheses 1-3) identical to the rule *4S-4T* from Table 1. But 4S-4T cannot be reduced to a simpler rule – there is no automaton with fewer states or transitions that delivers the same sequence of actions in response to all histories. By contrast, the rule *reducible* can be replaced with (reduced to) the rule we call *reduced* (shown to the right of *reducible*) – the rule *reduced* produces exactly the same set of actions after all sequences of events as *reducible*, but has half as many states and transitions. Thus, to the degree that states or transitions drive complexity, *4S-4T* is more complex in the sense of State Reduction than machine *reducible* even though the two are superficially almost identical. Moreover, to the degree that decision makers represent rules efficiently, we would expect them to treat a rule described as *reducible* as equivalently complex as a rule described as *reduced*.[21]

2.2.2 Replacing States with Working Memory

More fundamentally, the automata literature implicitly assumes that decision makers represent rules as finite state machines (FSMs) – an assumption that limits the way decision makers can economize on complexity in some cases. For example, the FSM of the rule *countable* (left side of Table 2) has the same number of states, transitions etc. as the FSM of *4S-4T* (Table 1), but states serve a much more primitive role in the former rule than in the latter. While states in *4S-4T* provide four unique sets of instructions for what to do and how to interpret events, three of the states in *countable* are identical: these states exist purely to keep track of the number of 'B' events that have occurred so far. Although states serve a much simpler (largely mnemonic) function for *countable* than *4S-4T*, under the descriptive lens of FSMs the two rules are identical.

To represent the intuitive difference between the way the two rules make use of states, we must allow the DM to represent rules with algorithmic structures nuanced enough to separate out the memory function of a state from its more complicated functions. This requires only a minimal iteration beyond an FSM called a "pushdown automaton" (PDA). In addition to states and transitions, a PDA maintains a "stack" – a string of symbols – in working memory and can (i) "read" the top element of the stack to determine when to transition, and (ii) can either add ("push") a new top element to the stack or can remove ("pop") the top element during any transition. The right side of Table 2, shows a PDA representation of *countable*. Here the transition notation `h, i` $\rightarrow$ `j` says "Make this transition if event h has occurred *and* the top element in the stack is i. Replace the top element in the stack with j." Under this notation, replacing an element with ε removes the element and the element ε is read by the machine only if the stack is empty. The PDA for *countable* begins with an initial transition that pushes a string of symbols,"BB," onto the empty stack ($\varepsilon \rightarrow BB$).

[21] In the experiment, following the structure of the algorithm, we describe *reducible* as "Choose x until you see b, after which switch to y. Then, choose y until you see a, after which point switch to x. Then, after you see b, switch to y. Then, start over after you see a" while we describe *reduced* as "Choose x until you see b, after which switch to y. Then, start over after you see a." Hypothesis 4 argues that subjects should internally replace the former description with the latter.

The stack is now of length two, ordered from left (top) to right (bottom). In the initial (left hand) state, when a 'B' occurs: (i) if the stack is non-empty, the top 'B' in the stack is 'replaced' with ε and thus removed ($B \to \varepsilon$) and (ii) once the stack is empty (i.e. once there are no 'B' symbols left, meaning the top of the stack contains ε) the automaton transitions to the right.

Representing *countable* as a PDA-like rather than FSM-like rule economizes on s-complexity: it removes two states without adding any transitions, and it does so by instructing the DM to *count* events rather than track many states.[22,23] Our final hypothesis is that rules that can be simplified in this way, using working memory to replace states, are less complex.[24]

Hypothesis 5. ***(Stack Representation)*** *Suppose rule B can be represented as a pushdown automaton with (i) no more states and transitions and (ii) either strictly fewer states or strictly fewer transitions than rule A. Then rule A is more complex than rule B.*

In order to test the hypothesis we describe the rule *countable* to subjects in the natural way using a PDA-like description: "Choose x until you've seen b three times so far (not necessarily in a row), after which switch to y. Then, start over after you see a." Our hypothesis is that this is less complex than "4S-4T" which can be described in no simpler way than the FSM-like "Choose x until you see a, after which switch to y. Then, after you see b switch to x. Then, after you see another b switch to y. Then, start over after you see a."[25] Counting can be represented in this particularly simple way in part because it uses an especially simple kind of memory that we call an "event stack": the rule adds a string of anticipated future events upon entering the initial state, removes elements iteratively as the corresponding events occur and transitions once the stack is empty.[26,27] Indeed, counting can be thought of as a simple instance of *sequencing*: remembering and eliminating a series of (possibly irregular) events before triggering a change in behavior. The rule *sequenceable* at the bottom of Table 2 gives an example. The FSM representation on the left is almost identical to the rule *countable* and like *countable* uses states mostly for memory and therefore can be reduced to a PDA with an event stack (in this case, initalized with "AB"). However, because *sequenceable* requires a richer (irregular) stack, the reduced version of *sequenceable* requires three additional (net) transitions in exchange for the removal of two states, and thus (unlike *countable*) doesn't strictly fit Hypothesis 5. Nonetheless, depending on the relative complexity costs of states versus transitions,

[22] A natural interpretation is that by separating states from working memory, PDAs distinguish between the (i) "information processing" and "behavioral" functions of a state and (ii) the plausibly less burdensome "memory" function of a state, and thus allow us to attribute different complexity contributions to these functions. In an FSM representation these functions are all constrained to be treated identically.

[23] By contrast, as we show in Online Appendix C, no such "free lunch" is available for rules like *4S-4T*: in order to remove states using a PDA representation, the rule must, in exchange, add a great many additional transitions to handle the information processing required by states in that rule.

[24] Note that this implicitly assumes that holding a string in working memory does not contribute much complexity cost of its own – an assumption that we therefore also test when we test this hypothesis.

[25] An alternative way to test the hypothesis would have been to describe *countable* to subjects with a much less natural FSM-like description: "Choose x until you see b. Then choose x until you see b. Then choose x until you see b at which point switch to y. Then, start over after you see a." We strongly suspect subjects would immediately reduce this to PDA form (and wonder why we chose to describe the rule in such an unnatural way).

[26] Event stacks can be used to simplify precisely when contiguous states prescribe exactly the same response and is thus related to response or r-complexity (Chatterjee & Sabourian 2000) discussed in Section 2.1.

[27] Pushdown automata are more general than finite state automata: any rule that can be represented by the latter can be represented by the former. Indeed, all of the rules we study in our experiment can be represented as 2-state PDAs, generally by adding a great many transitions. However, for most rules this cannot be done with an event stack, and require a far more elaborate coding system on the stack than an event stack allows. In Online Appendix C, we translate all of the rules studied in the paper into two state PDAs and discuss these representational issues in more detail.

we might expect a net complexity reduction for *sequenceable* as well (though a smaller one than for *countable*).

2.3 What Else Influences Procedural Complexity?

Other features of rule-following that have little to do with the structure or representation of a rule seem also likely to influence experienced complexity. Perhaps, most important among these is learning and resulting familiarity with a rule. We hypothesize that experience with a rule may allow a decision maker to build up skill that reduces experienced complexity via a process that cognitive scientists call *procedural learning*. Another potentially important factor that may influence complexity, is the context in which the rule is implemented. For instance, some procedures are followed by taking an explicit series of steps (for instance taking a series of actions in a repeated game), while others are followed to form beliefs or to forecast the behavior of others (predicting the consequences of another's strategies, calculating the consequences of an asset allocation, analyzing a sequence of pieces of information to form a belief etc.). Following a procedure "in one's head" seems more difficult and therefore plausibly more costly, even if the algorithmic structure of the procedures required in each case are identical.

In the experiment described below we consider both of these potential "non-algorithmic" drivers of complexity. However, we note that there are many other factors, both algorithmic and non-algorithmic, that plausibly influence complexity and that our design does not explore. For instance, it seems plausible that features of the underlying decision problem like the number of potential events or the number of potential responses to events influence complexity. Because none of the hypotheses above concern these features of the decision environment, our experimental design will hold both constant (at two events and two responses). Likewise, none of the hypotheses in the extant literature explicitly concern issues of the architecture of rules (e.g. complex branching structures), beyond those summarized by states and transitions. We stated a hypothesis concerning absorption (Hypothesis 3) which has to do with whether the architecture of automata are "linear" or "circular." But clearly many, more complicated branching structures could be explored and their influence on complexity characterized. We believe these issues are important topics for future research.[28]

3 Experimental Design

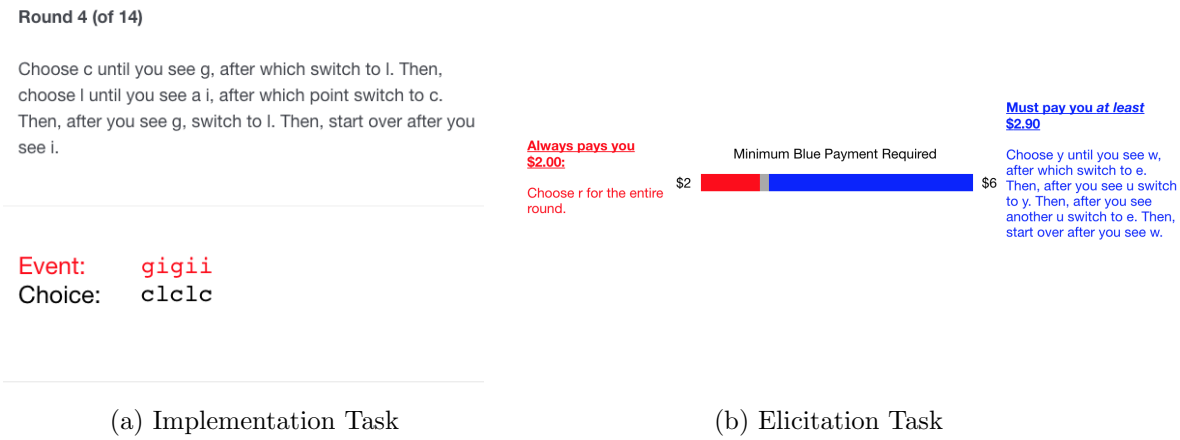
The experimental design consists of five Stages (summarized in Figure 3), run in sequence in each session. In Stages 1, 3 and 5 we ask subjects to complete a series of *Implementation Tasks*:

Implementation Task: In each Implementation Task, we assign subjects a verbal *rule* and show them a sequence of twenty 'Events' (randomly selected letters), one at a time.[29,30] The subject's

[28] Likewise, our design hardly exhausts the set of possible influences of *context* on complexity. For instance *meaning* may influence complexity directly. A canonical example is the empirical literature on the Wason selection task (a card task testing deductive reasoning): when the task is framed abstractly subjects do extremely poorly, but they often solve the task correctly when it is framed more meaningfully, for instance as a social exchange or cheater detection problem (e.g. Cosmides 1987). It is possible that rules implemented in non-abstract, meaningful contexts have lower complexity costs and examining whether this is the case seems a promising direction for future work.

[29] For example the rule we call *2S-1T* (the Grim rule from Figure 1) has a verbal description "Choose X until you see B, then switch to Y forever," while the rule *4S-4T* (shown in Table 1) has a description "Choose X until you see A, then switch to Y. Afterwards, if you see B switch to X and then to Y if you see B again. Start over if you see A."

[30] These Events are randomly selected by the computer from a set of two possible Events following each Choice, and



(a) Implementation Task (b) Elicitation Task

Figure 2: Screenshots from the experimental software.

task is to "implement the rule" by typing the sequence of letters ("Choices") prescribed by the rule, given the random events that have occurred. Given the sequence of events, there is a unique sequence of twenty Choices that complies with the rule. If the subject perfectly types this sequence, she earns a monetary reward (otherwise she earns nothing).[31]

Example: *Figure 2 (a) shows an example of an Implementation Task. The subject was shown a rule at the top of the screen ("Choose c until you see g, after which...") and typed a first letter ('c') which was displayed in the first position on the "Choice" line. The computer responded by randomly selecting an Event 'g' (drawn equiprobably from a set of 2 possible letters) and displayed it in red in the first position on the "Event" line. The subject, following the rule, next typed 'l' and a new random Event 'i' immediately appeared in the next position on the Event line, etc. In the example, the subject has made 5 (out of 20 total) Choices in the Task so far.*

In Stage 1, we assign subjects 14 Implementation Tasks in sequence, each with a different assigned rule (and each consisting of 20 Choices). The rules are drawn from a fixed set of 14 (described in Section 3.1) and are assigned to subjects across Tasks in a random order. At the end of the experiment, the computer draws one Stage 1 Implementation Task at random and the subject earns $4 if she implemented that Task correctly. The purpose of this Stage is to study how markers of complexity like implementation time (how long it takes the subject to implement the rule) and mistake-making (how likely the subject is to make a mistake) vary with characteristics of the rule.

In Stages 2 and 4 of the experiment we ask subjects to complete several *Elicitation Tasks*:

Elicitation Task: In each Elicitation Task we show subjects a maximally simple "red rule" (directing subjects simply to type the same letter repeatedly) and a more elaborate "blue rule." The subject knows that one of these two rules will be assigned to her in a future Implementation Task. In that future Task, the red rule will pay $2 while the blue rule (if correctly implemented) will pay some unknown random amount x drawn uniformly from $[\$2, \$6]$. The subject's task is to use a slider to give the minimum amount m for which she would be willing to be assigned the blue rule

subjects are informed that Events are randomly selected. At the beginning of each Implementation task, we randomly select the set of two letters used to represent Events and the two possible Choices (responses). That is, we replace the letters in the event set $\{A, B\}$ and response set $\{X, Y\}$ each with a randomly selected letter in forming the rule.

[31] The subject is not told whether she made a mistake until the end of the Task, at which point she is shown the correct sequence of Choices on her screen as feedback.

rather than the simpler red rule in that future Implementation Task. If $x \geq m$ the subject will be assigned the complex blue rule, otherwise she will be assigned the simple red rule.[32]

Example: *Figure 2 (b) shows an example of an Elicitation task. The subject was shown the red rule ("Chose r...") on the left side of the screen and a blue rule ("Choose y until you see w...") on the right. The subject submits m (the "minimum blue payment required") by moving the slider and sees the results ("must pay you at least..") above the blue rule. In this example, the subject has set the slider to \$2.90, meaning she would prefer to be assigned the red rule unless the blue rule pays at least \$2.90.*

The choice, m, gives us a measure of the subject's subjective cost of implementing the blue rule – the amount the subject is willing to "leave on the table" to avoid implementing the rule in the future. Since the simple red rule pays \$2, our cost measure – the money the subject is willing to sacrifice to avoid the more complex blue rule – is $c = m - 2$ and we measure this cost for each rule assigned in an Elicitation Task. In the example, the subject has submitted \$2.90, meaning her elicited cost, c, of implementing the blue rule is \$0.90.

In Stage 2, we assign subjects 14 Elicitation Tasks, each eliciting willingness-to-pay to avoid a different (blue) rule. These rules are drawn (and assigned in a random order) from the same set of 14 rules we assign for Implementation Tasks in Stage 1 (meaning subjects have implemented each of these rules prior to being asked to value them).[33] In order to incentivize the elicitations, we inform subjects that at the end of the Stage, one Elicitation will be randomly drawn and used to determine which rule the subject will be assigned to implement repeatedly (in ten separate, sequenced Implementation Tasks) in Stage 3. The purpose of Stage 2 is to measure the complexity cost of each of our rules and study how this cost varies with characteristics of the rules.

3.1 Rules Studied in the Design

We designed our 14 rules to evaluate the hypotheses in Section 2. Table 3 lists all 14 rules, with columns designating the number of states in the finite state machine representation of the rule and rows the number of net transitions (transitions in excess of the number of states). We divide these rules into three types:

- Most cells of Table 3 include a rule with a name of the form "xS-yT" where x designates number of states and y number of (net) transitions. These are our **core rules**, designed to test the effects of states and transitions on complexity (Hypotheses 1 and 2).
- Some cells include a rule with a name of the form "xS-yTa," where 'a' designates rules containing an absorbing state. These **absorbing rules** are used to test the effects of transience

[32] This task is formally equivalent to a Becker-Degroot-Marschak (BDM) elicitation. Cason & Plott (2014) document that in BDMs framed as auctions, some proportion of subjects confuse the second price auction incentives of the mechanism with first price auction incentives (e.g. they believe they will earn their bid). We designed our mechanism with these results in mind, taking several steps to reduce the likelihood of such errors. First, we do not frame the mechanism as an auction but instead ask each subject explicitly to submit a payment amount that will make her indifferent between the simple and complex rule. Second we explicitly describe the incentive compatibility of submitting one's true indifference point. Finally, we explicitly warn subjects against the specific mistake Cason & Plott (2014) document in their data, emphasizing to subjects that they will be paid the randomly drawn payment x if assigned the complex rule, *not* the indifference point m they submit. Our detailed instructions for the Elicitation task are reproduced in Online Appendix D.

[33] After completing the 14 Elicitation Tasks one-at-a-time, subjects are shown their choices for all of the elicitations and given a last chance to fine tune them.

	2 State	3 State	4 State
1 Transition	2S-1Ta	3S-1Ta	4S-1Ta
2 Transition	2S-2T reduced	3S-2T	4S-2T
3 Transition		3S-3T	4S-3T 4S-3Ta
4 Transition			4S-4T reducible countable sequenceable

Table 3: Rules implemented in the design.

and absorption (Hypothesis 3).

- Some cells include one or more rule with names that aren't of the "xS-yT" form. These are our **representation rules**, designed to be compared to a core ("xS-yT") rule in the same cell to test Hypotheses 4 or 5. For instance the rules *reducible* and *countable* are designed to be compared to *4S-4T*, with which they share a cell.

Our empirical strategy is to compare behavior across rules in order to measure the average effects of rule characteristics (e.g. states, transitions) on complexity measures gathered in Stages 1 (implementation time, mistakes) and especially Stage 2 (complexity costs). This, in turn, allows us to evaluate our hypotheses. For instance *2S-2T*, *3S-2T* and *4S-2T* vary states while holding transitions constant at two. Comparing elicited costs (or implementation time or mistakes) across these rules lets us test s-complexity (Hypothesis 1). Likewise, *4S-2T*, *4S-3T* and *4S-4T* vary transitions while holding states constant at four, letting us evalute t-complexity (Hypothesis 2) by comparing their costs. 4S-3T and 4S-3Ta share identical states and transitions but only the latter contains an absorbing state, enabling a test of Hypothesis 3. Comparing *reducible* and *countable* each to *4S-4T* give us tests of Hypotheses 4 and 5 respectively. The other rules are used in the analysis to support and clarify these hypothesis tests.

Most (nine) of the rules from Table 3 are shown in automaton form in Tables 1 and 2 and Figure 1. The remaining rules are shown in Online Appendix A, which also provides the verbal descriptions we gave to subjects.

3.2 Further Stages

In Stage 3, we assign subjects a single rule to implement repeatedly over ten consecutive Implementation Tasks (each with 20 events and choices as in Stage 1). The primary purpose of Stage 3 is to incentivize the elicitations in Stage 2 by assigning subjects rules based on the preferences they submitted. However, a second purpose is to study how learning and familiarity impact complexity. In order to accomplish both goals, we randomly assign half of our subjects as "Endogenous Assignment" types and half as "Exogenous Assignment" types (subjects were never told which type they were). For Endogenous Assignment types, the computer randomly selects one Stage 2 elicitation and a random payment x, and uses these to determine which rule to assign the subject in Stage 3. Exogenous Assignment subjects are instead assigned a rule without reference to

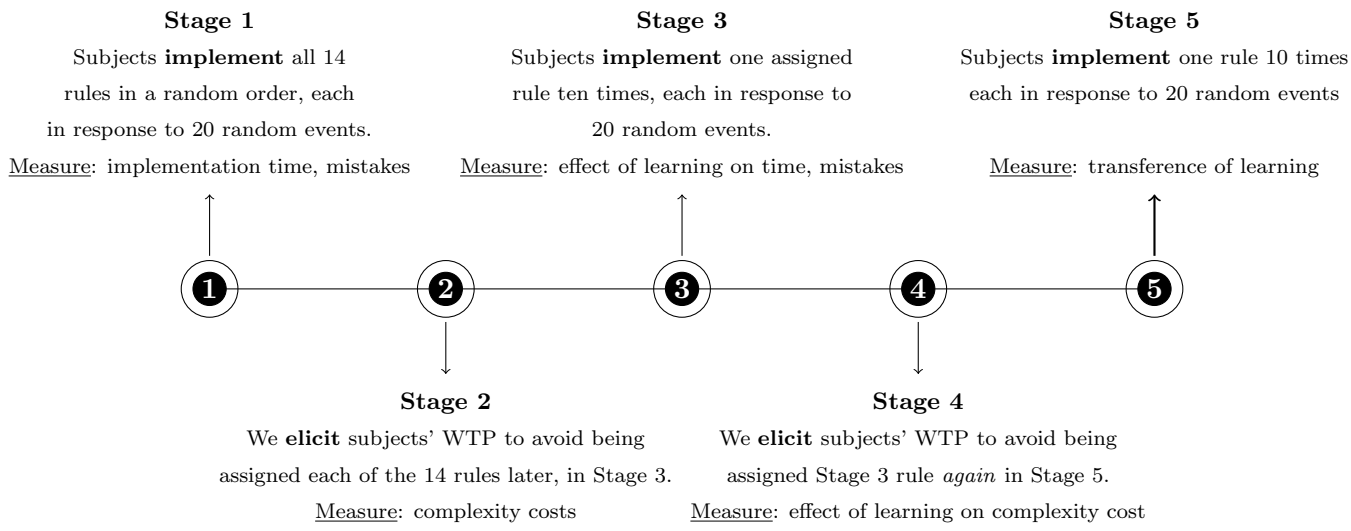


Figure 3: Summary of the Stages in the experimental design *Notes: "Measure" gives the aspect of complexity each Stage was designed to allows us to measure.*

their willingness-to-pay, and in the main portion of our dataset, are simply assigned either the rule "4S-4T" or "reducible." While Endogenous Assignment subjects self-select into their Stage 3 rule via their submitted preferences, Exogenous Assignment subjects do not and their learning data is therefore easier to interpret. Regardless, subjects are paid based on their performance in one of these Implementation Tasks, randomly selected at the end of the experiment.

In Stage 4, we re-elicit subjects' willingness to pay to avoid being assigned (in Stage 5) the same rule they Implemented repeatedly in Stage 3. Our goal with this Elicitation Task is to study whether familiarity with a rule reduces subjects' costs of implementing the rule. By comparing willingness-to-pay before and after learning in Stage 3, we are able to measure the effects of learning on complexity costs for the Stage 3-assigned rule. In order to understand learning effects better, we assign two different Elicitation Tasks in this Stage: one is for the Original version of the rule (the exact same rule implemented in Stage 3) and the other is for a superficially Perturbed version of the same rule. Specifically the Perturbed version is representable by the same underlying automaton as the Original version, but we randomly change all of the letters designating events and choices.

In Stage 5, we "pay off" the Stage 4 elicitation by having subjects once again play a rule ten times (ten Implementation Tasks). As in Stage 3, Endogenous Types are assigned based on their (this time, Stage 4) preferences and a randomly drawn x, while Exogenous Types are randomly assigned either the Original or Perturbed version of the Stage 3 rule. Subjects are paid based on their performance in one of these Implementation Tasks, randomly selected at the end of the experiment.

3.3 Details

We ran the experiment using 275 subjects (mostly undergraduates) in March, April, July and August of 2019 at UC Santa Barbara. We recruited subjects using ORSEE (Greiner (2015)), allowed subjects to participate in only one session each and used custom Javascript software programmed by the author and deployed using Qualtrics. Sessions lasted roughly 90-120 minutes and subjects, on average, earned about $20.

We ran sessions using three contextual variations on the Implementation task (varied between subjects) which we pool for most of the analysis.[34] In the **base** version (110 subjects) subjects follow the Implementation task described above. In the **recall** version (111 subjects), the task is identical but subjects can only see the *latest* event and choice on their screen at any one time. Finally, in the **reasoning** version (54 subjects) subjects see all 20 events at the very beginning and are asked to forecast what a perfect rule-follower would choose in the very last (of 20) choices, forcing them to implement the rule entirely "in their heads." By comparing the Base version to the Reasoning version we can study the difference between (i) actually enacting a rule (e.g. implementing a strategy) and (ii) using a rule mentally to make an inference or forecast an event (e.g. reasoning through a decision tree, forecasting another's choices in a game). On the other hand, while Base mirrors a context in which decision makers have easy access to the history of events (for instance in which 'events' are options in a list or pieces of information being procedurally considered in some order), Recall mirrors a condition in which the decision maker must work to remember events (plausibly the case in implementing strategies in some repeated games contexts).

Each session began with detailed instructions on the Implementation task which we read out loud to subjects, and which focused especially on explaining the syntax of the rules. We encouraged subjects to ask questions during these and during five unpaid practice Implementation Tasks.[35] We paused the experiment to orally deliver instructions prior to Stage 2, and subjects afterwards moved along at their own pace, reading computerized instructions before each Stage. At the end of the experiment subjects participated in a battery of incentivized cognitive tests.[36] Additional details and instructions to subjects are reproduced in Online Appendix D.

We collected data in two "runs" of the experiment: Run 1 (57 subjects under Base, 59 under Recall) and Run 2 (53 in Base, 54 in Reasoning and 52 in Recall). The main difference between the Runs is that we allowed subjects to leave the lab as soon as they finished the experiment in Run 1 but forced subjects to wait until everyone was finished in Run 2 (to eliminate the possibility that subjects submit preferences for simpler rules because these rules take less time to implement, allowing them to leave the lab sooner). We also made slight clarifying changes to the wording of a few rules in Run 2 and changed the set of rules we assigned to Exogenous Assignment subjects in Stage 3: in Run 1 we randomly assigned from the full set of rules while in Run 2 we randomly assigned only from the two most measured-complex rules, *4S-4T* and *reducible*.[37] The data across the two runs are not statistically different[38] and we pool them in the main analysis below (our results do not change if we instead drop Run 1 from the dataset).

[34] We pool these treatments because, as we discuss in Section 4.4, behavior is very similar (except for level shifts easily measured in regressions), across variations.

[35] In Reasoning sessions, we gave subjects initial instructions and practice periods from the Base condition in order to help them understand how to read and interpret the rules. We afterwards told them they would be required to forecast final choices rather than fully implement the rules in the actual experiment.

[36] We paid subjects \$0.25 for each question they got right. Cognitive tests include a truncated (3-question) version of Raven's Progressive Matrices (used to test abstract reasoning, fluid intelligence and generalized intelligence), three Belief Bias questions (used to test for logical over intuitive reasoning), three Cognitive Reflection questions (used to measure ability to overcome instinctive responses), the Wason selection task (used to measure deductive reasoning abilities) and five rounds of the Operation Span task (used to measure working memory capacity).

[37] We made this change to Stage 3 rule assignment in Run 2 in order to better study learning. Run 1 data produced too few observations on any given rule to be statistically useful and rules featured obvious variation in baseline complexity. By assigning subjects only two very complex rules in Run 2 we gather significant data on learning under each rule and are able to focus on a setting in which learning is especially valuable.

[38] We include an indicator variable for Run 1 in our main econometric work reported in Table 5 and find no effect. Further specifications interacting an indicator for Run 1 with each of the specifications in Table 5 likewise shows no differences across runs that are significant at conventional levels.

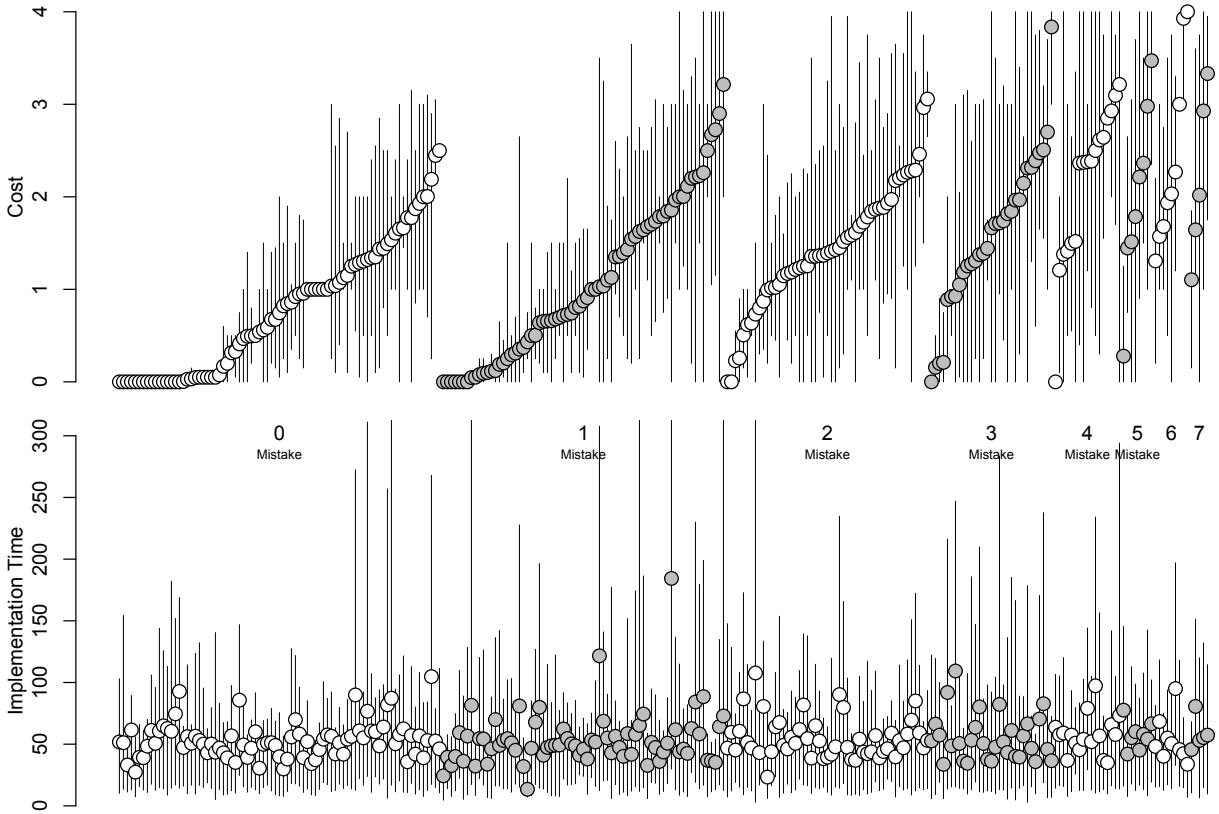


Figure 4: Average cost and implementation time for all subjects *Notes: Subjects are binned according to the number of mistakes they made (out of 14) in Stage 1, and organized from lowest to highest cost within bin. In the top panel the subject's mean cost is shown as a dot and her range of costs across rules as a vertical line. Mean implementation time and range of implementation times is shown in the bottom panel at corresponding horizontal locations for each subject.*

4 Results

Figure 4 provides a broad overview of the data at the subject level. The upper panel plots the average elicited complexity cost (over all 14 rules) for each subject in the dataset as a circle. Subjects are organized horizontally by the number of mistakes they made (out of 14) and are ordered from lowest to highest average cost within each of these mistake-count bins. Vertical lines plot the range of costs (from lowest to highest) for each subject. The bottom panel plots the corresponding average implementation time and range of implementation times across rules for each of these subjects (arranged horizontally according to the mistakes/cost ordering from the upper panel). The figure highlights several facts about the data.

First, most subjects make relatively few mistakes: the modal subject makes 0 and the median subject makes 1 mistake over the 14 rules. Second, most subjects – including subjects that rarely make mistakes – are willing to pay a considerable fraction of their expected earnings to avoid implementing the average rule. Just under 10% of subjects are willing to pay nothing to avoid complexity.[39] Third, there is a great deal of variation in costs across rules, within subject, indicating that subjects make strong distinctions across rules when evaluating cost: the median least costly rule costs \$0.20 while

[39]Given our elicitation methods, we cannot rule out the possibility that these subjects experience negative costs and actually prefer complexity.

Rule	States	Trans.	Absorbing?	Mean Cost	Median Cost	Low Mistake Cost	Mean Time	Mistakes Rate
2S-1Ta	2	1	Y	0.69	0.30	0.48	15	0.01
3S-1Ta	3	1	Y	0.78	0.50	0.59	20	0.01
4S-1Ta	4	1	Y	0.89	0.65	0.64	31	0.04
reduced	2	2	N	1.00	1	0.71	57	0.08
2S-2T	2	2	N	1.00	0.90	0.70	60	0.17
countable	4	4	N	1.23	1	0.87	49	0.07
3S-2T	3	2	N	1.29	1.05	0.93	67	0.15
3S-3T	3	3	N	1.33	1	0.91	63	0.24
4S-3Ta	4	3	Y	1.38	1.10	0.99	35	0.04
4S-2T	4	2	N	1.47	1.35	1.02	74	0.21
4S-3T	4	3	N	1.59	1.50	1.15	76	0.19
sequenceable	4	4	N	1.60	1.55	1.12	69	0.29
4S-4T	4	4	N	1.73	1.80	1.28	74	0.16
reducible	4	4	N	1.79	1.85	1.33	73	0.09

Table 4: Rule-level summary statistics. *Notes: Rules are ordered by the mean cost of the rule. The table lists the number of states in the rule, the number of transitions (net of states) and whether the rule contains an absorbing state. The table also provides mean and median cost, mean time spent implementing the rule and the mistakes rate under the rule. "Low Mistake Cost" is the mean cost for the subset of subjects who made no more than one mistake in Stage 1.*

the median most costly rule costs an order of magnitude more at $2.35. Fourth, average complexity costs are quite heterogeneous across subjects (the standard deviation of subject-wise means is $0.91), as are average implementation times (standard deviation 18 seconds). However, at the subject level, there is little relationship between the two: the cross-subject correlation between mean cost and mean implementation time is 0.084 ($p = 0.17$). Finally, regressing costs, implementation time and mistakes rates on a composite statistic of our battery of cognitive tests reveals that all three of these measures are significantly influenced by measures of cognitive ability (see Table 5, described below). That is, for all three measures of complexity, subjects that perform worse in cognitive tasks find rules, on average, more complex.

Result 0. *Most subjects make few mistakes but nonetheless find the average rule costly to implement and make strong distinctions in cost across rules. Variation in complexity measures including cost, implementation time and mistakes rates are significantly linked to independent measures of cognitive ability.*

4.1 Findings on Automata Structure: States, Transitions and Absorption

We now turn to evaluating our hypotheses concerning the determinants of complexity across rules (Results 1-5 below numerically correspond to the numbering of hypotheses in Section 2). Figure 5 summarizes the data at the rule level by showing average cost, implementation time and mistakes rate as a function of total transitions (transitions + states) across rules. Table 4, shows similar statistics numerically, ordering rules instead by mean cost.

The experimental design allows us to test our main hypotheses about the drivers of complexity by comparing average measured costs (and implementation time and mistakes rates) across rules. For instance, the rules *2S-2T* and *3S-2T* (shown in the s-complexity row of Table 1) have the same number of transitions but the latter has one additional state and is therefore more *s-complex*.

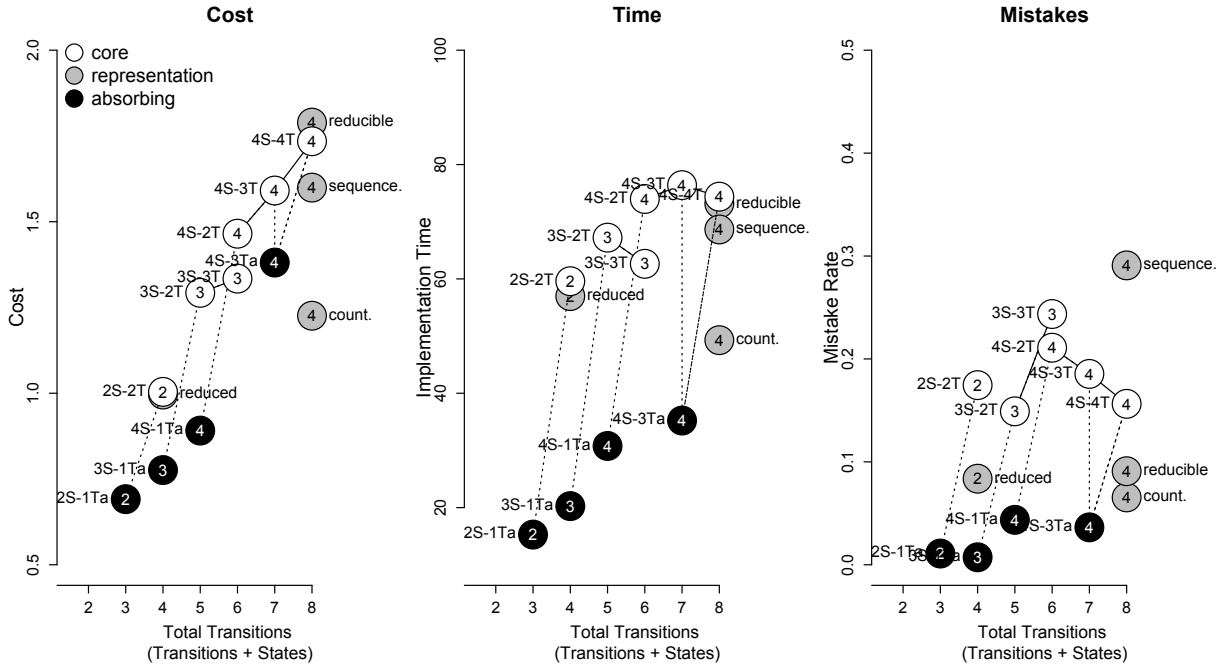


Figure 5: Average complexity measures for all rules. *Notes: Each circle in each panel represents a different rule. Horizontal location of each rule is "Total Transitions" (states plus transitions) and the number of states are shown inside each rule's circle. Lines connect all of the non-representation rules that contain the same number of states; dotted lines mark transitions that cause the rule to become transient (non-absorbing). Panels show mean cost, implementation time and mistakes rates respectively.*

As Table 4 shows, the average cost for *2S-2T* is 1.00 and for *3S-2T* is 1.29 indicating a 1.29-1.00=0.29 increase in costs (a 29% increase) from adding a single state to a rule. Averaging across all such controlled comparisons gives us an average effect of adding a state of 0.236 (a 20% increase), supporting the hypothesis that states generate complexity costs (our Hypothesis 1).

Formally, we test our hypotheses using a set of regressions, summarized in Table 5. Each specification corresponds to a different complexity metric (cost, implementation time and mistakes rate) and includes right-hand variables for the number of states, the number of transitions and an indicator variable for whether the rule is absorbing (interacted with both state and transition counts). The regressions also include indicators for the rules *reducible*, *countable* and *sequenceable* which we will examine in the next subsection, indicators for the Run of the experiment and for the context (Reasoning, Recall) in which the rule was implemented. Standard errors are clustered at the subject level.[40] Estimates from these regressions are visualized in Figure 8.[41]

We begin by examining the non-absorbing "core rules" (*2S-2T*, *3S-2T*, *3S-3T*, *4S-2T*, *4S-3T* and *4S-4T*), which were designed to be compared directly with one another to study state and transition

[40] A caution in interpreting the standard errors from these estimates is that some effects (e.g. states, transitions, absorption) are estimated using multiple rules and therefore multiple decisions by each subject, while others (e.g. countability, reducibility) are estimated using only a single per-subject choice.

[41] In Online Appendix B we include several robustness checks on our main cost specification including a Tobit and a specification that includes the word count of the rule as a dependent variable. In neither case do the results of our main Hypotheses tests change. For the Tobit, estimates are similar though we lose precision and significance on the *sequenceable* indicator. For the word count specification, the word count variable is not significant but other variables are attenuated due to its inclusion; the main qualitative change is that the *sequenceable* indicator, again, ceases to be significantly different from zero at conventional levels.

effects (s-complexity and t-complexity). The ordering in Table 4 (which, recall, is by mean cost), clearly shows that these rules are ordered first by state (every 4 state rule is more costly than every 3 state rule in this set, etc.) and secondarily by transitions. Moreover, the six most complex rules in the dataset are those with the maximal number of states used in the design. Table 5 confirms that, on average, adding a state has a large, significant impact on cost (equal to almost 20% of the average cost and a 26% increase over the intercept).[42] States also significantly impact implementation time and mistakes rates.

Result 1. *Adding a state to a rule has a large and significant positive effect on the cost of implementing it.*

By comparison, Table 5 shows that adding a transition, though statistically significant, adds half as much cost as adding a state.[43] Transitions have no significant effect on implementation time or mistakes.

Result 2. *Adding a transition to a rule has a significant positive effect on its cost, but the effect is half as large as that of adding a state.*

The Absorbing indicator in Table 5 is large and statistically significant: changing a rule from absorbing to non-absorbing adds as much cost as adding a state. Indeed, as Table 4 and Figure 5 show, the three least costly rules are all absorbing rules. Absorption also blunts the impact of states on complexity costs: adding states to absorbing rules has a much smaller impact on complexity costs than does adding states to non-absorbing rules ($Absorbing \times State$ is significantly negative). Finally, because all transitions added to absorbing rules simultaneously remove the absorbing state in our design, the large significant effect of the interaction $Absorbing \times Transition$ tells us that adding a transition that simultaneously causes an absorbing state to become non-absorbing, has double the effect of adding a conventional transition. Absorbing rules have similarly sizable impacts on our other complexity measures.

Result 3. *Absorbing rules are significantly less costly than non-absorbing rules, equivalent to removing a state. Adding a state to an absorbing rule generates substantially less cost than adding one to a non-absorbing rule.*

To summarize, structural automoton characteristics collectively have strong effects on complexity costs: at the median, the most complex rule (according to Hypotheses 1-3) is six times more costly than the least complex rule. States and absorption (but not transitions) have qualitatively similar effects on implementation time and mistakes rates, our other two complexity measures.

[42] There is some evidence of diminishing marginal costs in states. We can show this by comparing the individual-level effect of increasing states from 2 to 3 to the effect of increasing from 3 to 4. The design allows exactly one such comparison in our dataset that isn't confounded with changes in transitions or absorption: the comparison between *2S-2T* and *3S-2T* versus *3S-2T* and *4S-2T*. Adding a state to a 2 state rule adds a mean cost of 0.288, while adding one to a 3 state rule adds a mean cost of 0.172 and the difference between these two marginal costs is statistically significant using a paired Wilcoxon test ($p < 0.01$).

[43] We find no evidence of diminishing marginal costs in transitions. As with states, we can test this with exactly one comparison in the dataset without introducing confounds: by comparing the difference between *4S-2T* and *4S-3T* to the difference between *4S-3T* and *4S-4T*. Adding a transition to a 2 transition rule adds a cost of 0.144, while adding one to a 3 transition rule adds 0.126 and the difference between these two marginal costs is not statistically significant using a paired Wilcoxon test ($p = 0.947$).

	Cost	Time	Mistakes
	(1)	(2)	(3)
States (**H1**, # of states in rule -2)	0.239***	8.788***	0.027**
	(0.021)	(1.083)	(0.012)
Transitions (**H2**, # of transitions in rule -1)	0.116***	−0.752	−0.003
	(0.019)	(1.357)	(0.013)
Absorbing? (**H3**, indicator absorbing state in rule)	−0.203***	−44.098***	−0.142***
	(0.039)	(1.959)	(0.020)
Reducible? (**H4**,indicator for rule 'reducible')	0.072**	−0.621	−0.100***
	(0.034)	(2.527)	(0.025)
Countable? (**H5**, indicator for rule 'countable')	−0.490***	−24.491***	−0.125***
	(0.048)	(2.254)	(0.024)
Sequenceable? (**H5**, indicator for rule 'sequenceable')	−0.117***	−5.128**	0.100***
	(0.041)	(2.371)	(0.030)
Recall? (indicator for Recall sessions)	0.020	0.359	0.005
	(0.114)	(1.548)	(0.017)
Reasoning? (indicator for Reasoning sessions)	0.301**	17.473***	0.015
	(0.150)	(4.147)	(0.020)
Run 1? (indicator for Run 1 of the experiment)	0.047	−3.062*	0.006
	(0.112)	(1.582)	(0.018)
Female? (indicator for gender, 1 = Female)	−0.021	−1.047	−0.016
	(0.105)	(2.035)	(0.016)
STEM? (indicator for STEM major)	−0.157	0.334	−0.006
	(0.106)	(2.340)	(0.015)
Cog Score (cognitive battery score)	−1.801***	−11.445*	−0.287***
	(0.337)	(6.897)	(0.048)
Absorbing X State	−0.140***	−1.037	−0.010
	(0.023)	(1.276)	(0.014)
Absorbing X Transition	0.131***	3.435**	0.003
	(0.027)	(1.456)	(0.014)
Constant	0.903***	56.560***	0.151***
	(0.123)	(2.973)	(0.027)
Observations	3,850	3,850	3,850
R^2	0.182	0.337	0.081
Residual Std. Error (df = 3835)	1.011	30.426	0.318
Note:			*p<0.1; **p<0.05; ***p<0.01

Table 5: Estimates of drivers of complexity metrics. *Notes: The dependent variables include cost (specification 1), implementation time (specification 2) and an indicator taking a value of 1 if the subject made a mistake (specification 3). Each specification clusters standard errors at the subject level. Right hand variables include number of states and transitions and a dummy for whether the rule includes an absorbing state. We subtract 2 from states and 1 from transitions (i.e. the minimal number of states and transitions in the design) in order to make the intercept easier to interpret. Reducible?, Countable? and Sequenceable? are dummy variables corresponding to the representation rules of the same name. Recall and Reasoning are dummies for the versions of the Implementation Task of the same name. Run 1 is a dummy variable for the first run of the experiment and Female and Stem are gender and college major dummies. Cognitive Battery is an index of scores from a battery of cognitive tests, ranging in value from a minimum of 0 to a maximum of 1 and demeaned.*

4.2 Findings on Representation

We next examine the hypothesis (Hypothesis 4) that decision makers efficiently remove unnecessary states from rules by representing them in their least state-complex form. Recall that the rule

reducible has identical states, transitions and transience to rule *4S-4T* but can be represented as (reduced to) the much simpler (2 state, 2 transition) rule *reduced*. We find no evidence that subjects do this: Figure 5 shows that the elicited cost of implementing rule *reducible* is in fact slightly *higher* than its state/transition equivalent *4S-4T* (and the *reducible* dummy is statistically significant and positive in our cost specification).[44] Furthermore, as is clear from Figure 5, *reducible* remains far more costly than the rule it can be reduced to, *reduced*.

Result 4. *Subjects do not economize on state complexity by reducing rules to simpler isomorphic finite state machine representations.*

Finally, we test the hypothesis that decision makers are constrained to remember events by tracking states, representing rules exclusively as finite state machines (FSMs). Our alternative hypothesis (Hypothesis 5) is that when it is possible to replace states with simple sequences of events in working memory, subjects do so and that this reduces complexity. Our key rule for examining this hypothesis is *countable* which is identical to the rule *4S-4T* when represented as an FSM, but can be represented as a 2-state, 4-transition rule if represented as a pushdown automaton (PDA) that can track elapsed events in working memory.

Figure 5 strongly supports Hypothesis 5: subjects treat the rule *countable* as substantially less complex than *4S-4T*. The *countable* variable in the cost specification of Table 5 is highly significant and quantiatively large. Indeed, the magnitude of this reduction is highly consistent with what we would expect if subjects represented the rule as a PDA rather than an FSM: removing 2 states and 0 transition from the rule (as reduction from a 4-state, 4-transition FSM to a 2-state, 4-transition PDA would dictate) should lead to a 0.472 reduction in cost (based on estimates from Table 5), which is nearly identical to our -0.49 estimate. Thus the cost reduction we observe for *countable* is exactly what we would expect if subjects used a PDA-like representation rather than an FSM-like representation. (Implementation time and mistakes rates are also significantly smaller for *countable* than for *4S-4T*.)

Result 5. *Rules that can be represented with fewer states (and no more transitions) by using working memory to track events are significantly less complex.*

A similar but (as expected) weaker pattern occurs with the rule *sequenceable* which can also be state-reduced using a PDA-like representation, though only by adding transitions in exchange for the removed states. Again, Figure 5 shows (and the *sequenceable* dummy in Table 5 confirms) that complexity costs and implementation time are smaller for this rule than for the reference rule 4S-4T. The cost drop is much smaller than the drop for *countable* (0.11 vs. 0.49) but as we argue in Section 2.2.2, we expect the effect to be much smaller here because the PDA representation requires several more transitions than the PDA representation of *countable*.[45,46]

[44] This is true, despite the fact that subjects make half as many mistakes in *reducible* as in *4S-4T*. This low mistake rate is likely a consequence of the fact that the pattern of actions required by *reducible* is relatively robust to errors in correctly perceiving and responding to events (Lipman & Srivastava 1990), a characteristic which is orthogonal to the fact that the rule is reducible. See Section 4.5 below for a discussion of this.

[45] The estimated cost reduction for *sequenceable* is, again, exactly what we would expect, given our estimates, from removing 2 states and adding 3 transitions from the rule (as would occur by transitioning to a PDA representation). At estimated costs of states for transitions, we should expect *sequenceable* to be about one transition less costly than a rule represented as a 4 state, 4 transition FSM and that is exactly the size of the *sequenceable* estimate.

[46] All of our three state rules (3S-1T, 3S-2T, 3S-3T) can also be reduced to two state rules using an event stack PDA, but only by adding transitions. Given our estimated exchange rates between states and transitions, this would not actually reduce complexity costs for any of these rules. For instance, 3S-3T can be transformed into a 2 state, 5 (net) transition rule by adding an event stack. But the two additional transitions would exactly offset the removal of a state given estimated costs. See Online Appendix C for details.

In Online Appendix C, we consider the merits of describing rules as pushdown automata rather than finite state machines in more depth. Our findings suggest that further study of what classes of algorithms (e.g. finite state machines, pushdown automata, Turing machines etc.) best describe and predict complexity is an important direction for future research.

4.3 Heterogeneity and Prevalence

We next consider heterogeneity of these effects across subjects. First, recall (Figure 4), that a cluster of subjects (about 17%) submit identical WTP numbers across *all* of the rules. Most of these are subjects who never adjusted their sliders at all, causing them to submit a default cost of 0 perhaps because they suffer no complexity costs (or are complexity seeking). For the remainder of subjects, we individually fit our main estimating equation from Table 5 for the determinants of cost and implementation time, and plot our key coefficients as scatter plots (one for each dependent variable) in Figure 6. We highlight several points.

First, as kernel density plots of the marginal distributions, plotted on the edges of these graphs, show, some of the cost effects we've documented occur for almost all of these subjects (almost 90% for s-complexity and countability), some occur for most of them (2/3 for transitions and absorption) and some for only half of them (reducibility and sequenceability).[47] Second, as the scatter plots show, the effects of rule characteristics on complexity costs and implementation time are generally uncorrelated within-subject or only very weakly correlated: in none of these cases is there a correlation coefficient greater than 0.16 and all but absorption are statistically insignificant. Thus knowing the size of the effect of e.g. states on implementation time for a subject provides little information on the size of its effect on the subject's complexity cost.

Finally, there is significant heterogeneity in the magnitude of each effect across subjects, meaning there is considerable unexplained subject-to-subject variation in *quantitative* effects on costs and implementation times in the data. We can quantify the heterogeneity of these effects by studying partial correlations between our main independent variables and our complexity metrics (cost and time).[48] The results of this analysis show, for example, that although states increase complexity costs for the great majority of subjects, the magnitude of the effect is heterogeneous enough that they predict only about 25% of residual variation in costs even after demeaning cost at the subject level. Smaller fractions of the variance are explained by countability (22%), absorption (14%), transitions (11%) and reducibility (3%).[49] Thus, although many of these characteristics impact complexity in a consistent direction for most subjects and have quantitatively large effects on average, the magnitude of their effects on individual responses are quite varied.

[47] Prevalence of effect also sometimes varies across measure: while states and counting impact both cost and time for a similar proportion of subjects, transitions and absorption have dramatically different effects on the two measures.

[48] The partial correlation of y and x is calculated by regressing each on a vector of other relevant variables $\mathbf{z}$ and then calculating the correlation coefficient between the residuals. It describes how much of the variation in y is explained by x after controlling for the explanatory contribution of other relevant variables. We calculate partial correlations on subject-wise demeaned costs and include variables for for all of our hypotheses by including state and transition counts, an absorption indicator and indicators for the *countable*, *sequenceable* and *reducible* rules. Since states and transitions in particular have very different characteristics under transient than absorbing rules, we calculated these and other non-absorbing effects using only data from transient rules.

[49] Corresponding explained variance (via partial correlation analysis) for implementation time include absorption (47%), countability (21%), states (18%), transitions (1%) and reducibility ($< 1\%$).

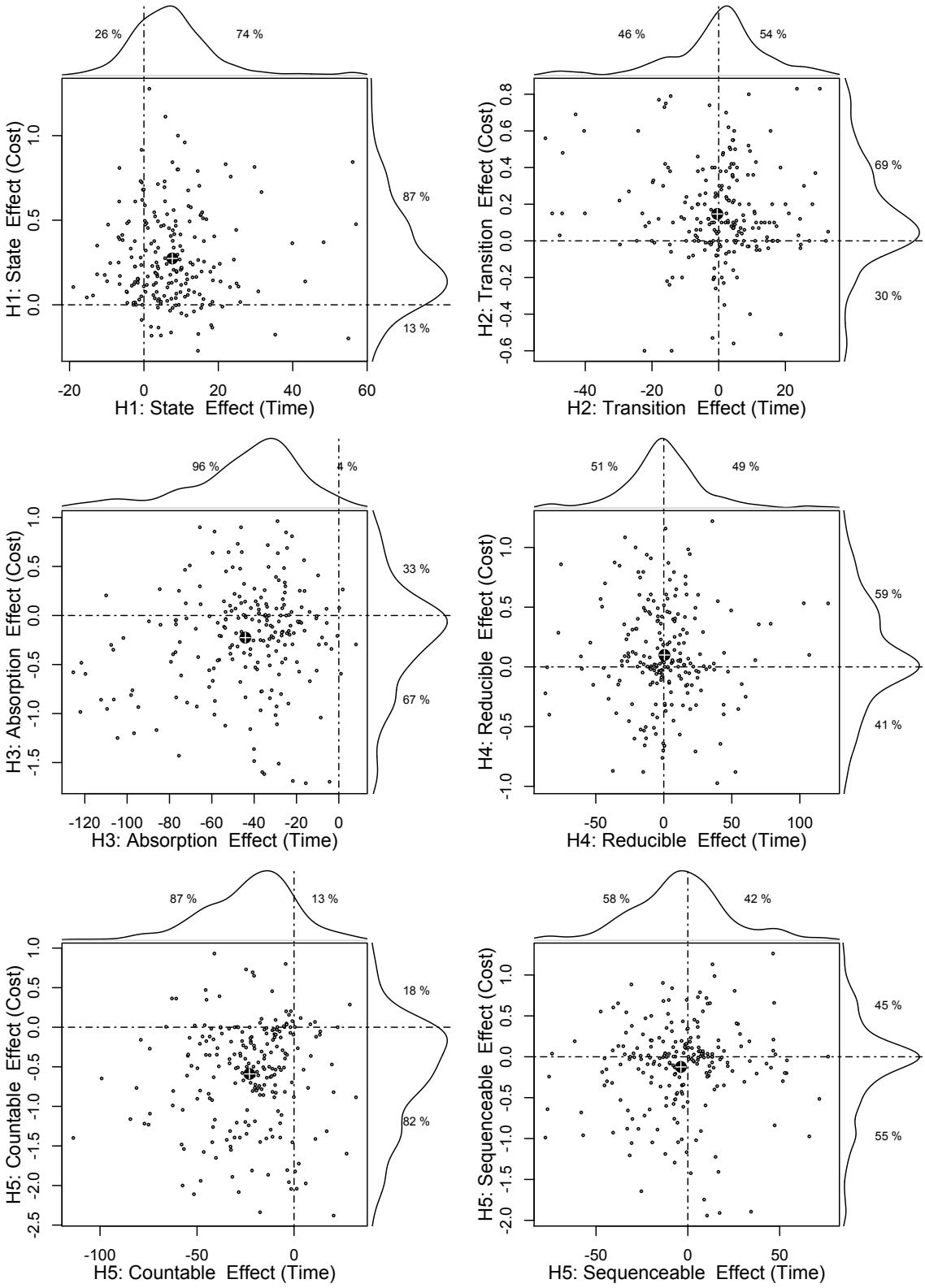


Figure 6: Individual regression effects. *Notes: Each panel shows a scatter plot (with implementation time on the x-axis and cost on the y-axis) of estimates from individual subject-level regressions using specifications (1) and (2) from Table 5. Labels give the name of the variable reported in the panel (matching variable names in Table 5) and H1,...H5 labels give the corresponding hypothesis numbers. On the edges of the graphs are plotted kernel density estimates of the marginal distributions of estimates for implementation time (x-axis) and cost (y-axis) Numbers on each side of zero on these density plots report the percentage of subjects with negative vs. positive estimated coefficients for the variable. We cut out 2.5% outliers from either side for scatterplot visibility.*

4.4 Learning, Familiarity and Context

Stages 3-5 of the experiment allow us to study the influence of *familiarity* on complexity directly by allowing subjects to procedurally learn rules via repetition. In Stage 3 (Run 2) we randomized half of our subjects into one of two maximally complex rules: *4S-4T* and *reducible* (the two most cost-complex rules in our data).[50] These subjects then implemented their assigned rule 10 times (each time in response to 20 events). Panel (a) of Figure 7 shows the evolution of mean implementation time over the 10 tasks of Stage 3. Implementation time falls substantially with experience, eventually falling to 66% of initial levels.[51,52]

Familiarity also reduces the complexity costs of implementing a rule. In Stage 4, we elicited subjects' willingness to pay to avoid *implementing the same rule they had previously implemented repeatedly in Stage 3* for a further 10 tasks (in Stage 5). In order to focus on the effect of learning and familiarity on preferences (rather than on the rate of mistakes), we restrict attention to subjects who made almost no mistakes to begin with in Stage 1.[53] The datapoint "Learn" in panel (b) of Figure 7 presents the change in cost between initial elicitation for this rule in Stage 2 and the re-elicitation in Stage 4. The effect is statistically significant (paired Wilcoxon test, $p < 0.001$, $N = 42$) and quite large, equivalent to stripping more than two states from the rule.[54]

These familiarity and procedural learning effects are strong but they are also quite superficial and specific to details of the exact rule learned. In Stage 4, we also elicited costs for a very slight perturbation of the rule implemented in Stage 3. In the perturbation, we replaced each possible choice and event in the rule (for example the letters 'a' and 'x') with a random set of letters (for example 'a' is replaced with 'b' and 'x' is replaced with 'y' throughout). Structurally (and under the lens of our Hypotheses) this "perturbed rule" is identical but it has been superficially – almost cosmetically – altered. The datapoint "Transfer" in panel (b) of Figure 7 presents the change in cost (relative to the original elicitation of the non-perturbed rule in Stage 2) using the exact same subjects shown for the datapoint "Learn." There is still a significant reduction in cost (paired Wilcoxon test, $p = 0.025$, $N = 42$), but the size of the effect is half that observed for the original rule ($p < 0.001$, $N = 42$).

In Stage 5, we similarly examined whether the effects of learning on *completion time* transfer to superficial perturbations of the learned rule by randomly assigning some subjects to implement the original rule and another set to implement the perturbed rule for 10 tasks. In panel (c) of Figure 7 we plot a time series of implementation times for both the original rule and its perturbation. While the implementation time of the original rule continues its Stage 3 trajectory downward with only a

[50] We focus on Run 2 here because the Run 2 experimental procedure produces more homogeneous data (in Run 1 we randomized assignment across all 14 rules instead of just the two most complex) and features rules that are, at baseline, more complex, giving us a much clearer window into the potential effects of learning and familiarity. If we pool with Run 1 data and re-conduct the analysis to follow, the results are qualitatively unchanged.

[51] This reduction is statistically significant by a paired Wilcoxon test ($p < 0.001$, $N = 84$). By contrast, mistakes rates are low throughout and change very little.

[52] Subjects learn faster under the rule *reducible* than under rule *4S-4T*, dropping 37 seconds rather than 22 seconds and this difference is statistically significant ($p = 0.003$, Wilcoxon test). This may suggest that some subjects can learn to reduce rules (Hypothesis 4) to some degree with experience.

[53] The results are similar but attenuated if we use the entire sample instead: using the entire sample, the effect of learning on costs for the original rule (perturbed rule) is -0.371 (-0.169) instead of -0.479 (-0.25). Wilcoxon tests indicate an effect significantly different from zero in all cases.

[54] The cost reduction in *reducible* is no larger than in 4S-4T ($p = 0.731$). Thus, although implementation time falls faster with experience for reducible than non-reducible rules, subjects' distaste for reducible rules does not fall to any greater degree.

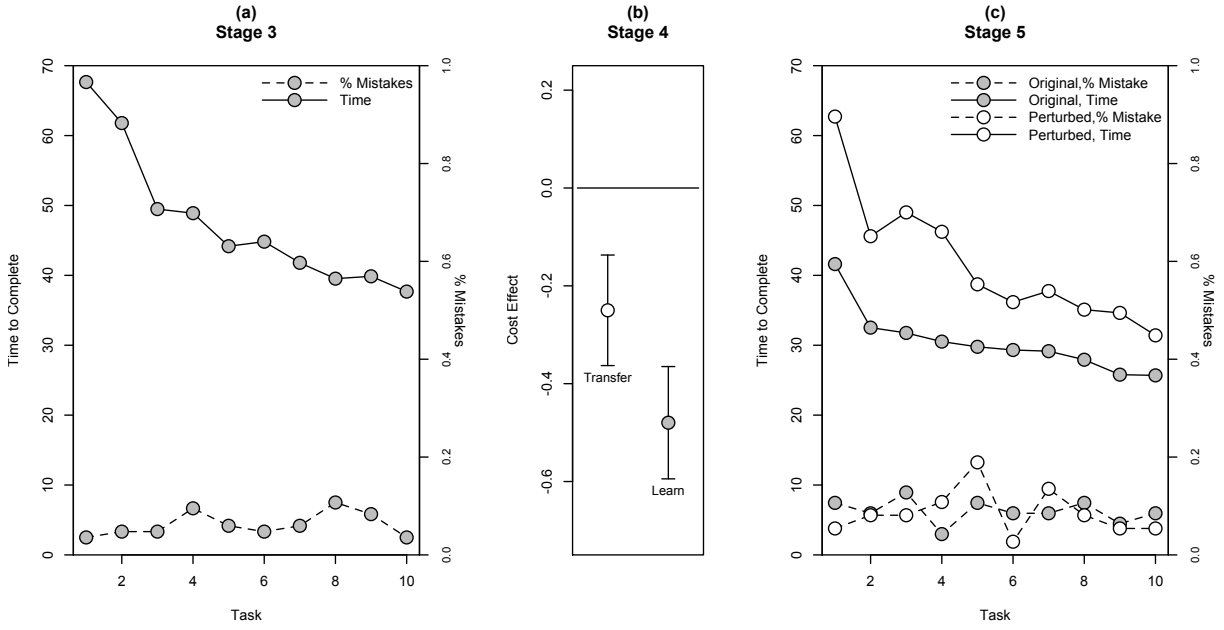


Figure 7: Learning and familiarity effects. *Notes: Panels for Stage 3 and Stage 5 present mistakes rates (dashed lines) and mean implementation times (solid lines) over 10 Implementation Tasks (each task is a separate Implementation Task consisting of 20 events). The panel for Stage 4 shows the mean difference (with standard error bars) between Stage 2 cost elicitation and Stage 4 cost elicitation for the rule implemented in Stage 3 (Learn) and a perturbation of this rule (Transfer).*

slight restart effect, *nearly all of the implementation time reduction* achieved in Stage 3 disappears for the perturbed rule: subjects must start the learning process almost entirely over again.[55]

Result 6. *Familiarity resulting from procedural learning substantially reduces complexity – the effect is equivalent to removing two states from the rule. However, this effect is extremely rule-specific and the impact of familiarity on complexity does not fully transfer to even slight perturbations of the learned rule.*

The *context* in which a rule is implemented also has effects on complexity. In our design we varied the context of our Implementation Tasks (without changing the algorithmic structure of the rules assigned to subjects) in such a way as to vary whether subjects had to perform the rule mentally (the Reasoning variation) or remember past events (Recall) to implement the rule. The estimates in Table 5 include indicator variables Recall and Reasoning to measure main effects of these variations. The results reveal that subjects find implementing a rule mentally to form a belief about the correct final choice prescribed by the rule (the Reasoning variation) considerably more costly than implementing the full rule via a sequence of actual actions (Base). The size of the effect is large, equivalent to adding more than one state to the rule, on average. Reasoning through a rule is also considerably slower but does not produce more mistakes. By contrast, having to remember recent history to perform the rule (as subjects must do in the Recall version) seems to have no significant effect on any measures of complexity.

[55]Similar difficulties with "learning transfer" have been documented in many economics and psychology experiment. See Cooper & Kagel (2010) for a recent example in economics and a review of the literature in economics and psychology.

Result 7. *Reasoning through a rule is considerably more costly and time-intensive than enacting a rule. The effect is equivalent to adding more than one state to the rule, on average.*

Interestingly, although the Reasoning variation produces an increase in average cost, it does not much alter the way automata characteristics influence these costs. Indeed, interacting indicator varables for "Recall" or "Reasoning" with the regressors in Table 5 produces no significant interactions and plotting (a'la Figure 5) costs separately for each variation shows essentially the same cost function (with respect to e.g. states or transitions) across variations. (The only exception is absorption: costs across variations are similar for absorbing rules, but non-absorbing rules are are much costlier under Reasoning than under the other variations.[56]) Our results thus suggest that the algorithmic drivers of complexity costs we identify are remarkably robust to variations in context.

4.5 Complexity Costs, Implementation Time and Mistakes

How does complexity cost relate to other metrics of complexity, particularly implementation time (a common complexity metric in economics experiments, see e.g. Gill & Prowse (2018))? Figure 8 summarizes our findings by plotting the effects (as a proportion of the average) of hypothesized drivers of complexity on costs vs. time (most from estimates in Table 5). Clearly cost and implementation time are shaped by similar forces: the drivers of each are strikingly similar in Figure 8, with proportional effects that are mostly linearly related on the 45-degree line. However, as individual level data visualized in Figure 4 and 6 make clear, the two are actually only weakly related at the individual level: both individual averages and coefficients are insignificantly correlated across these measures. Indeed including time in specification (1) of Table 5 (see Online Appendix B) reveals only very weak explanatory power of a rule's implementation time on cost at the subject level. We conclude that although complexity costs and implementation time are driven in the aggregate by very similar forces (both quantitatively and qualitatively), these are relatively independent effects, underscoring the likely importance of measuring complexity costs directly.[57]

By contrast, even in the aggregate, the drivers of costs and *mistakes* are only very weakly related, as Figure 5 and Table 4 show. For instance, the rule *reducible* and its equivalent *reduced* have wildly different costs and implementation times attached to them but they are almost identical in terms of mistakes. The reason for this is that, as Lipman & Srivastava (1990) show, automata vary dramatically in their *robustness* to mistakes in accurately tracking the sequence of events and *reducible* and *reduced* are identically robust in this sense. By contrast, the rules 2S-2T and *reduced* have identical states and transitions and are treated as equally costly by subjects, but the former is far more forgiving of mistakes in tracking. As a result subjects make mistakes twice as often in *2S-2T* as they do in *reduced* (and in the equivalently robust rule *reducible*).[58,59] Subjects clearly do not take mistakes into account in a sophisticated way when assessing cost (even after having

[56]The interaction between Reasoning and the Absorbing dummy is -0.361 ($p = 0.006$).

[57]This finding is surprising because it is intuitive to think of complexity costs as arising from per-second costs of time spent enacting a rule. One possibility is that there is a much stronger latent relationship between time and cost that is attenuated by noisy measurement of each in our experiment. Another possibility is that subjects are averse to complex rules in large part because of fixed cognitive costs (e.g. the costs of absorbing and thinking through the structure of the rule) that have relatively little to do with the length of time they will have to spend actually implementing the rule.

[58]When we add an index of robustness proposed in Lipman & Srivastava (1990) to our cost regression, we find it is insignificantly different from zero and has no significant impact on other estimates.

[59]Likewise, subjects treat the rule *sequenceable* (which can be represented with fewer states using a memory stack) as less costly than similar state/transition rules (e.g. *4S-4T*) but make mistakes twice as often.

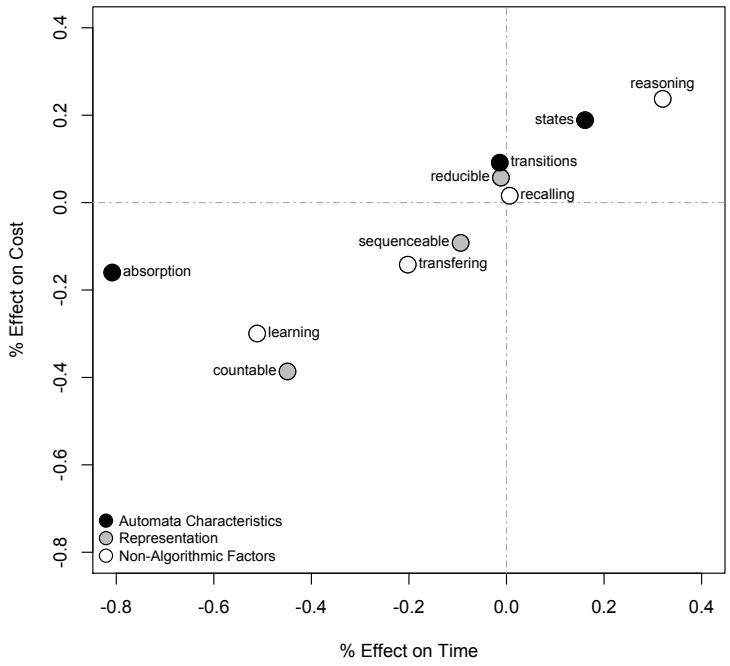


Figure 8: Estimates of effects on cost complexity and implementation time. *Notes: Most estimates are taken from (and match in name) coefficient estimates from Table 5, and are reported as a fraction of mean cost/implementation time. Learning and transfering measures are taken from treatment effects reported in Section 4.4.*

experienced each of these rules!), and so do not attach low costs to low-mistake rules like *reduced* and *reducible* or high costs to high-mistake rules like *sequenceable*.

This lack of association between costs and mistakes holds more generally and this is important for interpreting our cost estimates. Ex ante, WTP to avoid complexity might be driven either by (i) sheer subjective distaste for complexity or (ii) fear of making future mistakes (and thereby sacrificing expected earnings). In Online Appendix B we provide evidence that, in fact, the drivers of complexity costs we have identified are overwhelmingly driven by the former (distaste for complexity). Repeating our cost analysis on subjects who make very few mistakes (and therefore are unlikely to pay to avoid mistakes in the elicitation) or including mistakes explicitly as a dependent variable generates almost identical parameter estimates to those reported in specification (1) of Table 5. This, in turn, suggests that most of the cost-generating effects (e.g. s-complexity, t-complexity etc.) measured in the experiment are driven by subjects' distaste for implementing complex features of rules.[60]

Our design purposely minimizes the relationships among these complexity metrics, but in many realistic settings they are likely to be more entangled. For instance our design minimizes the opportunity cost of implementation time, but in many natural settings we expect such costs to

[60]Mistakes rates are low enough in our experiment that it is relatively easy to separate the contribution of distaste for complexity from fear of making mistakes. Extending this research strategy to more difficult tasks may require iterations on these methods that separately elicit (i) beliefs about the likelihood of mistakes and (ii) willingness to pay to avoid the task in the future, making it possible to separate distaste for the task from fear of payoff consequences from making mistakes directly.

contribute significantly to the cost of complexity. Likewise, our subjects are not time constrained, but in many realistic settings time constraints will surely produce a higher rate of mistakes for complex rules. Finally, our incentives are strong enough to motivate subjects to (mostly) properly implement rules, but in many settings this won't be true (either because of higher complexity costs or lower incentives), leading to a greater incidence of apparent "mistakes."

5 Applications

In the introduction we emphasized that procedural complexity has potential applications across economics, and we close the paper by illustrating how evidence like ours might be useful for interpreting and predicting behavior in applications. To do this, we fit our complexity cost estimates to the automaton characteristics (number of states, transitions etc.) of some important rules in economics (rational choice procedures and repeated game strategies), and relate these calibrations to observations from prior empirical work. These are, of course, ex post exercises but our purpose in doing this is primarily to hint at the potential uses of our methods for future, prospective research.

Rational Choice Procedures. Rational choice is a core assumption in economics, but there is a long tradition beginning with Simon (1955) arguing that rational choice is too procedurally complex to be a realistic description of human behavior. When choosing between options (e.g. in a list), it is much simpler to instead *satisfice* by setting a reservation value ("aspiration level") and choosing the first option that meets or exceeds it (instead of exhaustively considering all options as is required by rational choice). Salant (2011) uses an automata model to show that even in very basic settings rational choice is far more procedurally complex than satisficing: procedures that select the optimal (most valuable) option from a list of N options are extremely s-complex, requiring $N-1$ states, while satisficing requires only 1 state regardless of N. Satisficing and related choice procedures are simple, but they lead to suboptimal choices, producing classic behavioral phenomena like recency and primacy effects, status quo bias and choice overload.

We calibrate the cost of rational choice using Salant's characterization for lists containing N =10, 20 and 40 options and plot the results in the left panel of Figure 9. Salant focuses on the consequences of assuming that states drive complexity: following this and using our estimated cost of states (and including the penalty from implementing a rule mentally reported in Result 7 and the intercept) we get complexity cost estimates of using rational choice procedures that range from just under \$3 for 10 option lists to around \$10 for 40 option lists. Moreover, our results suggest this is probably an underestimate: if we add the cost of the $(N+1)N/2$ transitions required to implement rational choice, the complexity cost rises to as high as nearly \$80 for a 40 option list!

How might we use complexity calibrations like these to predict and interpret behavior? We illustrate using an experimental design reported in Caplin et al. (2011) in which subjects evaluate and compare options from a list and make a choice after consideration. Their setting is useful for this exercise because the options their subjects choose between have objective monetary values: subjects are shown the monetary payoff attached to each object in an opaque fashion (as a list of numbers to be added together) meaning some effort and memory is required to evaluate and compare. Caplin et al. (2011) vary the size of their option lists between 10, 20 and 40 in their design and we can use the lists employed in the experiment to calculate, for comparison, the opportunity cost of using a satisficing procedure (the expected money sacrificed by satisficing instead of using rational choice)

for a range of possible satisficing reservation values.[61] We plot these expected satisficing costs for reservation values 5, 10, 15, 20 and 25, shown from left to right as a dotted line (for each list length) on the plot.

Even for the smallest lists, the complexity costs of rational choice dwarf the expected payoffs sacrificed by satisficing and so we would expect satisficing to be common (and rational choice rare) in all three cases based on procedural complexity alone. Indeed, Caplin et al. (2011) find strong evidence of satisficing for all three list sizes, with subjects regularly choosing payoff dominated options and, further, displaying distinctive order effects that indicate satisficing (i.e. when the highest value item is lower in the list, subjects make more mistakes). Moreover, choice process data that tracks subjects' provisional choices show distinctive evidence of satisficing.

Our calibration suggests that rational choice may be procedurally complex enough to induce high rates of satisficing, but we emphasize that careful prospective experiments are required to convincingly identify procedural complexity as an explanation and especially to separate its role from that of other candidate explanations (such as e.g. optimal search behavior).[62] For example, experiments that correlate individual procedural complexity costs with satisficing or that systematically vary the opportunity cost of satisficing relative to the complexity costs of rational choice would provide much stronger evidence. Developing synthetic choice environments that vary procedural complexity costs without altering the predictions of alternative models like rational search would be particularly valuable (see Footnote 66, below, for an example). More generally, using these research strategies to study the capacity of procedural complexity to explain and tie together other key regularities from behavioral economics such as random choice (Kalai & Solan 2003), non-Bayesian inference (Chauvin 2020), biases in information processing (Wilson 2014), failures of backwards induction (Neyman 1985) and perhaps others such as myopia in intertemporal choice, failures to contingently reason through decision trees, status quo bias etc. is an important next step in this research.

This exercise also illustrates some of the subtleties involved in deploying automata models in practice and points towards potentially important extensions of these models. One important finding from Caplin et al. (2011) is that the effort required to evaluate individual items (in their experiment the number of numbers that must be summed to determine an item's value) has an influence on mistakes and rates of satisficing. A natural way of understanding this in the context of automata models, is that in some settings evaluating *events* can sometimes be costly: each time subjects in Caplin et al. (2011) evaluate a new item (each time they make a transition, in the language of Salant (2011)), they must perform a computation and as these computations grow costly, the costs of implementing elaborate procedures (like rational choice) must grow. Such costs are set extremely low in our experiment, where subjects immediately (with low required cognitive effort) know what event has occurred and thus how to transition behavior in response. If our rules had required subjects to add 3 or 7 numbers to determine which letter-event had occurred, our cost estimates would have been more comparable to those faced by subjects in Caplin et al. (2011), and we would expect subjects to be more strongly averse to rules that tend to require more event-processing and transitions. From this perspective our cost calibrations probably understate the procedural costs faced by subjects in Caplin et al. (2011). As Salant (2011) emphasizes, understanding how the "marginal" complexity costs of evaluating each option interacts with the "fixed" costs of the decision procedure itself (for

[61] Note that satisficing has only one state and so has no complexity costs given Salant's model and our framework.

[62] Caplin et al. (2011) find that comparative statics from search models incompletely organize differences in behavior between their treatments, concluding "our subjects may be satisficing for the reasons that Simon (1955) originally proposed, as a rule of thumb that performs adequately across a broad range of environments, rather than finely honing their search strategy to each choice environment they face." This procedural interpretation may suggest that procedural complexity might have an important role in producing satisficing behavior.

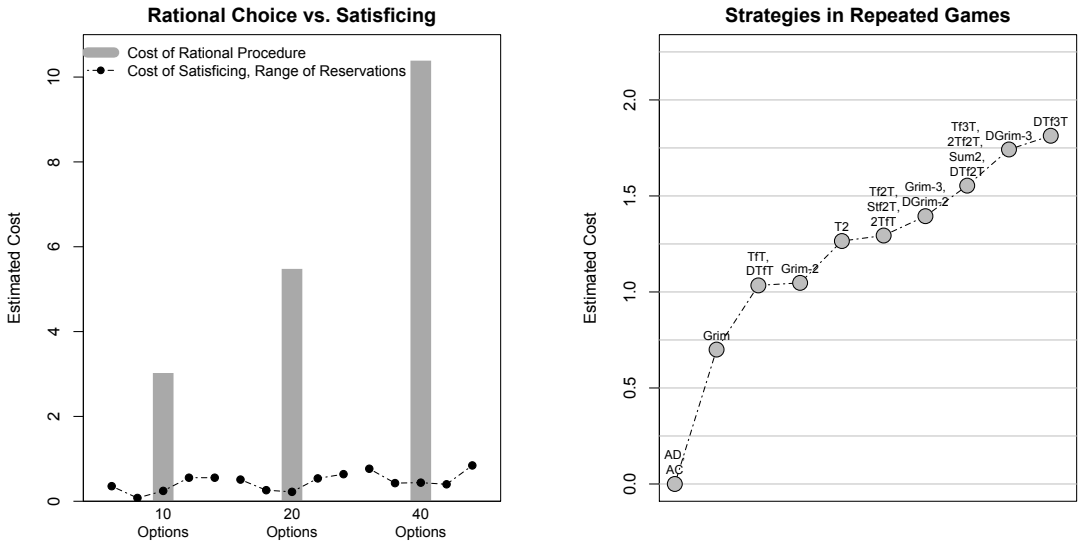


Figure 9: Two examples of calibrated complexity costs. *Notes: The left panel calibrates complexity costs (gray bars) of rationally choosing the highest value item from lists of options of size 10, 20 and 40. For reference, in each case, we plot (dotted lines) the opportunity cost of satisficing (from Caplin et al. (2011)) with a range of reservation values (5, 10, 15, 20 and 25). The right hand panel calibrates complexity costs for a set of strategies for repeated prisoner's dilemmas.*

instance the number of states in the procedure) is an important task for future work.

Repeated Game Strategies. Repeated games arise across economics, but the theory suffers from a fundamental indeterminacy, described by the folk theorem: with low enough discount rates, a wide variety of outcomes can generally be supported if players are willing to use sufficiently complex punishment strategies. The automata literature began as an attempt to study how aversion to complex strategies might reduce this indeterminacy, making game theory more predictive.

In the right panel of Figure 9 we calibrate the costs of a number of strategies from repeated prisoner's dilemmas.[63] Specifically, we calibrate complexity costs using our estimates of the costs of states, transition and absorption for most (18) of the strategies discussed and estimated in Fudenberg et al. (2012).[64] The results reveal a wide range of complexity costs, with the least complex non-trivial strategy (Grim) costing 1/3 as much as the most complex (DTf3T).

Based on these cost estimates alone (i.e. without taking account of the benefits of strategies or important issues of equilibrium and strategic risk), several predictions about the use of these strategies naturally arise. For one thing, ceteris paribus, we would expect the "cheapest" strategies – AC (always cooperate), AD (always defect), Grim (the grim trigger strategy), TfT (tit for tat) and D-TfT (tit for tat beginning with defection) – to be particularly common and for our most

[63]Here AD and AC are "always defect" and "always cooperate," Grim-N is the Grim rule from the introduction with N periods of counterpart defection preceding own defection. Tit-for-tat specifies cooperating and then defecting after counterpart defection and returning to cooperation the next round if the counterpart does not defect again. (M)Tf(N)T is tit-for-tat that moves from cooperation only after (N) rounds of counterpart defection and then remains in defection for (M) rounds before returning if counterpart is cooperative. Rules that begin with a 'D' add an initial period of defection to the beginning of the rule. T2 is a strategy that defects twice in response to a defection.

[64]For instance, the rule "Grim 3" – cooperate until your counterpart defects three times, then defect forever – has four states, three transitions and an absorbing state. Using estimates from specification (1) in Table 5, we estimate a cost by adding our intercept 0.903 (which estimates the baseline cost of a 2-state, 1 transition, transient rule), 2×(0.239-0.14) (the cost of states plus the interaction with an absorbing state) and 2×(0.116+0.131) (transitions plus the interaction with an absorbing state) and subtracting 0.203 (because of the absorbing state) we get a cost of $1.40.

"expensive" strategies (e.g. DGrim3, DTf3T, 2Tf2T) to be relatively rare. For another, because there are significant costs attached to implementing strategies more complex than the extremely simple (one-shot Nash) strategy, AD, we should expect to see subjects abandoning this simplest strategy exactly when the parameters of the game increase the potential payoffs from avoiding a mutual defection outcome (a moderate payoff outcome that subjects can guarantee for themselves using a maximally simple rule).

Indeed, both of these patterns are typical in the experimental literature. For instance Dal Bó & Fréchette (2018) review the literature and calculate that our estimated five 'cheapest" strategies (AC, AD, Grim, TfT and D-TFT) account for the majority of strategies in almost all papers that estimate strategies, and that they account for more than 3/4 of strategies in the majority of studies (in fact AD, Grim and TfT alone account for more than 70% of strategies in the majority of papers/treatments). Dal Bó & Fréchette (2011) and Dal Bó & Fréchette (2019) systematically vary the cooperation payoffs and the discount rate (which together determine the potential benefits of using more complex strategies than AD) and find subjects are, indeed, much more likely to abandon AD for richer strategies when either of these parameters rise. In some settings (e.g. prisoner's dilemmas with randomly perturbed actions) the benefits of cooperation are difficult to achieve in any other way than to use relatively complex strategies and it is only in these settings that the literature documents any significant use of moderately complex, costly strategies like Grim-3 and Tf3T (rather than their simpler cousins Grim and TfT)(Fudenberg et al. 2012).[65] Under the lens of our findings, it seems that subjects actively avoid complex rules when possible, deploying them only when the gains to doing so are sufficiently high.

Again, we emphasize that further research is needed before we can draw firm conclusions on the role of procedural complexity in strategy selection. This is particularly important in rich settings like repeated games where equilibrium and strategic uncertainty considerations cross with complexity costs in surely complicated ways. New experiments that correlate individual complexity cost measures with strategy choice and especially novel designs built on environments capable of separating the role of procedural complexity from that of other key forces will be important for this task.[66]

6 Conclusion

We use a new type of experiment to identify the algorithmic characteristics of rules that make them complex and therefore costly to implement. We find evidence of significant structure in the determinants of complexity, with a rule's algorithmic characteristics (particularly the number of states required to implement it) significantly influencing its complexity costs. We also find evidence that the way a rule can be represented (for instance whether it can be simplified using counting),

[65] Fudenberg et al. (2012) report experimental prisoner's dilemmas run both with and without noisy implementation of actions (i.e. with and without a small probability the computer alters the subject's actions), under identical payoff parameters. They too find relatively low usage of complex strategies without noisy implementation, but find that with noisy implementation a majority of subjects use strategies more complex than TfT. Simpler strategies in this setting can produce costly punishment cycles (based purely on random events) that more complex strategies allow the subject to avoid.

[66] An example of the type of design that might be expanded upon for this task is Jones (2014) who artificially alters prisoner's dilemmas in such a way as to increase the state complexity of any cooperative strategy. Mirroring our findings in support of state complexity, Jones (2014) finds that, indeed, increasing the state complexity of cooperative strategies reduces the rate of cooperation observed in prisoner's dilemmas. Taking a similar approach in other games and choice problems and relating the results to the direct measurement techniques we introduce here seems like a promising template for future procedural complexity research.

familiarity with a rule acquired via learning, and the context in which a rule must be implemented (e.g. physically vs. mentally) have additional influence on complexity costs.

Our methods and iterations on them may allow us to use procedural complexity as a way of explaining, modeling and predicting departures from standard economics predictions documented in behavioral and institutional economics. Testing the ability of these types of measurements to explain behavioral anomalies, equilibrium selection and policy failures – both through new diagnostic experimental designs and correlational studies – is an important next step in developing this research. Furthermore, careful work examining to what degree richer algorithmic descriptions of rules might improve our ability to predict and explain complexity and its costs seems particularly important.[67] Finally, although we have applied our methods to the complexity cost of rules, in principle they may be useful for measuring the costs of other types of behavior and thus may have a much broader range of application.

References

Abeler, J. & Jäger, S. (2015), 'Complex Tax Incentives', *American Economic Journal: Economic Policy* **7**(3), 1–28.

Abreu, D. & Rubinstein, A. (1988), 'The Structure of Nash Equilibrium in Repeated Games with Finite Automata', *Econometrica* **56**(6), 1259–1281.

Alaoui, L. & Penta, A. (2018), 'Cost-Benefit Analysis in Reasoning'.

Banks, J. S. & Sundaram, R. K. (1990), 'Repeated Games, Finite Automata, and Complexity', *Games and Economic Behavior* **2**(2), 97–117.

Bossaerts, P. & Murawski, C. (2017), 'Computational Complexity and Human Decision-Making', *Trends in Cognitive Sciences* **21**(12), 917–929.

Campbell, D. J. (1988), 'Task Complexity: A Review and Analysis', *The Academy of Management Review* **13**, 40–52.

Caplin, A. (2016), 'Measuring and Modeling Attention', *Annual Review of Economics* **8**, 379–403.

Caplin, A., Csaba, D., Leahy, J. & Nov, O. (2019), 'Rational Inattention, Competitive Supply, and Psychometrics'.

Caplin, A., Dean, M. & Martin, D. (2011), 'Search and Satisficing', *American Economic Review* **101**(7), 2899–2922.

Cason, T. N. & Plott, C. (2014), 'Misconceptions and Game Form Recognition: Challenges to Theories of Revealed Preferences and Framing', *Journal of Political Economy* **122**, 1235–1270.

Chatterjee, K. & Sabourian, H. (2000), 'Multiperson Bargaining and Strategic Complexity', *Econometrica* **68**(6), 1491–1509.

Chatterjee, K. & Sabourian, H. (2009), 'Game Theory and Strategic Complexity', *Encyclopedia of Complexity and Systems Science* pp. 4098–4114.

[67] For instance, we provide some preliminary evidence that pushdown automata – an iteration on the finite state machines economists usually study – might be a more empirically useful way of taxonomizing rules.

Chauvin, K. P. (2020), 'Euclidean Properties of Bayesian Updating'.

Cooper, D. J. & Kagel, J. H. (2010), 'The Role of Context and Team Play in Cross-Game Learning', *Journal of the European Economic Association* **7**(5), 1101–1139.

Cosmides, L. (1987), 'The Logic of Social Exchange: Has Natural Selection Shaped How Humans Reason? Studies with the Wason Selection Task', *Cognition* **31**, 187–276.

Dal Bó, P. & Fréchette, G. R. (2011), 'The Evolution of Cooperation in Infinitely Repeated Games: Experimental Evidence', *American Economic Review* **101**(1), 411–429.

Dal Bó, P. & Fréchette, G. R. (2018), 'On the Determinants of Cooperation in Infinitely Repeated Games: A Survey', *Journal of Economic Literature* **56**(1), 60–114.

Dal Bó, P. & Fréchette, G. R. (2019), 'Strategy Choice In The Infinitely Repeated Prisoners Dilemma', *American Economic Review* p. forthcoming.

Dean, M. & Neligh, N. (2019), 'Experimental Tests of Rational Inattention'.

Dewam, A. & Neligh, N. (2020), 'Estimation Information Cost Functions in Models of Rational Inattention', *Journal of Economic Theory* **187**.

Echenique, F., Golovin, D. & Wierman, A. (2011), 'Complexity and Economics: Computational Constraints May Not Matter Empirically', *ACM SIGecom Exchanges* **10**(1), 2–5.

Ergin, H. & Sarver, T. (2010), 'A Unique Costly Contemplation Representation', *Econometrica* **78**(4), 1285–1339.

Fershtman, C. & Kalai, E. (1993), 'Complexity Considerations and Market Behavior', *The RAND Journal of Economics* **24**(2), 224–235.

Fudenberg, D., Rand, D. & Dreber, A. (2012), 'Slow to Anger and Fast to Forgive: Cooperation in an Uncertain World', *American Economic Review* **102**(2), 720–749.

Gale, D. & Sabourian, H. (2005), 'Complexity and Competition', *Econometrica* **73**(3), 739–769.

Gill, D. & Prowse, V. L. (2018), 'Strategic Complexity and the Value of Thinking'.

Greiner, B. (2015), 'Subject Pool Recruitment Procedures: Organizing Experiments with orsee', *Journal of the Economic Science Association* **1**(1), 114–125.

Halevy, Y. & Mayraz, G. (2020), 'Modes of Rationality: Act versus Rule-Based Decisions'.

Hopcroft, J. E. & Ullman, J. D. (1979), *Introduction to Automata Theory: Languages and Computation*, Addison-Wesley.

Jones, M. T. (2014), 'Strategic Complexity and Cooperation: An Experimental Study', *Journal of Economic Behavior & Organization* **106**, 352–366.

Kalai, E. (1990), 'Bounded Rationality and Strategic Complexity in Repeated Games', *Game Theory and Applications* pp. 131–157.

Kalai, E. & Solan, E. (2003), 'Randomization and Simplification in Dynamic Decision-Making', *Journal of Economic Theory* **111**(2), 251–264.

Kalai, E. & Stanford, W. (1988), 'Finite Rationality and Interpersonal Complexity in Repeated Games', *Econometrica* **56**(2), 397–410.

Kool, W. & Botvinick, M. (2018), 'Mental Labour', *Nature Human Behavior* **2**, 899–908.

Lipman, B. L. & Srivastava, S. (1990), 'Informational Requirements and Strategic Complexity in Repeated Games', *Games and Economic Behavior* **2**(3), 273–290.

Liu, P. & Li, Z. (2012), 'Task Complexity: A Review and Conceptualization Framework', *International Journal of Industrial Ergonomics* **42**(6), 553–568.

Murawski, C. & Bossaerts, P. (2016), 'How Humans Solve Complex Problems: The Case of the Knapsack Problem', *Scientific reports* **6**, 34851.

Neyman, A. (1985), 'Bounded Complexity Justifies Cooperation in the Finitely Repeated Prisoners' Dilemma', *Economics Letters* **19**(3), 227–229.

Nielsen, K. & Rehbeck, J. (2020), 'When Choices are Mistakes'.

Ortoleva, P. (2013), 'The Price of Flexibility: Towards A Theory of Thinking Aversion', *Journal of Economic Theory* **148**(3), 903–934.

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

Sabourian, H. (2004), 'Bargaining and Markets: Complexity and the Competitive Outcome', *Journal of Economic Theory* **116**(2), 189–228.

Salant, Y. (2011), 'Procedural Analysis of Choice Rules with Applications to Bounded Rationality', *American Economic Review* **101**(2), 724–748.

Shenhav, A., Musslick, S., Lieder, F., Kool, W., Griffiths, T. L., Cohen, J. D. & Botvinick, M. M. (2017), 'Toward a Rational and Mechanistic Account of Mental Effort', *Annual Review of Neuroscience* **40**, 99–124.

Sherstyuk, K. (2011), 'Complexity and Bidder Behavior in Iterative Auctions', *Economics Bulletin* **31**(4), 2769–2776.

Simon, H. A. (1955), 'A Behavioral Model of Rational Choice', *The Quarterly Journal of Economics* **69**(1), 99–118.

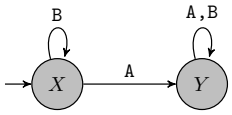
Sims, C. A. (2003), 'Implications of Rational Inattention', *Journal of Monetary Economics* **50**(3), 665–690.

Wilson, A. (2014), 'Bounded Memory and Biases in Information Processing', *Econometrica* **82**(6), 2257–2294.

Online Appendices

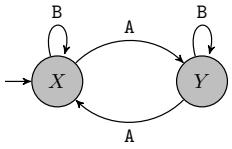
A Rules Studied in the Experiment

2S-1Ta



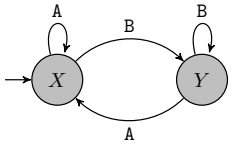
Choose x until you see a, after which switch to y for the rest of the round.

2S-2T



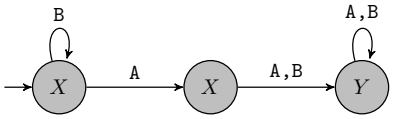
Choose x until you see a, after which switch to y. Then, start over after you see a.

reduced



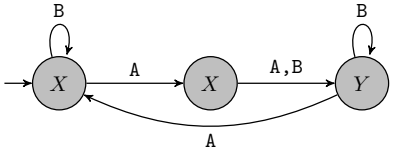
Choose x until you see b, after which switch to y. Then, start over after you see a.

3S-1Ta



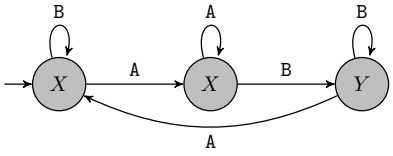
Choose x until you see a, after which choose x once more and then switch to y for the rest of the round.

3S-2T



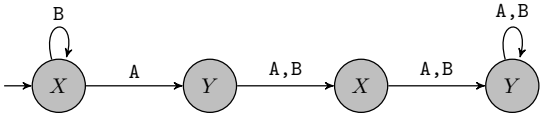
Choose x until you see a, after which choose x once more and then switch to y. Then, start over after you see a.

3S-3T



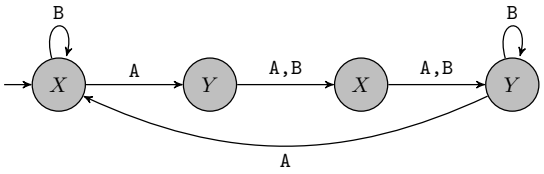
Choose x until you see a followed at some point by b, after which switch to y. Then, start over after you see a.

4S-1Ta



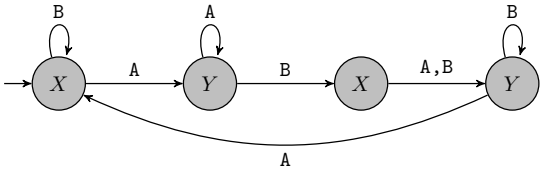
Choose x until you see a, after which choose y once followed by x once and then switch to y for the rest of the round.

4S-2T



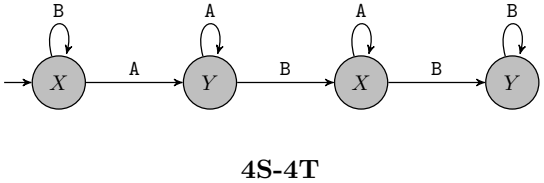
Choose x until you see a, after which choose y once followed by x once and then switch to y. Then, start over after you see a.

4S-3T



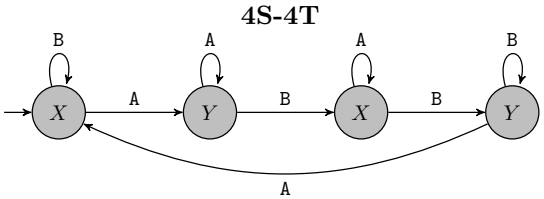
Choose x until you see a, after which switch to y. Then, after you see b, choose x once and then switch to y. Then, start over after you see a.

4S-3Ta



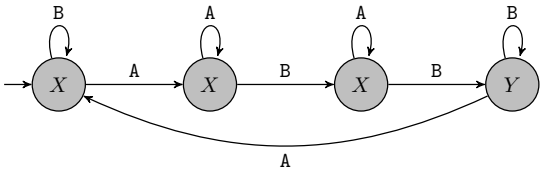
Choose x until you see a, after which switch to y. Then, choose y until you see b, after which switch to x. Then, if you see b, switch to y for the rest of the round.

4S-4T



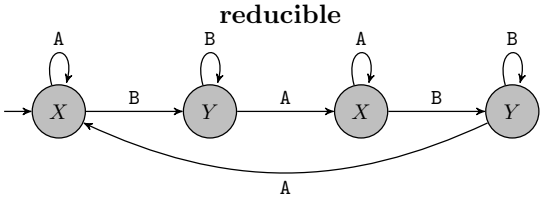
Choose x until you see a, after which switch to y. Then, after you see b switch to x. Then, after you see another b switch to y. Then, start over after you see a.

sequenceable



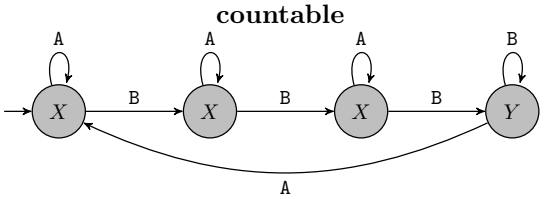
Choose x until you've seen a followed at some point by b followed at some point by another b, after which switch to y. Then, start over after you see a.

reducible



Choose x until you see b, after which switch to y. Then, choose y until you see a, after which point switch to x. Then, after you see b, switch to y. Then, start over after you see a.

countable



Choose x until you've seen b three times so far (not necessarily in a row), after which switch to y. Then, start over after you see a.

B Additional Data Analysis

B.1 Additional Estimation

Table 6 displays estimates from several robustness regressions. First, our cost measurement is left-truncated since subjects cannot submit a willingness to pay below \$2 (i.e. a cost of 0) and about 16% of WTP observations are at this lower bound. To study the robustness of estimates reported in the body of the paper, we estimate a Tobit model with identical dependent variables to specification (1) from Table 5 in the main body of the paper. Results are similar to those reported in our main specification. The most noticible difference is that the "Sequenceable?" variable, which has almost identical coefficients in both cases, is less precisely estimated in the Tobit and is no longer distinguishable from 0.

Model (4) repeats the cost specification from Table 5 and includes a variable for the word count of the rule. This variable ("Words") is statistically insignificant, but its inclusion attenuates other variables that are highly correlated with word count. The main qualitative change relative to estimates from the paper is that the *Sequenceable?* indicator variable is no longer statistically significantly different from zero.

Models (2) and (3) are estimates of the determinants of costs that attempt to remove the influence of subjects' concerns about lost earnings due to future mistakes from the interpretation of the estimates. We will discuss these in Section B.2 below.

B.2 Mistakes and Costs

In Section 4.5 of the paper, we highlight a generally weak relationship between factors driving complexity costs and factors driving mistakes. Here we examine this relationship in more detail. Complexity costs in our experiment may be driven either by (i) sheer subjective distaste for complexity or (ii) fear of making future mistakes (and thereby sacrificing earnings). To distinguish these two explanations for our costs we follow two strategies.

First, we repeat our cost analysis using the (majority) subset of subjects who make no more than 1 mistake in Stage 1 of the experiment. These subjects not only made few mistakes in Stage 1, they also make mistakes fewer than 2% of the time in later stages of the experiment and their costs are therefore unlikely to be driven by fear of making future mistakes. In specification (2) of Table 6, we re-estimate our main cost regression from the main paper on only this subset of subjects. Second, specification (3) of Table 6 attempts to control for fear of mistakes in a different way by including an indicator variable that takes a value of 1 if the subject made a mistake in implementing the rule in Stage 1 (we also include Stage 1 implementation time in this specification); because this specification includes endogenous variables we include subject fixed effects in this specification.

These specifications produce very similar qualitative results to our main specification from the body of the paper and, for the most part, similar quantitative estimates. Subjects who make few mistakes certainly have lower average complexity costs (the intercept for specification (2) is smaller than for our main specification in the body of the paper), and subjects clearly assign greater costs to rules they've made mistakes on in the past (the Mistake? indicator in specification 3 is significantly greater than zero). However, the similar estimates for the remaining variables indicate that concern about the costs of mistakes mostly causes level shifts in the cost function rather than alterations

	Cost	Cost	Cost	Cost
	Tobit	*OLS*	*OLS*	*OLS*
	(1)	(2)	(3)	(4)
States (**H1**, # of states in rule -2)	0.262^{***}	0.175^{***}	0.226^{***}	0.166^{***}
	(0.038)	(0.020)	(0.021)	(0.052)
Transitions (**H2**, # of transitions in rule -1)	0.120^{***}	0.101^{***}	0.117^{***}	0.068^{*}
	(0.047)	(0.018)	(0.019)	(0.036)
Absorbing? (**H3**, indicator absorbing state in rule)	-0.298^{***}	-0.112^{***}	-0.136^{***}	-0.246^{***}
	(0.089)	(0.036)	(0.042)	(0.050)
Reducible? (**H4**,indicator for rule 'reducible')	0.066	0.075^{**}	0.085^{**}	-0.001
	(0.103)	(0.036)	(0.034)	(0.060)
Countable? (**H5**, indicator for rule 'countable')	-0.521^{***}	-0.387^{***}	-0.447^{***}	-0.384^{***}
	(0.091)	(0.047)	(0.049)	(0.083)
Sequenceable? (**H5**, indicator for rule 'sequence')	-0.126	-0.134^{***}	-0.123^{***}	-0.071
	(0.098)	(0.050)	(0.041)	(0.049)
Recall? (indicator for Recall sessions)	0.020	0.046		0.020
	(0.042)	(0.140)		(0.114)
Reasoning? (indicator for Reasoning sessions)	0.344^{***}	0.259		0.301^{**}
	(0.058)	(0.189)		(0.150)
Words (word count of rule)				0.015
				(0.010)
Run 1? (indicator for Run 1 of the experiment)	0.059	-0.096		0.047
	(0.041)	(0.134)		(0.112)
Female? (indicator for gender, 1 = Female)	0.030	0.162		-0.021
	(0.039)	(0.135)		(0.105)
STEM? (indicator for STEM major)	-0.184^{***}	-0.086		-0.157
	(0.040)	(0.131)		(0.106)
Cog Score (cognitive battery score)	-1.941^{***}	-1.167^{***}		-1.801^{***}
	(0.123)	(0.446)		(0.337)
Mistake? (indicator for mistake in rule in Stage 1)			0.122^{***}	
			(0.044)	
Implementation Time (implementation time for rule in Stage 1)			0.001^{**}	
			(0.0005)	
Absorbing X State	-0.134^{**}	-0.095^{***}	-0.137^{***}	-0.133^{***}
	(0.062)	(0.021)	(0.023)	(0.024)
Absorbing X Transition	0.160^{**}	0.072^{***}	0.127^{***}	0.101^{***}
	(0.068)	(0.025)	(0.027)	(0.034)
Log(scale)	0.145^{***}			
	(0.013)			
Constant	0.760^{***}	0.594^{***}		0.939^{***}
	(0.076)	(0.178)		(0.124)
Observations		2,128	3,850	3,850
R^2		0.145	0.754	0.182
Adjusted R^2		0.139	0.734	0.179
Residual Std. Error		0.855 (df = 2113)	0.575 (df = 3565)	1.011 (df = 3834)
Note:				*p<0.1; **p<0.05; ***p<0.01

Table 6: Robustness estimates of drivers of complexity costs. *Notes: The dependent variables is cost in each case. Each specification clusters standard errors at the subject level. Right hand variables include number of states and transitions and a dummy for whether the rule included an absorbing state. We subtract 2 from states and 1 from transitions (i.e. the minimal number of states and transitions in the design) in order to make the intercept easier to interpret. Reducible? , Countable? and Sequenceable? are dummy variables corresponding to the rule of the same name. Recall and Reasoning are dummies for the variations on the Implementation task of the same name. Run 1 is a dummy variable for the first run of the experiment and Female and Stem are gender and college major dummies. The Cognitive Battery is an index of scores from a battery of cognitive tests, ranging in value from a minimum of 0 to a maximum of 1, demeaned. Mistake? is a dummy variable for whether the subject made a mistake in implementing the rule in Stage 1 and Implementation Time is the corresponding Stage 1 implementation time. Words is the number of words in the description of the rule given to subjects.*

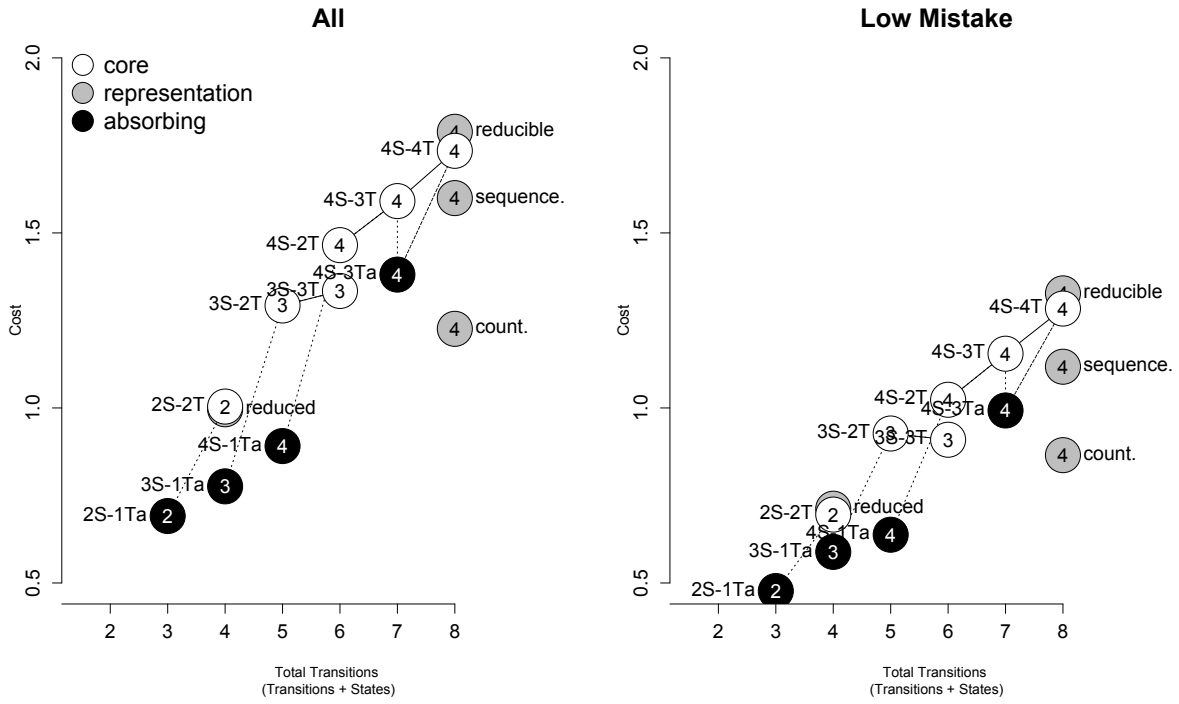


Figure 10: Average complexity measures for all rules. *Notes: Each circle represents a different rule. Horizontal location of each rule is "Total Transitions" (states plus transitions) and the number of states are shown inside each rule's circle. Lines connect all non-representation rules with the same number of states; dotted lines mark transitions that cause all states to be transient (non-absorbing). On the left we show average costs from the full sample and on the right average costs for the subset of subjects who made no more than one mistake during Stage 1.*

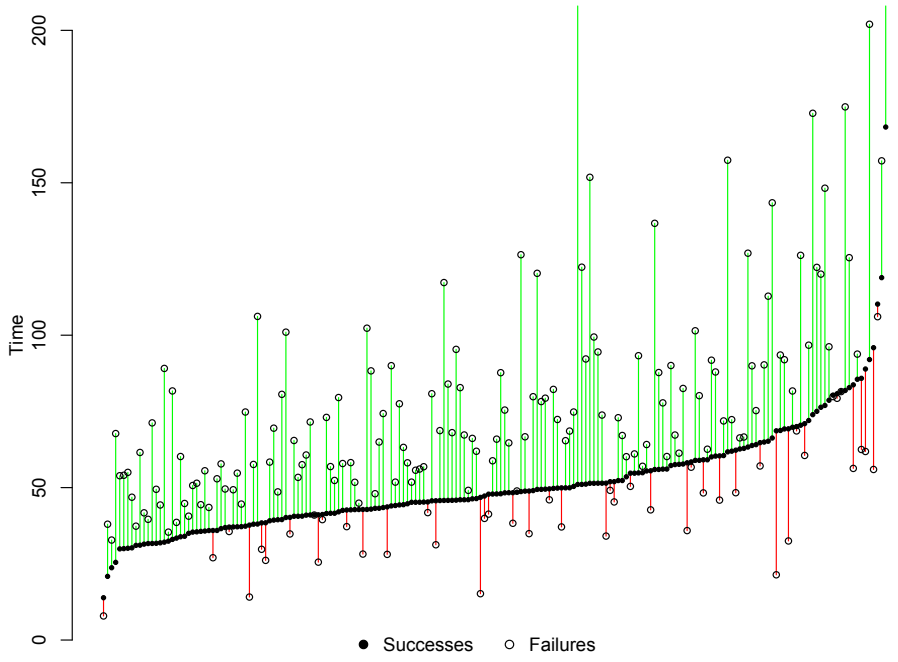


Figure 11: Mean success and failure times. *Notes: Solid dots are average implementation time for successful implementation for each subject, ordered from fastest to slowest. Hollow dots plot, for each subject, the corresponding implementation times for failed implementations for the same subjects. Green lines connect dots for subjects that took longer on average to complete failed than successful implementations while red lines connect dots for subjects for whom the opposite was true.*

in how characteristics of rules influence complexity. This, in turn, suggests that most of the cost-generating effects (e.g. s-complexity, t-complexity etc.) measured in the experiment are likely driven by subjects' distaste for implementing complex features of rules.

In Figure 10 we provide complementary visual evidence by ploting average costs for each of our rules (mirroring Figure 5 from the main text) for (i) the entire sample and (ii) the subset of "low mistake" ($\leq$ 1 mistake) subjects. In line with our estimation results, the shape of the two cost functions are remarkably similar with the latter a downward shift of the former.

B.3 Why Do Subjects Make Mistakes?

Why do subjects make mistakes (fail to implement rules perfectly)? We interpret mistakes as true errors in tracking states or interpreting the sequence of events, and we attribute the increase in mistakes as rules grow complex to increases in difficulty and cognitive load. In this appendix we consider other potential explanations.

A first alternative possibility is that mistakes are driven by confusion over the meaning of the rules as stated. We took great pains to remove this possibility via intensive training on the meaning of specific phrases used in the statements of rules, by providing 5 unpaid practice tasks and by fielding many questions during the practice tasks. The evidence suggests that we were successful, on the whole, at removing this type of confusion. Conditional on making any mistake, the median subject makes their first mistake ten events into the task, long after confusion about the meaning of the rule should have generated a first mistake, on average. We conclude that this is not the main source

of mistakes.

A second is that subjects are insufficiently motivated to attempt to implement the rule and that mistakes are due to subjects choosing not to put effort into implementation. We cannot measure effort directly, but we can show that the evidence is inconsistent with some versions of such an account, and that the data at least provides us little reason to believe that weak incentives drive mistakes.

First, as just discussed, when subjects fail at high mistake-rate rules, the failures occur relatively deep into the task. That is, subjects show evidence of at least initial effort which is inconsistent with a story in which subjects choose not to attempt to follow complex rules at all.

Second, and more importantly, under many incentive stories we would expect to see less effort for rules subjects fail to properly implement than rules they successfully implement. However, Figure 11 shows the opposite pattern. The figure presents as solid dots the average time each subject (among those that made at least one mistake) spent implementing rules successfully, with dots horizontally ordered from the fastest to the slowest subject. Time spent on a rule is the best measure of effort we have in our data. Hollow dots show for each of these subjects the amount of time, on average, they spent implementing a rule unsuccessfully. Green lines connect the dots for subjects who spent more time on their errors than successes and red the reverse. Clearly, almost all subjects spend more time on their failures than their successes, which cuts against an incentives story. Moreover, there is an extremely strong *positive* relationship across rules between the average time taken and average mistake rate (correlation 0.785).

Of course, we cannot rule out the possibility that higher incentives would intensify these effects further (that subjects would put yet more time and effort into difficult rules if given higher incentives) – indeed we would not be surprised if this were true. But this data at least suggests that mistakes are not driven simply by subjects choosing not to attempt to implement rules due to their subjective costs.

C Finite State vs. Pushdown Automata

In this appendix we consider an alternative to the hypothesis that subjects represent rules as "finite state machines" (FSMs), as is typically assumed in automata models in economics. We examine a minimal extension of an FSM called a "pushdown automaton" (PDA) that adds a "memory stack" to the representation of the algorithm. PDAs are usually considered the next step in sophistication and flexibility of an algorithm beyond an FSM (they are one step lower on the "Chomsky hierarchy" used to classify the sophistication of automata) and they allow for computations that FSMs are incapable of. For our purposes and in the context of our experiment, PDAs allow an agent to use simplifications that FSMs can't employ, in particular simplification by *counting*.

We describe PDAs formally and then show how they can allow decision makers to simplify some types of rules studied in the experiment (in particular rules *countable* and *sequenceable*), but also that they do not simplify most of our other rules in any significant way relative to FSM representations. We conclude by re-examining our data through the lens of PDAs and consider what this tells us about modeling rules.

C.1 Pushdown Automata

A pushdown automaton is much like an FSM except that the transition function depends not only on events, but also on the top (first) element in a "stack" – a string of symbols held in memory, with the set of possible symbols in the stack drawn from an "alphabet" set Γ. The PDA is a four-tuple (S, s^0, f, τ) where S is a set of *states*, s^0 is the initial state, $f : S \to R$ is an output function, specifying a response in each state, and $\tau : S \times E \times \Gamma \to S \times \Gamma^*$ (where Γ^* is the set of all finite strings formable from elements in the set Γ) is a *transition* function that specifies a next state and a new stack as a function of the current state, the current event and the top (first) element in the current stack.

Transitions are allowed to either add to the top of a stack ("push" to the stack) or remove the top element of a stack ("pop" from the stack) or both. A typical notation for describing these operations is with the instruction `h, i` $\to$ `j` which says "Make this transition if event h has occurred *and* the top element in the stack is i. Replace the top element in the stack with string j." In this notation, replacing an element with ε removes the element and the element ε is read by the machine only if the string is empty.

C.2 Counting

For example, the rule *countable* is represented in FSM form on the left side of and PDA form on the right side of the top row of Table 8. The PDA representation of the rule starts with an initial transition triggered by no event (ε) that says to replace the initially empty stack with a stack of two symbols 'BB.' The rule says to choose X in this initial state and to ignore 'A' while in this state. Whenever a 'B' occurs, if the top element of the stack is a 'B', remove it (replace it with nothing, ε). If a 'B' occurs when the stack is empty (i.e. two B's have occurred and the stack is ε), transition to the right hand state and choose Y. Whenever an 'A' occurs in this right hand state, repopulate the stack with 'BB' and return to the first state.

The PDA representation describes *counting*, the intuitively natural way of representing this rule and

this allows it to take advantage of a feature of this rule that it does not shared with other similarly s-complex/t-complex rules. Specifically, unlike similar rules like *4S-4T*, the FSM representation of rule *countable* does not use its first three states to change behavior or to process information – it only uses states to remember events. Formally, in contrast to *4S-4T*, responses (i.e. actions) and transitions are exactly the same in each of the first three states of the FSM of *countable*.

Because of this, representing *countable* as a PDA rather than an FSM is highly efficient (at least from the perspective of s-complexity and t-complexity). The FSM representation has 4 states and 4 (net) transitions, while the PDA representation has only 2 states and 4 (net) transitions. Thus, under the hypotheses of s-complexity and t-complexity, (and assuming that holding a simple string in working memory adds few costs), we would expect people to find the PDA representation less complex. This is exactly what we find evidence of in our data. Compared to other rules that cannot be simplified by counting (e.g. rules like 4S-4T which have the same number of states and transitions in FSM-form but are not reducible by counting), the rule *countable* is faster, less costly and generates fewer mistakes.

C.3 Event Stacks and Countable Stacks

Not all PDA representations will generate efficiency gains as dramatic as the rule *countable* and as we will see below most generate no efficiencies at all. Reduction to a PDA for rule *countable* is particularly efficient for two reasons.

First the PDA representation of *countable* can use a special kind of memory stack that we will call an "event stack." In an event stack, the DM (i) uses a stack alphabet Γ consisting of the same characters as the set of possible events E and (ii) eliminates the top element in the stack precisely when she sees an event matching the top event of the stack. Transitions occur when the stack is empty. An event stack can be used to reduce a rule exactly when it contains a series of states in sequence that all prescribe the exact same response. As is clear from the FSM representation of *countable*, in the first three states the prescribed response never varies – it is always X. The rule can thus be represented with two states and an initial event stack consisting of all of the events that trigger transitions across the first three states in the FSM representation. Simply put, states are replaced with a string of anticipated events, held in working memory.

The rule *sequenceable* in the second row of Table 8 has just as many FSM states and transitions as *countable* and it can also be reduced using an event stack (because it, too, specifies the same response across each of the first three states). However a glance at the reduced PDA on the right shows that the reduction is less dramatic. While the PDA for *countable* has 2 states and 4 transitions, the PDA for *sequenceable* has 2 states and 7 transitions. While reducing *countable* is a "free lunch," removing states without adding any compensating transitions, *sequenceable* can only be reduced by two states by adding three transitions in exchange.

The difference between these two cases is that *countable* uses not only an "event stack" but a "counting stack" – an event stack in which the stack library is a singleton, consisting of only one event to be added or subtracted from the stack. Counting stacks require fewer transitions than otherwise identical event stacks and in our example, this allows the rule to be reduced without adding transitions. While an FSM can be reduced with an "event stack" whenever it contains a sequence of contiguous states with identical responses, a rule can be reduced with a "counting stack" if, additionally, the transitions between all of these states are triggered by the same event.

Again, the data bears out what the PDA representations seem to suggest. The rule *sequenceable* is significantly less costly than rules that can't be reduced with event stacks, and the rule *countable* is significantly less costly than *sequenceable*. This evidence is thus consistent with the hypothesis that people do represent rules in PDA-like ways when an event stack representation is possible, leading to complexity efficiencies. Or, to put it another way, the formal properties of an FSM are not complete descriptions of the complexity of all rules and the possibility of using PDA representations help to rationalize this descriptive shortcoming.

C.4 Countable Rules are Not Necessarily More Efficient

The rule *countable* is not the only one in our data that can be reduced to a counting stack PDA. In fact, all of our 3-state rules (3S-1Ta, 3S-2T and 3S-3T) also have this property. Again, in Table 8 we present (on the left) FSM representations and (on the right) PDA representations of these three rules.

What the Table shows is that the efficiency gains of representing these rules using counting stack PDAs are smaller than those achievable for the rule *countable*. In order to reduce these 3-state rules by *one state*, the agent must add *two transitions* in the conversion to a PDA. (Recall that for *countable* a greater – 2 state – reduction is achieved without adding *any* transitions.)

Whether this trade off is ultimately efficent depends on the relative costs of adding transitions vs. states to a rule. Given our estimates in the main body of the paper (two transitions are cost-equivalent to one state), we would expect there to be no efficiency gains from representing 3-state rules as PDAs rather than FSMs and so would expect no difference in complexity for these rules relative to other, non-countable rules.

Adding a dummy variable for 3 state rules to our main cost regression from Table 5 in the main body of the paper confirms this. This dummy variable (i) is insignificantly different from zero ($p = 0.738$) and (ii) including it has no effect on the remaining estimates. Thus, countable rules seem to differ in complexity costs exactly when we would expect them to (given our estimates on the relative costs of states and transitions), and are no different otherwise.

C.5 Rules that Can't Be Reduced with Event Stacks

Any rule that can be represented with an FSM can be represented with a PDA. Indeed, all of the rules we study in the experiment can be reduced to two-state PDAs (because they require only two different responses). However, most cannot be reduced to "event stack" PDAs, which severely limits the efficiencies of the reduction. Table 9 shows automata for all of these "non-event stack" rules and their PDA conversions.

These non-event stack rules have in common that no two contiguous states prescribe the same response (i.e. action by the subject). Because of this, they also have a much less intuitive PDA representation than the event stack rules. Importantly, the stack in each case does not contain a set of future events but instead an arbitrary token (we chose 'T' in our representation) that it varies not to remember events (as in an event stack) but to conduct more sophisticated information processing.

For all of these non-event stack rules, removing states in the reduction requires adding many transitions. Specifically, for the non-event rules we consider, removing N states from an FSM requires

adding an additional $2N-1$ transitions in the PDA representation. Given the 2-1 cost conversion rate of transitions to states estimated in Table 5, the cost savings of representing each as PDAs rather than FSMs should be equivalent to about 1 transition.

Perhaps just as importantly, it isn't clear that PDAs of non-event stack rules actually create *simpler* representations than FSMs, in contrast to event stack rules like *countable*. It may actually be *less* cognitively taxing to represent these rules with a full set of four states as in an FSM representation. Consider the rule *4S-4T*. In PDA form it says to begin with an arbitrary symbol (call it 'T'). Choose 'X' in the first state and (i) if the stack is full (contains a T), transition if you see A but (ii) if the stack is empty, transition of you see B. While in the second state, choose 'Y' and (i) if the stack is empty, transition of you see A (and repopulate the stack with T), (ii) if the stack is full, transition if you see B (and empty the stack).

What seems important here, is that the PDA representation of the non-event rule is still "state like" in the sense that it requires the decision maker to divide history up into four possibilities for the purpose of deciding both what to do and how to respond to future events. Formally it contains only two states (because it contains only two responses, X and Y), but it achieves a four state partition using the composition of the memory stack. By contrast, event stack rules reformulate the problem into a qualitatively different two state rule with the character of a waiting problem: take an action until a sequence of events occur and then transition. The economies of a PDA representation are clear in the latter case but not in the former and this is partly summarized in the differences in the economy of transitions across the two types of cases.

C.6 FSM vs. PDA Representations and Complexity

We have argued that representing rules using pushdown automata allows us to describe distinctions between some rules (in particular *4S-4T* relative to *countable* and *sequenceable*) that do not formallly arise in finite state machine automa. This raises the question: are PDAs overall better suited to describing the complexity of rules than FSMs?

To study this question, we report several regressions in Table 7 that examine the determinants of complexity costs using FSM characteristics versus PDA characteristics. For this exercise, we demean cost by subject (removing subject-to-subject variation in average costs) in order to focus on drivers of within-subject cost differences. In specification (1) we include the features of an FSM: states, transitions and absorption required by the pattern of action described by the rule. In specification (2) we include the feature of the (minimal state) PDA that differ across rules: transitions and absorption (note that states are always equal to 2). Notice that we do not include indicator variables for our representation rules (*countable*, *sequenceable* and *reducible*) as in Table 5 because our aim is to study and compare the explanatory value of characteristics of the automaton representations themselves.

Our key observation is that the R^2 is actually a bit *higher* in the PDA specificiation (2), even though the PDA includes fewer explanatory variables. This is driven entirely by the improved ability of PDAs to account for the efficiencies of the event-stack rules, *countable* and *sequenceable*. If we remove all of our reducible rules (*countable*, *sequenceable* and *reducible*) and re-run the two specifications as we do in specifications (3) and (4), we find that there in fact is a small R^2 advantage to the FSM representation.

These results suggest that subjects may sometimes represent rules in PDA form and sometimes

	Cost			
	(1)	(2)	(3)	(4)
States - 2 (FSM)	0.244***		0.239***	
	(0.021)		(0.021)	
Transitions - 1 (FSM)	0.047***		0.116***	
	(0.015)		(0.019)	
Transitions - 3 (PDA)		0.146***		0.145***
		(0.011)		(0.011)
Absorbing?	−0.283***	−0.254***	−0.203***	−0.256***
	(0.039)	(0.039)	(0.039)	(0.039)
Absorbing X Transitions - 3 (PDA)		−0.009		−0.008
		(0.010)		(0.010)
Absorbing X States - 2 (FSM)	−0.144***		−0.140***	
	(0.023)		(0.023)	
Absorbing X Transitions - 1 (FSM)	0.201***		0.131***	
	(0.027)		(0.027)	
Constant	−2.300***	−2.423***	−2.380***	−2.421***
	(0.031)	(0.036)	(0.037)	(0.037)
Observations	3,850	3,850	3,025	3,025
R^2	0.240	0.254	0.261	0.251
Residual Std. Error	0.568 (df = 3844)	0.563 (df = 3846)	0.551 (df = 3019)	0.554 (df = 3021)
Note:				*p<0.1; **p<0.05; ***p<0.01

Table 7: Estimates of drivers of complexity cost *Notes: Dependent variables inclue an absorption indicator variable and either (i) FSM states/transitions or (ii) PDA transmission. Specification (3) and (4) are identical to (1) and (2) but have the rules* countable, sequenceable *and* reducible *removed from the dataset. All standard errors are clustered at the subject level.*

in FSM form – in particular they may represent rules representable by event stacks as PDA-like and other rules as FSM-like. Accounting for characteristics of the FSM representation is clearly valuable when considering the set of rules that *cannot* be represented with an event stack. To this evidence we add that, in PDA form, rules *4S-4T* and *sequenceable* are identical but the latter is about one-transition less costly in our main estimates from the paper. One interpretation is that subjects only take advantage of the efficiencies of a PDA-like representation when the associated stack is an intuitive event stack.

Nonetheless, on net, given the rules we study, PDAs seem a more effective way of taxonomizing rules than FSMs for the purposes of explaining and predicting complexity. However, we emphasize that further empirical study of this question is needed, as our dataset is not ideally suited to fully answering this question. In particular, because we constrained our experiment to rules with two possible responses, the number of states required in a minimal PDA is always fixed at two. A more careful examination of the relative value of PDAs and FSMs would vary the response set and the set of possible events independently so as to separately vary the number of states and transitions required of a rule both in PDA and FSM form. Doing this would provide a stronger basis for arguing that PDA are more valuable than FSM representations for modeling complexity. We further emphasize that exploration of alternative algorithmic descriptions seems important – there may be broader ways of describing algorithms (or specifying cost functions over them) that endogenously account for these differences in representation.

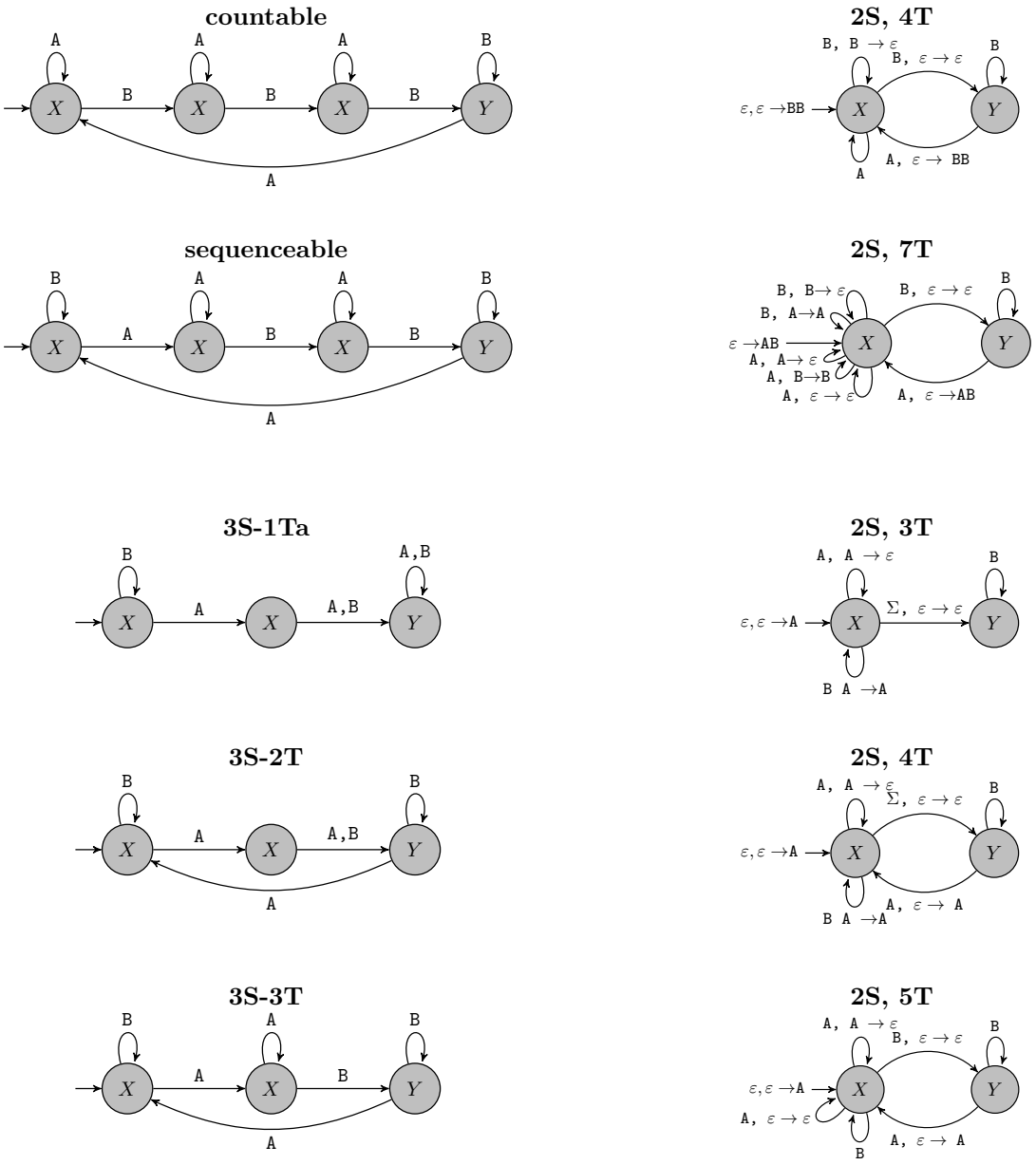


Table 8: Event Stack Rule PDA conversions. *Notes: On the left we show the FSM representation (with the rule's name above) and on the right the corresponding PDA representation (with the number of states and transitions (net of states) for the PDA representation).*

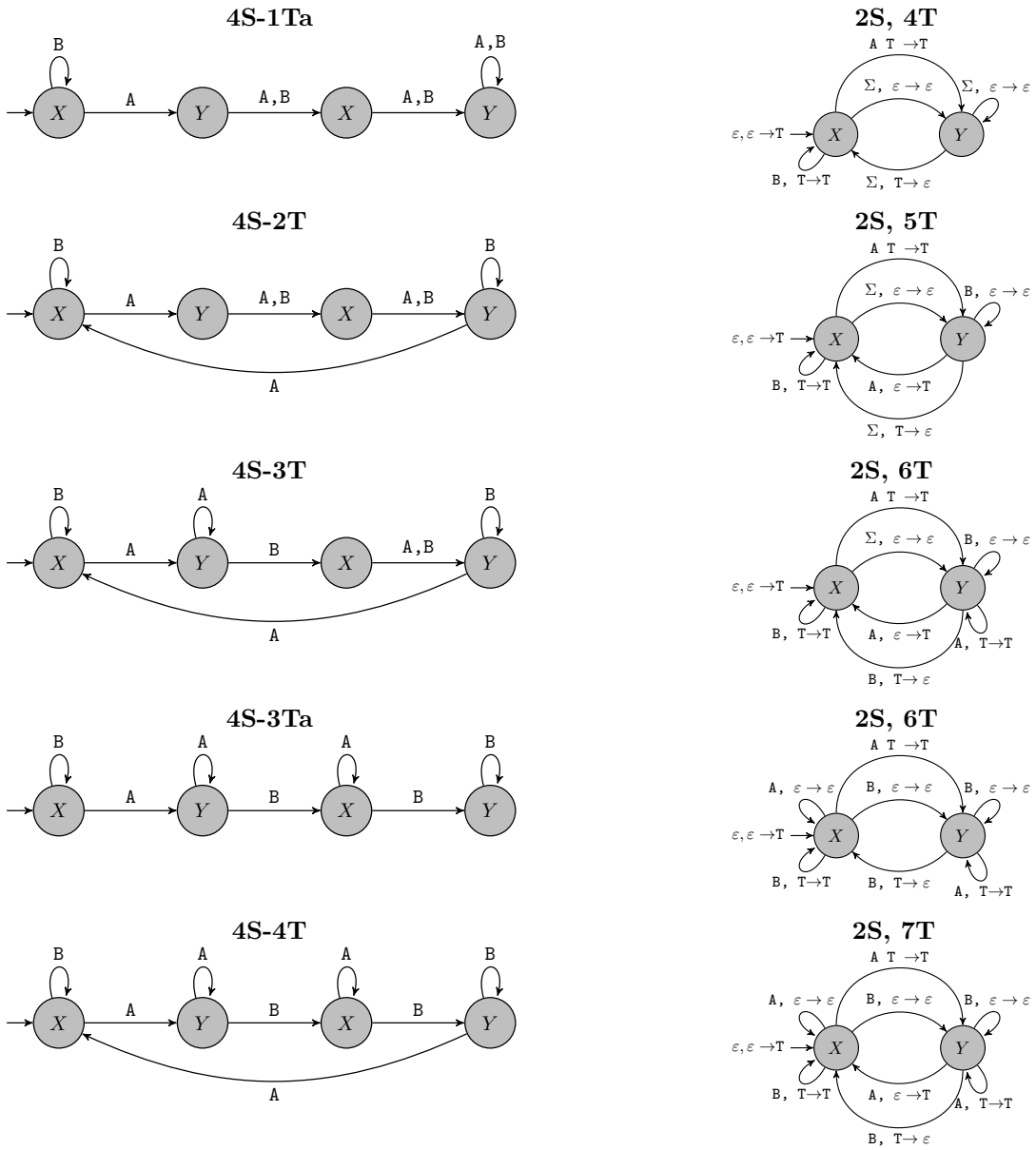


Table 9: Non-Event Stack Rule PDA conversions. *Notes: On the left we show the FSM representation (with the rule's name above) and on the right the corresponding PDA representation (with the number of states and transitions (net of states) for the PDA representation).*

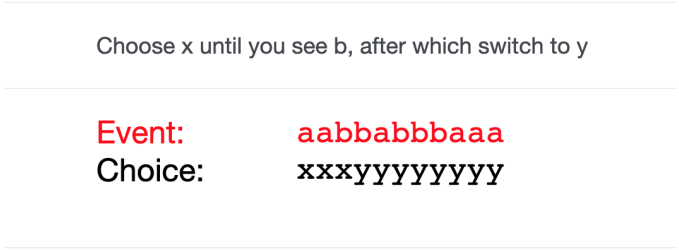
D Instructions to Subjects

D.1 Stage 1 Instructions

1. **ROUNDS, TICKS and EVENTS:** In Stage 1 you will participate in a **rule-following task.** The Stage will be divided into 14 **ROUNDS** and each round will be divided into 20 **TICKS**. In each tick two things will happen in sequence:

   - you will type a letter (which we will call your **CHOICE**)
   - after you've typed your letter you will see an **EVENT** (a letter **randomly** chosen by the computer and displayed in red on your screen)

   Your screen will look like the following:

2. **RULES and EARNINGS:** In each round we will display a **RULE** on your computer screen specifying what letter to type each tick (in the example above the rule is "Choose x until you see b, after which switch to y"). The rule will instruct you what to type in the first tick and, afterwards, what to type in response to the sequence of events you have seen so far.

   If you **perfectly follow the rule** (type exactly what the rule asks for in each tick) you will **earn $4** for the round (otherwise you will earn 0 for the round).

   - Example: *Suppose the rule was "Choose x for the entire round." Then you would simply type x in every tick regardless of what events have occurred.*
   - Example: *Suppose instead the rule was "Choose x until after see b, after which switch to y." Now your choice depends on whether you have seen a b so far. If the events for the round were aabbabbbaaa, then you would need to type* ***xxxyyyyyyyy*** *to earn money.*
   - Example: *In the previous example, if the events had instead been aaaabbbbbbb you would need to type* ***xxxxyyyyyyy*** *to earn money.*

3. **RULES and EARNINGS:**

   - When a rule says to **switch** to a new letter, it is telling you to type that new letter every subsequent tick, until the rule instructs you to change again.
   - When a rule says to **start over** it is telling you, for the rest of the round, to act as though all previous ticks and events had never occurred. You must therefore start by typing the first letter the rule prescribes and then follow the rule based on the events that occur after the start over (exactly as though the previous ticks and events hadn't happened).

- Example: *Suppose the rule is "Choose x until you see b, after which switch to y. Then, after you see an a start over." If the events are ababb then in order to follow the rule you must type **xxyxy**. Following the rule, you choose x until you see the first b (in the second tick) and then switch to y after, in the third tick. Since a occurs in the third tick, you start over in the fourth tick – from then on, you'll ignore the events in the first three ticks. The rule says to choose x until you see the first b, which occurs here in the fourth tick (remember the rule tells us to ignore anything that happened in the first three ticks), then switch to y. So in the fifth tick you switch to y.*

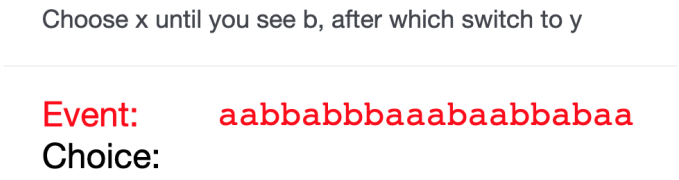
- When a rule says something like "if you see a **followed at some point** by c, switch to y" it is telling you that you should switch to y after you see c occur following an a, even if the c did not happen immediately after the a.
- If your rule tells you something like "choose x until you see a b, after which switch to y" and you see a b in the same tick you first chose x, then you should switch to y. That is, even if the event that triggers a switch happens immediately, you should switch right away.

4. **DETAILS:** You will start each round with a **blank screen**, make your first choice and you will then see the event that follows in the first tick. As you make choices, you will see all of the previous ticks of the round. (In the "Recall" variation: You will only ever see your choice and the event for the current tick.) When all 20 ticks have occurred for the round (when you have typed 20 letters), you will learn whether you made a mistake (and see the correct series of choices) and see a **button** that you must click to move onto the next round. The computer will select one round from this stage randomly for payment.

In the "Reasoning" variation, the following instructions appeared after subjects engaged in 5 practice periods under the Base variation:

"In the actual experiment, you will not be asked to follow the rule on the screen. Instead, you will see a full set of events and be asked to **forecast** what decision a person would choose in the final tick (e.g. prior to seeing the event in the final tick) if he or she **perfectly** followed the rule . This means you will type only one letter (make one choice) per round and will be paid based on whether this choice is correct."

""Your screen will look like the following:"



D.2 Stage 2 Instructions

- **YOUR NEXT RULE:** Later on, in Stage 3, you will again follow a rule but this time it will be the **same rule repeatedly** for 10 rounds. In this Stage (Stage 2) you will have a

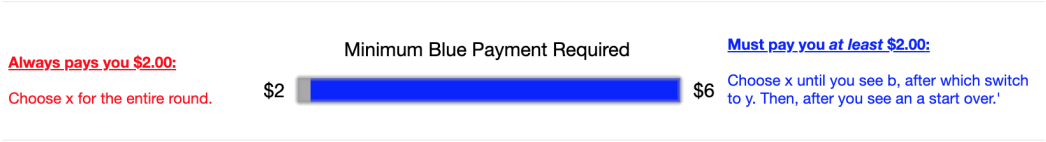
chance to influence which rule you are asked to follow for those 10 rounds.

You will be shown a series of blue rules (on the right side of the screen) and a very simple red rule (on the left side of the screen) and you will tell us **how much you'd have to be paid** to want to play the blue rule insted of the red rule.

- **MYSTERIOUS BLUE PAYMENT:** The red rule always pays $2.00 (if successfully followed) but is very simple. The blue rule is more involved and it will pay (if successfully followed) some amount chosen randomly by the computer between $2.00 and $6.00 . You will not find out how much the blue rule pays (the "**blue payment**") until the end of the experiment!

- **MINIMUM PAYMENT FOR BLUE:** Your task is to decide how much the blue rule would have to pay you, **at a minimum**, to make you want to follow it instead of the simpler red rule. If the randomly selected blue payment is higher than this number, you will play the blue rule (and get paid the blue payment) instead of the red rule. If the blue payment is below this minimum, you will instead play the red rule and get paid $2.00.

- **SLIDER:** You will choose your minimum payment for blue using a **slider**.



  - To the left and right of the slider are shown the two rules (red and blue). At the top of the red rule is a reminder of the fixed payment of $2.00 if you are asked to follow that rule. At the top of the blue rule is a **reminder** of the minimum payment for blue you've selected.
  - By moving the slider with your mouse (or using left/right arrow keys), you can adjust your minimum blue payment. As you move the slider, you'll notice that the minimum amount above the blue rule changes (between $2.00 and $6.00).
  - The area of the slider colored blue are all of the randomly selected blue payments that would cause the blue rule to be selected for you by the computer. The area colored red are all of the randomly selected blue payments that would cause the red rule to be selected for you instead. The further to the right is the slider, the more likely you'll get the red rule; the further to the left, the more likely you'll get the blue rule (but the lower that rule might pay).

- **THE SELECTED RULE:** We will have you make a **series** of slider choices, varying the blue rule each time. With 50% chance, the computer will randomly select one of these choices: if the blue payment is at least as large as your minimum blue payment you will play the blue rule in Stage 3; otherwise you will play the red rule. (With 50% chance the computer will assign a rule for you to play in Stage 3, independently of your choices.)

- **DETAILS:** Remember, if the blue rule is selected, you will earn the randomly selected blue payment not the minimum blue payment selected. The payment scheme is designed so that it is in your best interest to truthfully tell us the minimum amount you'd need to be paid to want to play the blue rule instead of the red rule. The initial value of the slider will be **$2**: please do not take the initial value as a suggestion about what you should choose!

D.3 Stage 3 Instructions

In Stage 3 you will participate in 10 rounds of the rule-following task from Stage 1 . The rule you will receive in all 10 rounds was determined in part by your decisions in Stage 2 and the blue payment randomly selected by the computer.

If this resulted in you receiving a red rule you will earn $2.00 if you successfully follow the rule; if it resulted in you receiving a blue rule you will earn the blue payment which was randomly selected between $2.00 and $6.00 by the computer if you successfully follow the rule.

At the end of the experiment, the computer will randomly select one round from this Section for payment.

D.4 Stage 4 Instructions

Later on, in Stage 5, you will again receive a rule to follow for 10 rounds. In this Stage (Stage 4) you will again be shown a red rule and a blue rule and decide on a minimum blue payment you require to be willing to receive the blue rule.

We will first show you a screen with a blue rule that is the same you received in Stage 3. Next, we will show you a screen with a different blue rule. In both cases you will tell us your minimum blue payment using a slider.

Everything will be just as in Stage 2: the computer will randomly choose a blue payment between $2.00 and $6.00 and if this is greater than your minimum blue payment choice, you will receive the blue rule and be paid the random blue payment the computer selected. Otherwise you will receive the red rule and be paid $2.00 for successfully following it.

D.5 Stage 5 Instructions

In Stage 5 you will participate in 10 rounds of the rule-following task from Stage 1 . The rule you will receive in all 10 rounds was determined in part by your decisions in Stage 4 and the minimum blue payment randomly selected by the computer.

If this resulted in you receiving a red rule you will earn $2.00 if you successfully follow the rule; if it resulted in you receiving a blue rule you will earn the blue payment that was randomly selected between $2.00 and $6.00 by the computer if you successfully follow the rule.

At the end of the experiment, the computer will randomly select one round from this Section for payment.