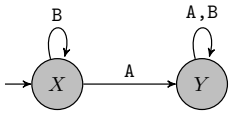
Online Appendix for "What Makes a Rule Complex?"

Ryan Oprea*

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

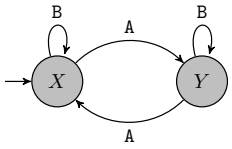
A Rules Studied in the Experiment

2S-1Ta



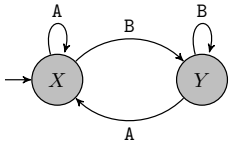
Choose x until you see a, after which switch to y for the rest of the round.

2S-2T



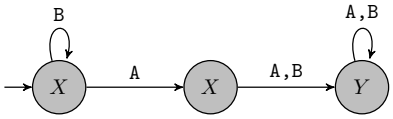
Choose x until you see a, after which switch to y. Then, start over after you see a.

reduced



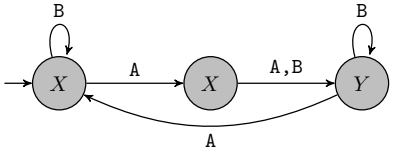
Choose x until you see b, after which switch to y. Then, start over after you see a.

3S-1Ta



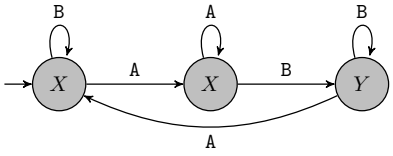
Choose x until you see a, after which choose x once more and then switch to y for the rest of the round.

3S-2T



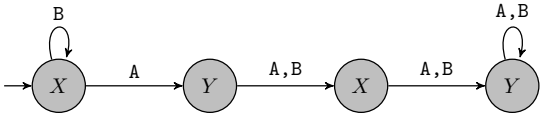
Choose x until you see a, after which choose x once more and then switch to y. Then, start over after you see a.

3S-3T



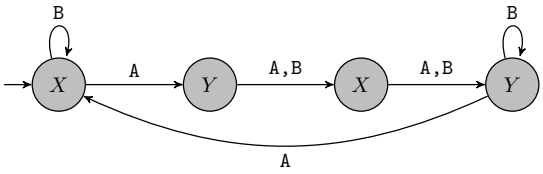
Choose x until you see a followed at some point by b, after which switch to y. Then, start over after you see a.

4S-1Ta



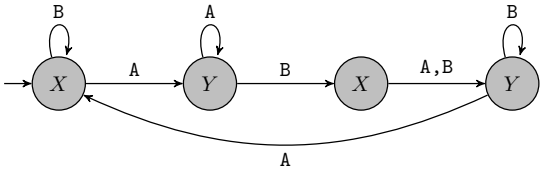
Choose x until you see a, after which choose y once followed by x once and then switch to y for the rest of the round.

4S-2T



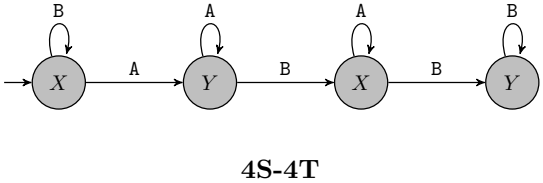
Choose x until you see a, after which choose y once followed by x once and then switch to y. Then, start over after you see a.

4S-3T



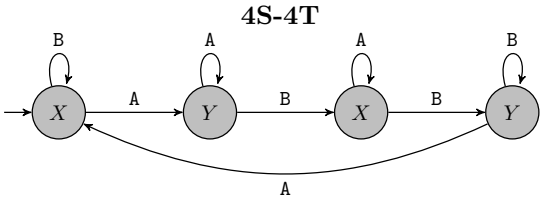
Choose x until you see a, after which switch to y. Then, after you see b, choose x once and then switch to y. Then, start over after you see a.

4S-3Ta



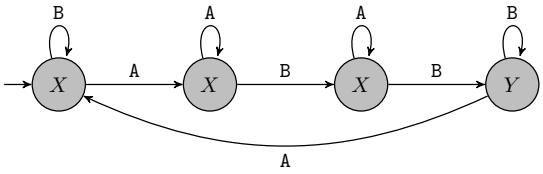
Choose x until you see a, after which switch to y. Then, choose y until you see b, after which switch to x. Then, if you see b, switch to y for the rest of the round.

4S-4T



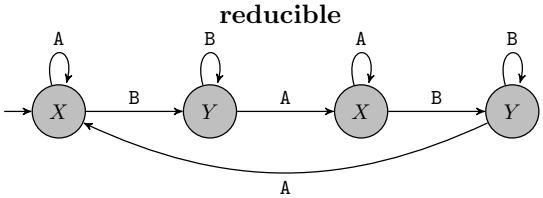
Choose x until you see a, after which switch to y. Then, after you see b switch to x. Then, after you see another b switch to y. Then, start over after you see a.

sequenceable



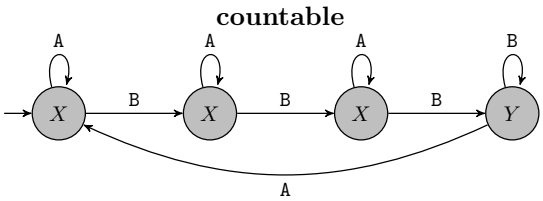
Choose x until you've seen a followed at some point by b followed at some point by another b, after which switch to y. Then, start over after you see a.

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 1 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 in the body of the paper 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 **??** 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 1, we re-estimate our main cost regression from the main paper on only this subset of subjects. Second, specification (3) of Table 1 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 1: 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.*

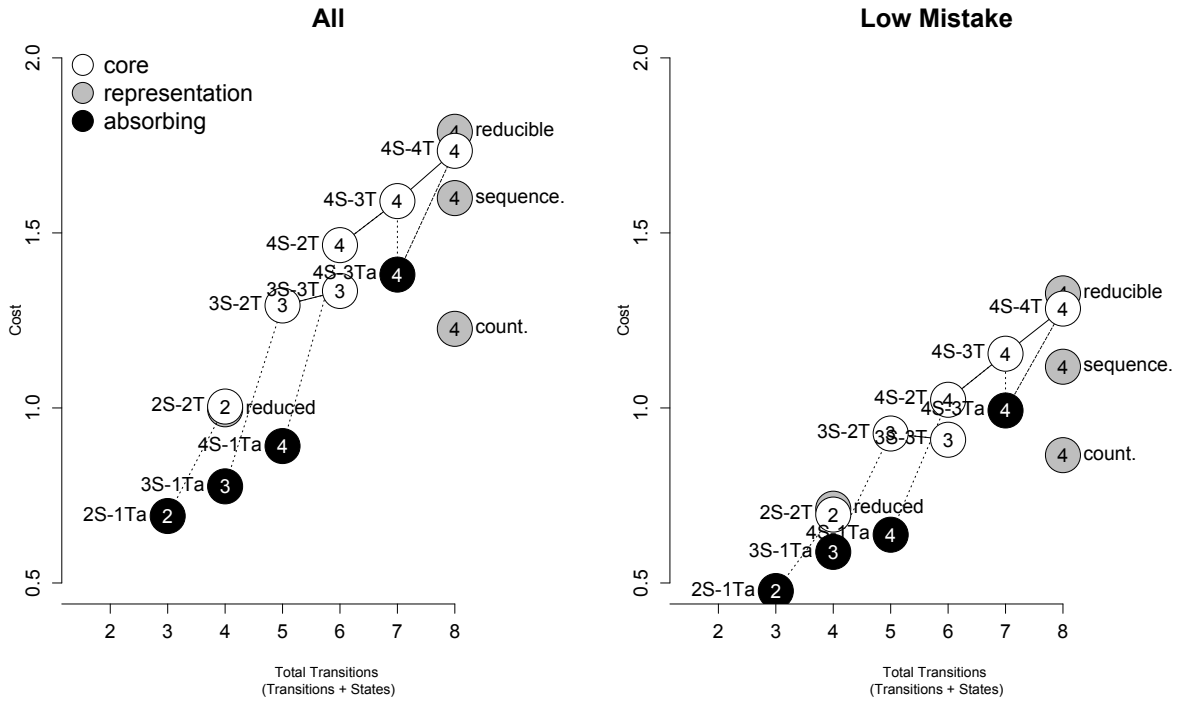


Figure 1: Average complexity measures for all rules. *Notes: Each circle represents a different rule. Horizontal location of each rule is "Total Transitions" (states plus transitions) and the number of states are shown inside each rule's circle. Lines connect all non-representation rules with the same 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.*

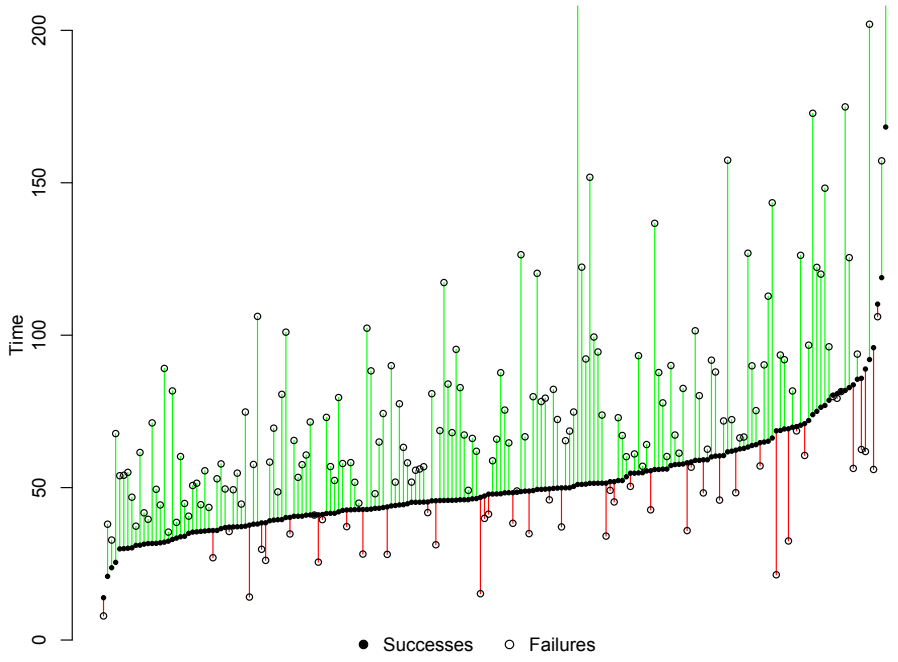


Figure 2: Mean success and failure times. *Notes: Solid dots are average implementation time for successful implementation for each subject, ordered from fastest to slowest. Hollow dots plot, for each subject, the corresponding implementation times for failed implementations for the same subjects. Green 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 1 we provide complementary visual evidence by ploting average costs for each of our rules (mirroring Figure **??** 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 an account 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 2 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 3. 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 3 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 3 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 **??** 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 4 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 **??**, 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 2 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 **??** 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 2: 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.

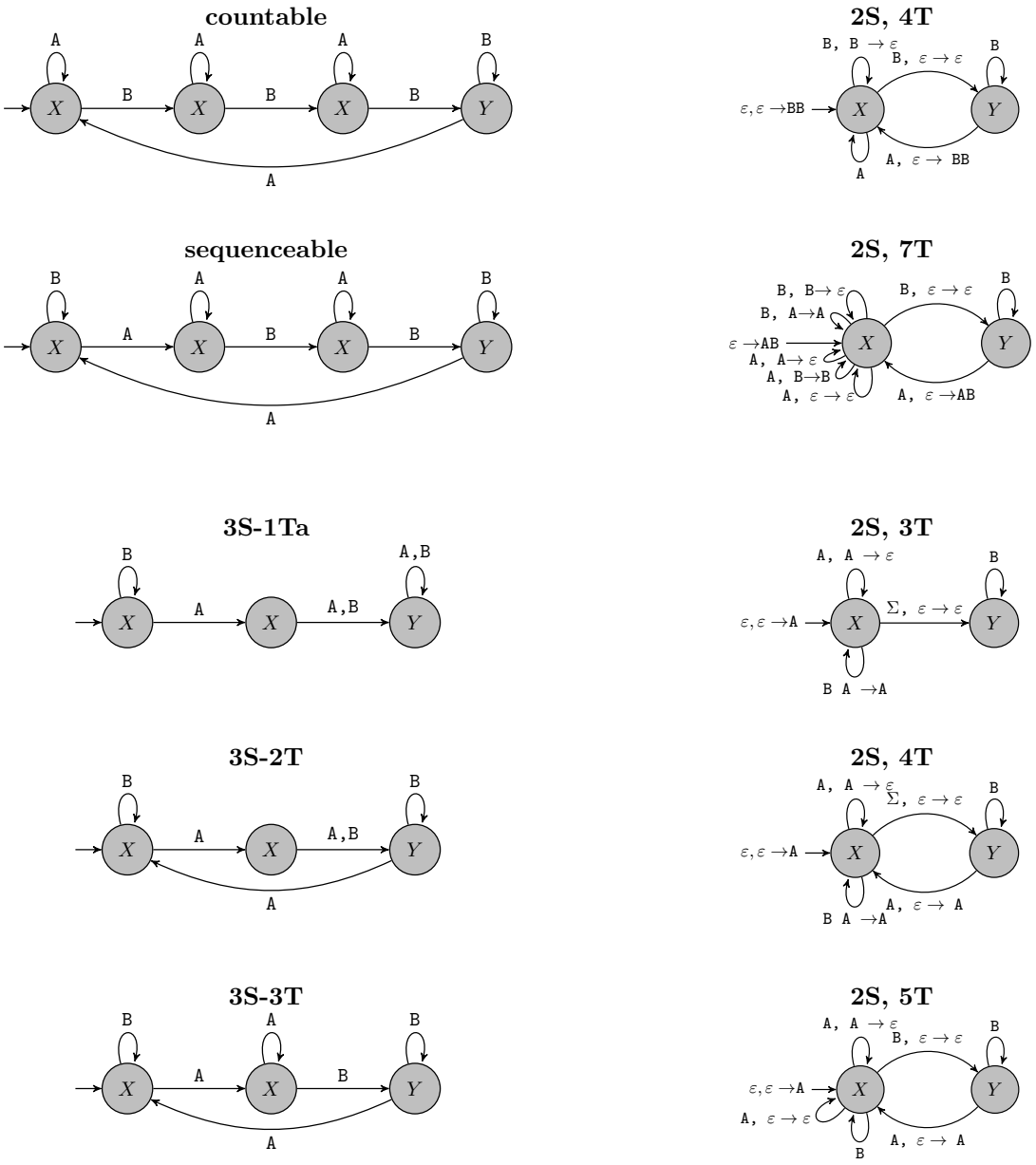


Table 3: Event Stack Rule PDA conversions. *Notes: On the left we show the FSM representation (with the rule's name above) and on the right the corresponding PDA representation (with the number of states and transitions (net of states) for the PDA representation).*

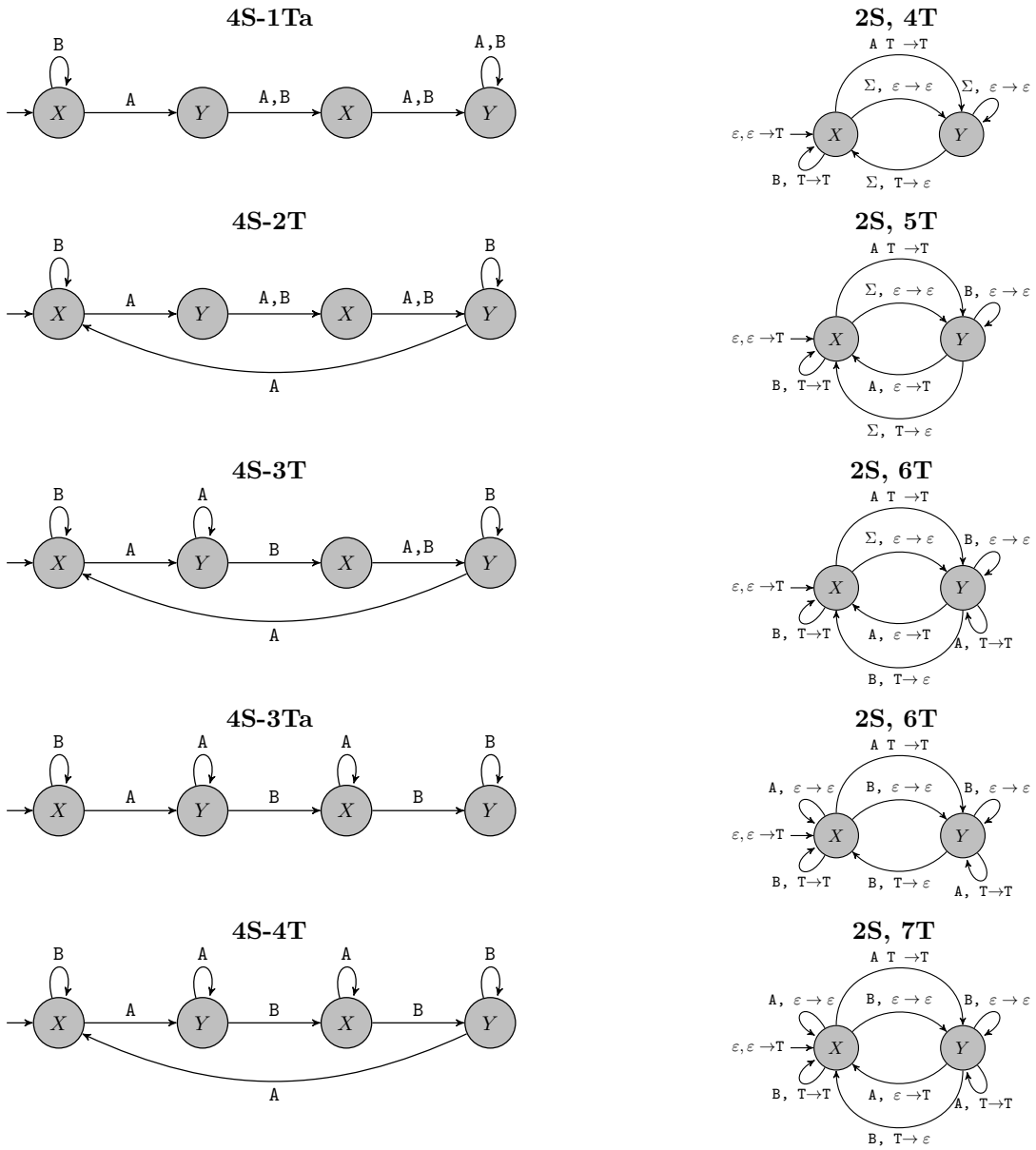


Table 4: Non-Event Stack Rule PDA conversions. *Notes: On the left we show the FSM representation (with the rule's name above) and on the right the corresponding PDA representation (with the number of states and transitions (net of states) for the PDA representation).*

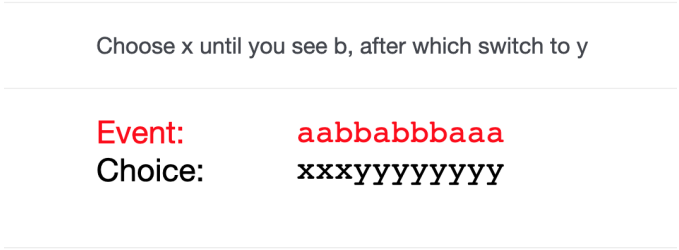
D Instructions to Subjects

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.*

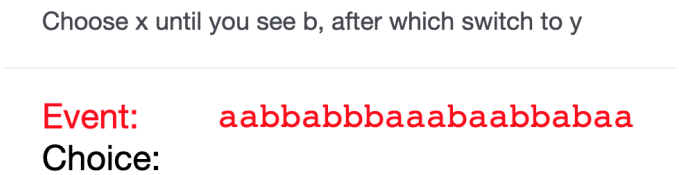
- When a rule says something like "if you see a **followed at some point** by c, switch to y" it is telling you that you should switch to y after you see c occur following an a, even if the c did not happen immediately after the a.
- If your rule tells you something like "choose x until you see a b, 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

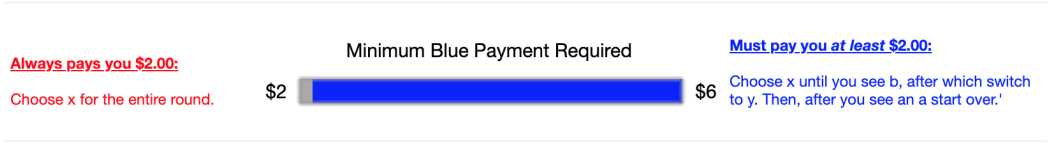
- **YOUR NEXT RULE:** Later on, in Stage 3, you will again follow a rule but this time it will be the **same rule repeatedly** for 10 rounds. In this Stage (Stage 2) you will have a chance to influence which rule you are asked to follow for those 10 rounds.

  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.