Online Appendix for "Slow and Easy: A Theory of Browsing"

Evgenii Safonov

In this Online Appendix, I provide details of calculations for various examples and auxiliary statements in the paper. In Section B.1, I provide details of the calculation of complexity and memory load of languages considered in the example in Section 3 of the paper. In Section B.2, I calculate the number of minimal adequate languages, adapted languages with profiles that solve problem (8) in the paper, minimal adapted languages, and minimal adapted lexicographic languages that are presented in Table 2 in the paper. In Section B.3, I provide calculations for Section 5.3 in the paper; in particular, I derive eqs. (18)-(20) in the paper. In Section B.4, I consider a special case of the model where all attributes must be binary. Finally, in Section B.5, I consider a trade-off between the memory load and transitional complexity from a different angle than in Section 5.1: instead of complexity being a convex combination of the two, I consider transitional complexity subject to automata with no more than a given number of states.

B.1. Example in Section 3.

Let us provide details for the example considered in Section 3. In particular, consider $A = \{a, b, c, d\}$, $a \succ b \succ c \succ d$, and languages Q, R, T, U, W are as shown in Table 1.

Consider first the memory load. Since a near optimal automaton must distinguish any pair of items a, b, c, d, we have $\mu(Q), \mu(R), \mu(T) \geq 2$, $\mu(U) \geq 1$, $\mu(W) \geq 3$. Since automata depicted in Figure 2 of the paper are near optimal, we get $\mu(Q) = \mu(R) = 2$, $\mu(U) = 1$, and $\mu(W) = 3$. Toward a contradiction, assume that for language T, there is a near optimal automaton σ with 2 memory states. The highest-probability paths for items $a = (1,1)$ and $b = (0,0)$ cannot go from state 1 directly to the decision state $\diamond$, since otherwise, one of $c = (1,0)$ or $d = (0,1)$ would use the same highest-probability path, in contradiction to Lemma 5 in the paper. It must be that either question 1 is asked in state 1 and question 2 is asked in state 2, or vice versa. Consider the first possibility; then $\omega^*(a) = (1-\eta)\cdot\tau(1,2,1)\cdot\tau(2,\diamond,1)$, $\omega^*(b) = (1-\eta)\cdot\tau(1,2,0)\cdot\tau(2,\diamond,0)$, $\omega^*(c) = \max\{\tau(1,\diamond,1), (1-\eta)\cdot\tau(1,2,1)\tau(2,\diamond,0)\}$, $\omega^*(d) = \max\{\tau(1,\diamond,0), (1-\eta)\cdot\tau(1,2,0)\cdot\tau(2,\diamond,1)\}$. By Lemma 4 in the paper, since $a \succ c$, then $\frac{\omega^*(c)}{\omega^*(a)} \longrightarrow 0$, and hence, $\frac{\tau(2,\diamond,0)}{\tau(2,\diamond,1)} \longrightarrow 0$. On the other hand, since $b \succ d$, then $\frac{\omega^*(d)}{\omega^*(b)} \longrightarrow 0$, and hence, $\frac{\tau(2,\diamond,1)}{\tau(2,\diamond,0)} \longrightarrow 0$, in contradiction. The other case, in which question 2 is asked in state 1, is symmetric. Therefore, $\mu(T) \geq 3$, and using Figure 2 in the paper, we conclude

that $\mu(T) = 3$.

Consider now the (transitional) complexity. Note that the proof of Theorem 3 goes through if we relax the assumption $\frac{27}{32} \cdot 2^k < m \leq 2^k$ to $\frac{3}{4} \cdot 2^k < m \leq 2^k$ but demand also that the solution of the optimization problem (8) in the paper must have $z_i = 2$ for all i. Since these requirements are satisfied for $m = 4$ (see Table 3 in the Appendix), a language that attains the lower complexity bound $\underline{\kappa}(4) = 6$ must contain a unique minimal adapted language Q, hence $\kappa(Q) = 6$, but $\kappa(L) > 6$ for all $L \in \{R, T, U, W\}$; to put it differently, this is because none of the languages R, T, U, W are adapted. Since automata shown in Figure 2 are near optimal, we conclude that $\kappa(R) = \kappa(U) = 7$.

Consider language T. By Lemma 5 in the paper, a near optimal automaton for language T must contain at least 2 vanishing transitions. Toward a contradiction, assume there are exactly 2 vanishing transitions with probabilities $\epsilon_1 \longrightarrow 0$ and $\epsilon_2 \longrightarrow 0$. Since each transition occurs in the highest-probability path for an item at most once, without loss of generality, $\omega^*(d) = C_d \cdot \epsilon_1 \cdot \epsilon_2$, $\omega^*(c) = C_c \cdot \epsilon_1$, $\omega^*(b) = C_b \cdot \epsilon_2$, $\omega^*(a) = C_a$, where $\lim C_i \neq 0$, and $\epsilon_1/\epsilon_2 \longrightarrow 0$. Note that $c = (1, 0)$, $d = (0, 1)$, and the vanishing probability ϵ_1 is associated with highest-probability paths for both of these items. Since highest-probability paths for items $c = (1, 0)$ and $d = (0, 1)$ contain disjoint sets of transitions, ϵ_1 must be associated with at least two different vanishing transitions. There is also one more vanishing transition associated with probability ϵ_2, thus the total number of vanishing transitions is at least 3. Since there are 3 states, and in each state, there are at least 2 non-vanishing transitions, $\kappa(T) \geq 9$. The near optimal automaton in Figure 2.c shows that $\kappa(T) = 9$.

Finally, consider language W. By Lemma 5, a near optimal automaton should have at least 2 vanishing transitions. Since a near optimal automaton must have at least 3 states, there are also at least 6 non-vanishing transitions. Therefore, $\kappa(W) \geq 8$. A near optimal automaton depicted in Figure 2.e shows that $\kappa(W) = 8$.

To see why the automata depicted in Figure 2 are near optimal, apply Lemma 4 and note that:

(a) For language Q, Figure 2.a: $q(a) = 1-\eta$, $q(b) = (1-\eta)\cdot\epsilon_2$, $q(c) = (1-\eta)\cdot\epsilon_1$, $q(d) = (1-\eta)\cdot\epsilon_1\cdot\epsilon_2$ with $1 \gg \epsilon_2 \gg \epsilon_1 > 0$.

(b) For language R, Figure 2.b: $q(a) = 1-\eta$, $q(b) = \epsilon_3 + (1-\eta)\epsilon_1$, $q(c) = \epsilon_3 + (1-\eta)\epsilon_1\epsilon_2$, $q(d) = (1-\eta)\epsilon_2$ with $1 \gg \epsilon_1 \gg \epsilon_3 \gg \epsilon_2 > 0$.

(c) For language T, Figure 2.c: $q(a) = 1-\eta$, $q(b) = (1-\eta)\cdot\epsilon_2$, $q(c) = (1-\eta)\cdot\epsilon_1$, $q(d) = (1-\eta)\cdot\epsilon_3$ with $1 \gg \epsilon_2 \gg \epsilon_1 \gg \epsilon_3 > 0$.

(d) For language U, Figure 2.d: $q(a) = 1$, $q(b) = \epsilon_2$, $q(c) = \epsilon_1$, $q(d) = \epsilon_0$ with $1 \gg \epsilon_2 \gg \epsilon_1 \gg \epsilon_0 > 0$.

(e) For language W, Figure 2.e: $q(a) = 1$, $q(b) = (1-\eta)^2 \cdot \epsilon_2$, $q(c) = (1-\eta)^2 \cdot \epsilon_1$, $q(d) = (1-\eta)^2 \cdot \epsilon_1 \cdot \epsilon_2$ with $1 \gg \epsilon_2 \gg \epsilon_1 > 0$.

B.2. Calculations for Table 2 in the paper.

Calculating minimal adapted and minimal adapted lexicographic languages. Let us denote items by the first letters of the alphabet, such that $a \succ b \succ c \succ \ldots$. Let us also work with the enumeration convention that larger values of an attribute correspond to better alternatives; put it differently, $v_i(\cdot)$ is increasing. It is without loss to normalize $v_i(0) = 0$ for any i. Of course, the particular values of $v_i(\cdot)$ matter only to the extent to which they induce a different preference on the vectors of attributes. Lastly, if the induced preference on the vectors of attributes is given, this pins down the corresponding language via identification of the k-th ranked vector of attributes with the k-th ranked alternative, attributes with partitions, and the value of an attribute with a cell of the corresponding partition. We start with calculating all minimal adapted languages for $2 \leq m = |A| \leq 8$ and identifying which of them are lexicographic.

$m = |A| = 2$**.** Thus, $A = \{a, b\}$, $a \succ b$. The solution of problem (8) is $K = 1$, $z = 2$. There is only one adequate language $Q = \{\{a\}, \{b\}\}$; it is adapted and lexicographic. It induces a preference on attributes $1 \succ 0$. An example of an additive utility representation is $v_1(1) = 1$.

$m = |A| = 3$**.** Thus, $A = \{a, b, c\}$, $a \succ b \succ c$. The solution of problem (8) is $K = 1$, $z = 3$. There is one minimal adapted language $Q = \{\{a\}, \{b\}, \{c\}\}$, and it is lexicographic. It induces a preference on attributes $2 \succ 1 \succ 0$. An example of an additive utility representation is $v_1(2) = 2$, $v_1(1) = 1$.

$m = |A| = 4$**.** Thus, $A = \{a, b, c, d\}$, $a \succ b \succ c \succ d$. The solution of problem (8) is $K = 2$, $z = (2, 2)$. There are only two parameters, $v_1(1)$ and $v_2(1)$. If $v_1(1) = v_2(1)$, then $10 \sim 01$, in contradiction to $\succeq$ being strict. Since we can enumerate partitions (questions) arbitrarily, it is without loss that $v_1(1) > v_2(1)$. This additive utility induces a preference $11 \succ 10 \succ 01 \succ 00$. Identifying $a = 11$, $b = 10$, $c = 01$, $d = 00$, we have the following minimal adapted language $Q = \{Q_1, Q_2\}$ with $Q_1 = \{\{a, b\}, \{c, d\}\}$, $Q_2 = \{\{a, c\}, \{b, d\}\}$, which is lexicographic. Note that the preference $11 \succ 01 \succ 10 \succ 00$ corresponds to the same language Q, since partitions Q_1 and Q_2 can be labeled arbitrarily.

Note that problem (8) has the same solution $K = 2$, $z = (2, 3)$ (or $z = (3, 2)$) for both $m = 5$ and $m = 6$. We will see that it would be easier if we first find all adapted languages for $m = 6$.

$m = |A| = 6$. Thus, $A = \{a, b, c, d, e, f\}$, $a \succ b \succ c \succ d \succ e \succ f$. Without loss, let the first attribute be 3-valued, and the second attribute be 2-valued. The 6 items, therefore, are $21, 20, 11, 10, 01, 00$. By our agreement on the enumeration of attributes' values, it must be that $21 \succ x$ for any $x \neq 21$, thus $21 = a$; it also must be that $x \succ 00$ for any $x \neq 00$, thus $00 = f$. Similarly, we must have $11 \succ 10, 01$, and $20 \succ 10$. There are only 5 linear orders that satisfy all these conditions; they are listed below. Therefore, there are at most 5 minimal adapted languages. As we will see below, we are able to find $v_i(\cdot)$ that correspond to the additive utility representation of $\succ$ for each of these linear orders on the vectors of attributes; therefore, each one of them gives rise to a distinct minimal adapted language.

1. $21 \succ 11 \succ 01 \succ 20 \succ 10 \succ 00$. This order is induced by $v_1(2) = 2$, $v_1(1) = 1$, $v_2(1) = 3$. The minimal adapted language is $Q^1 = \{Q^1_1, Q^1_2\}$, $Q^1_1 = \{\{a, d\}, \{b, e\}, \{c, f\}\}$, $Q^1_2 = \{\{a, b, c\}, \{d, e, f\}\}$. This is a lexicographic language: the first attribute matters for comparison between two items only if the values of the second attribute for two compared items coincide.

2. $21 \succ 11 \succ 20 \succ 01 \succ 10 \succ 00$. This order is induced by $v_1(2) = 4$, $v_1(1) = 2$, $v_2(1) = 3$. The minimal adapted language is $Q^2 = \{Q^2_1, Q^2_2\}$, $Q^2_1 = \{\{a, c\}, \{b, e\}, \{d, f\}\}$, $Q^2_2 = \{\{a, b, d\}, \{c, e, f\}\}$. This is not a lexicographic language.

3. $21 \succ 11 \succ 20 \succ 10 \succ 01 \succ 00$. This order is induced by $v_1(2) = 4$, $v_1(1) = 3$, $v_2(1) = 2$. The minimal adapted language is $Q^3 = \{Q^3_1, Q^3_2\}$, $Q^3_1 = \{\{a, c\}, \{b, d\}, \{e, f\}\}$, $Q^3_2 = \{\{a, b, e\}, \{c, d, f\}\}$. This is not a lexicographic language.

4. $21 \succ 20 \succ 11 \succ 01 \succ 10 \succ 00$. This order is induced by $v_1(2) = 4$, $v_1(1) = 1$, $v_2(1) = 2$. The minimal adapted language is $Q^4 = \{Q^4_1, Q^4_2\}$, $Q^4_1 = \{\{a, b\}, \{c, e\}, \{d, f\}\}$, $Q^4_2 = \{\{a, c, d\}, \{b, e, f\}\}$. This is not a lexicographic language.

5. $21 \succ 20 \succ 11 \succ 10 \succ 01 \succ 00$. This order is induced by $v_1(2) = 4$, $v_1(1) = 2$, $v_2(1) = 1$. The minimal adapted language is $Q^5 = \{Q^5_1, Q^5_2\}$, $Q^5_1 = \{\{a, b\}, \{c, d\}, \{e, f\}\}$, $Q^5_2 = \{\{a, c, e\}, \{b, d, f\}\}$. This is a lexicographic language: the second attribute matters for comparison between two items only if the values of the first attribute for two compared items coincide.

$m = |A| = 5$. Thus, $A = \{a, b, c, d, e\}$, $a \succ b \succ c \succ d \succ e$. Since the solution of problem (8) is the same as for $m = 6$, the same 5 orders on the vectors of attributes analyzed in the case $m = 6$ correspond to all minimal adapted languages for $m = 5$. However, for each linear order on the vectors of attributes, there are 6 ways to match 5 items with 6 vectors of attributes; that is, one of the vectors of attributes is missing. For example, take a linear order $21 \succ 11 \succ 01 \succ 20 \succ 10 \succ 00$,

then if 00 is unmatched, then $a = 21$, $b = 11$, $c = 01$, $d = 20$, $e = 10$, and thus, the five relevant vectors of attributes are $21 \succ 11 \succ 01 \succ 20 \succ 10$. If instead 10 is unmatched, then $a = 21$, $b = 11$, $c = 01$, $d = 20$, $e = 00$, and the five relevant vectors of attributes are $21 \succ 11 \succ 01 \succ 20 \succ 00$. Thus, eliminating one of the items from any of the five linear orders analyzed in the case $m = 6$ results in a linear order that is associated with one minimal adapted language for $m = 5$. In total, there are $5 \cdot 6 = 30$ such possible reductions, but some of them are duplicated and even appear 3 times. The total number of distinct linear orders reduced in this way is 20. Let us present them below with the associated minimal adapted languages $R^k = \{R_1^k, R_2^k\}$:

1. $21 \succ 11 \succ 01 \succ 20 \succ 10$. $R_1^1 = \{\{a,d\},\{b,e\},\{c\}\}$, $R_2^1 = \{\{a,b,c\},\{d,e\}\}$. Lexicographic.
2. $21 \succ 11 \succ 20 \succ 01 \succ 10$. $R_1^2 = \{\{a,c\},\{b,e\},\{d\}\}$, $R_2^2 = \{\{a,b,d\},\{c,e\}\}$.
3. $21 \succ 11 \succ 20 \succ 10 \succ 01$. $R_1^3 = \{\{a,c\},\{b,d\},\{e\}\}$, $R_2^3 = \{\{a,b,e\},\{c,d\}\}$.
4. $21 \succ 20 \succ 11 \succ 01 \succ 10$. $R_1^4 = \{\{a,b\},\{c,e\},\{d\}\}$, $R_2^4 = \{\{a,c,d\},\{b,e\}\}$.
5. $21 \succ 20 \succ 11 \succ 10 \succ 01$. $R_1^5 = \{\{a,b\},\{c,d\},\{e\}\}$, $R_2^5 = \{\{a,c,e\},\{b,d\}\}$. Lexicographic.
6. $11 \succ 01 \succ 20 \succ 10 \succ 00$. $R_1^6 = \{\{c\},\{a,d\},\{b,e\}\}$, $R_2^6 = \{\{a,b\},\{c,d,e\}\}$. Lexicographic.
7. $11 \succ 20 \succ 01 \succ 10 \succ 00$. $R_1^7 = \{\{b\},\{a,d\},\{c,e\}\}$, $R_2^7 = \{\{a,c\},\{b,d,e\}\}$.
8. $11 \succ 20 \succ 10 \succ 01 \succ 00$. $R_1^8 = \{\{b\},\{a,c\},\{d,e\}\}$, $R_2^8 = \{\{a,d\},\{b,c,e\}\}$.
9. $20 \succ 11 \succ 01 \succ 10 \succ 00$. $R_1^9 = \{\{a\},\{b,d\},\{c,e\}\}$, $R_2^9 = \{\{b,c\},\{a,d,e\}\}$.
10. $20 \succ 11 \succ 10 \succ 01 \succ 00$. $R_1^{10} = \{\{a\},\{b,c\},\{d,e\}\}$, $R_2^{10} = \{\{b,d\},\{a,c,e\}\}$. Lexicographic.
11. $21 \succ 01 \succ 20 \succ 10 \succ 00$. $R_1^{11} = \{\{a,c\},\{d\},\{b,e\}\}$, $R_2^{11} = \{\{a,b\},\{c,d,e\}\}$. Lexicographic.
12. $21 \succ 11 \succ 01 \succ 20 \succ 00$. $R_1^{12} = \{\{a,d\},\{b\},\{c,e\}\}$, $R_2^{12} = \{\{a,b,c\},\{d,e\}\}$. Lexicographic.
13. $21 \succ 11 \succ 20 \succ 01 \succ 00$. $R_1^{13} = \{\{a,c\},\{b\},\{d,e\}\}$, $R_2^{13} = \{\{a,b,d\},\{c,e\}\}$.
14. $21 \succ 11 \succ 20 \succ 10 \succ 00$. $R_1^{14} = \{\{a,c\},\{b,d\},\{e\}\}$, $R_2^{14} = \{\{a,b\},\{c,d,e\}\}$. Lexicographic.
15. $21 \succ 11 \succ 01 \succ 10 \succ 00$. $R_1^{15} = \{\{a\},\{b,d\},\{c,e\}\}$, $R_2^{15} = \{\{a,b,c\},\{d,e\}\}$. Lexicographic.
16. $21 \succ 11 \succ 10 \succ 01 \succ 00$. $R_1^{16} = \{\{a\},\{b,c\},\{d,e\}\}$, $R_2^{16} = \{\{a,b,d\},\{c,e\}\}$. Lexicographic.
17. $21 \succ 20 \succ 01 \succ 10 \succ 00$. $R_1^{17} = \{\{a,b\},\{d\},\{c,e\}\}$, $R_2^{17} = \{\{a,c\},\{b,d,e\}\}$.
18. $21 \succ 20 \succ 10 \succ 01 \succ 00$. $R_1^{18} = \{\{a,b\},\{c\},\{d,e\}\}$, $R_2^{18} = \{\{a,d\},\{b,c,e\}\}$. Lexicographic.
19. $21 \succ 20 \succ 11 \succ 01 \succ 00$. $R_1^{19} = \{\{a,b\},\{c\},\{d,e\}\}$, $R_2^{19} = \{\{a,c,d\},\{b,e\}\}$. Lexicographic.
20. $21 \succ 20 \succ 11 \succ 10 \succ 00$. $R_1^{20} = \{\{a,b\},\{c,d\},\{e\}\}$, $R_2^{20} = \{\{a,c\},\{b,d,e\}\}$. Lexicographic.

Similarly, note that problem (8) has the same solution $K = 3$, $z = (2,2,2)$ for both $m = 8$ and $m = 7$. Again, let us start with $m = 8$.

$m = |A| = 8$. Thus, $A = \{a,b,c,d,e,f,g,h\}$, $a \succ b \succ c \succ d \succ e \succ f \succ g \succ h$. In this case, all attributes/partitions are binary, and hence, there is a lot of symmetry: all attributes can be re-enumerated without loss. In particular, it is without loss of generality to assume that $v_1(1) > v_2(1) > v_3(1)$ (if $v_i(1) = v_j(1)$ for some i,j, then the number of indifference classes $m \leq 6$, in contradiction). Accordingly, $100 \succ 010 \succ 001 \succ 000$ and $111 \succ 110 \succ 101 \succ 011$. We also have $101 \succ 100$, hence there can be only two possibilities:

1. $111 \succ 110 \succ 101 \succ 100 \succ 011 \succ 010 \succ 001 \succ 000$. This order is induced by $v_1(1) = 4$, $v_2(1) = 2$, $v_3(1) = 1$. The minimal adapted language is $Q = \{Q_1, Q_2, Q_3\}$ with $Q_1 = \{\{a,b,c,d\}, \{e,f,g,h\}\}$, $Q_2 = \{\{a,b,e,f\},\{c,d,g,h\}\}$, and $Q_3 = \{\{a,c,e,g\},\{b,d,f,h\}\}$. This is a lexicographic language.

2. $111 \succ 110 \succ 101 \succ 011 \succ 100 \succ 010 \succ 001 \succ 000$. This order is induced by $v_1(1) = 4$, $v_2(1) = 3$, $v_3(1) = 2$. The minimal adapted language is $R = \{R_1, R_2, R_3\}$ with $R_1 = \{\{a,b,c,e\},\{d,f,g,h\}\}$, $R_2 = \{\{a,b,d,f\},\{c,e,g,h\}\}$, and $R_3 = \{\{a,c,d,g\},\{b,e,f,h\}\}$. This is not a lexicographic language.

$m = |A| = 7$. Thus, $A = \{a,b,c,d,e,f,g\}$, $a \succ b \succ c \succ d \succ e \succ f \succ g$. Using our results for $m = |A| = 8$, we get the following reduced orders and associated minimal adapted languages $L^k = \{L_1^k, L_2^k, L_3^k\}$:

1. $111 \succ 110 \succ 101 \succ 100 \succ 011 \succ 010 \succ 001$.
$L_1^1 = \{\{a,b,c,d\}, \{e,f,g\}\}$, $L_2^1 = \{\{a,b,e,f\},\{c,d,g\}\}$, $L_3^1 = \{\{a,c,e,g\},\{b,d,f\}\}$. Lexicographic.

2. $111 \succ 110 \succ 101 \succ 100 \succ 011 \succ 010 \succ 000$.
$L_1^2 = \{\{a,b,c,d\}, \{e,f,g\}\}$, $L_2^2 = \{\{a,b,e,f\},\{c,d,g\}\}$, $L_3^2 = \{\{a,c,e\},\{b,d,f,g\}\}$. Lexicographic.

3. $111 \succ 110 \succ 101 \succ 100 \succ 011 \succ 001 \succ 000$.
$L_1^3 = \{\{a,b,c,d\}, \{e,f,g\}\}$, $L_2^3 = \{\{a,b,e\},\{c,d,f,g\}\}$, $L_3^3 = \{\{a,c,e,f\},\{b,d,g\}\}$. Lexicographic.

4. $111 \succ 110 \succ 101 \succ 100 \succ 010 \succ 001 \succ 000$.
$L_1^4 = \{\{a,b,c,d\}, \{e,f,g\}\}$, $L_2^4 = \{\{a,b,e\},\{c,d,f,g\}\}$, $L_3^4 = \{\{a,c,f\},\{b,d,e,g\}\}$. Lexicographic.

5. $111 \succ 110 \succ 101 \succ 011 \succ 010 \succ 001 \succ 000$.
$L_1^5 = \{\{a,b,c\}, \{d,e,f,g\}\}$, $L_2^5 = \{\{a,b,d,e\},\{c,f,g\}\}$, $L_3^5 = \{\{a,c,d,f\},\{b,e,g\}\}$. Lexicographic.

6. $111 \succ 110 \succ 100 \succ 011 \succ 010 \succ 001 \succ 000$.
$L_1^6 = \{\{a,b,c\}, \{d,e,f,g\}\}$, $L_2^6 = \{\{a,b,d,e\},\{c,f,g\}\}$, $L_3^6 = \{\{a,d,f\},\{b,c,e,g\}\}$. Lexicographic.

7. $111 \succ 101 \succ 100 \succ 011 \succ 010 \succ 001 \succ 000$.
$L_1^7 = \{\{a,b,c\}, \{d,e,f,g\}\},\ L_2^7 = \{\{a,d,e\},\{b,c,f,g\}\},\ L_3^7 = \{\{a,b,d,f\},\{c,e,g\}\}$. Lexicographic.

8. $110 \succ 101 \succ 100 \succ 011 \succ 010 \succ 001 \succ 000$.
$L_1^8 = \{\{a,b,c\}, \{d,e,f,g\}\},\ L_2^8 = \{\{a,d,e\},\{b,c,f,g\}\},\ L_3^8 = \{\{b,d,f\},\{a,c,e,g\}\}$. Lexicographic.

9. $111 \succ 110 \succ 101 \succ 011 \succ 100 \succ 010 \succ 001$.
$L_1^9 = \{\{a,b,c,e\},\{d,f,g\}\},\ L_2^9 = \{\{a,b,d,f\},\{c,e,g\}\},\ L_3^9 = \{\{a,c,d,g\},\{b,e,f\}\}$.

10. $111 \succ 110 \succ 101 \succ 011 \succ 100 \succ 010 \succ 000$.
$L_1^{10} = \{\{a,b,c,e\},\{d,f,g\}\},\ L_2^{10} = \{\{a,b,d,f\},\{c,e,g\}\},\ L_3^{10} = \{\{a,c,d\},\{b,e,f,g\}\}$.

11. $111 \succ 110 \succ 101 \succ 011 \succ 100 \succ 001 \succ 000$.
$L_1^{11} = \{\{a,b,c,e\},\{d,f,g\}\},\ L_2^{11} = \{\{a,b,d\},\{c,e,f,g\}\},\ L_3^{11} = \{\{a,c,d,f\},\{b,e,g\}\}$.

12. $111 \succ 110 \succ 011 \succ 100 \succ 010 \succ 001 \succ 000$.
$L_1^{12} = \{\{a,b,d\},\{c,e,f,g\}\},\ L_2^{12} = \{\{a,b,c,e\},\{d,f,g\}\},\ L_3^{12} = \{\{a,c,f\},\{b,d,e,g\}\}$.

13. $111 \succ 101 \succ 011 \succ 100 \succ 010 \succ 001 \succ 000$.
$L_1^{13} = \{\{a,b,d\},\{c,e,f,g\}\},\ L_2^{13} = \{\{a,c,e\},\{b,d,f,g\}\},\ L_3^{13} = \{\{a,b,c,f\},\{d,e,g\}\}$.

14. $110 \succ 101 \succ 011 \succ 100 \succ 010 \succ 001 \succ 000$.
$L_1^{14} = \{\{a,b,d\},\{c,e,f,g\}\},\ L_2^{14} = \{\{a,c,e\},\{b,d,f,g\}\},\ L_3^{14} = \{\{b,c,f\},\{a,d,e,g\}\}$.

Calculating the number of languages with profile solving problem (8) in the paper. As usual, we analyze the case when $\succeq$ is a linear order, so $m = |A|$. Consider first m such that the constraint in the minimization problem (8) in the paper binds. Thus, the solution contains k 3-valued attributes, r 2-valued attributes, and $m = 3^k \cdot 2^r$. We will prove the following Lemma:

Lemma B.1. *Let $\succeq$ be a linear order. Suppose that (K,z) solves problem (8) where m is the number of indifference classes of $\succeq$, so $m = |A|$. Suppose also that the number of coordinates with $z_i = 3$ is k, the number of coordinates with $z_i = 2$ is r, and $m = 2^r \cdot 3^k$. Then the total number of adequate languages $\{L_i\}_{i=1}^n$ with profile $(n,(\chi_i(L))_{i=1}^n) = (K,z)$ is given by*

$$\#\left\{L = \{L_i\}_{i=1}^n \middle| L \text{ is adequate and } (n,(\chi_i(L))_{i=1}^n) = (K,z)\right\} \quad = \quad \frac{m!}{2^{r+k} \cdot 3^k \cdot r! \cdot k!} \tag{B.1}$$

Proof. Denote the set of vectors of attributes by $Y = \{0,1,2\}^k \times \{0,1\}^r$. Since $m = 2^r \cdot 3^k$, there are $m!$ bijections $f : A \to Y$ between the items in A and the vectors of attributes. Denote the set of these bijections by $\mathscr{F}$. Note: equivalently, $\mathscr{F}$ is the set of linear orders on the set of vectors of attributes. Denote the set of adequate languages $\{L_i\}_i^n$ with profile $(n,(\chi_i(L))_{i=1}^n) = (K,z)$ by $\mathscr{L}$.

There is a natural mapping $L : \mathscr{F} \to \mathscr{L}$ given by $L_{ij}(f) = \{x \in A | f_i(x) = j\}$. Thus, the cell with index j of the partition L_i of the language $L(f)$ contains items whose i-th attribute has value j. Note that L is a surjection. Indeed, for any language $\{Q_i\}_{i=1}^n \in \mathscr{L}$, we can construct an associated mapping $f : A \to Y$ given by $f_i(x) = j$ such that $x \in Q_{ij}$. Let us show that f is an injection and hence, a bijection. Since Q is adequate, then $y \neq x$ implies that there exists a partition i such that $x \in Q_{ij}$ and $y \in Q_{ik}$ with $j \neq k$ and thus, $y \neq x$ implies that $f(y) \neq f(x)$. For $f \in \mathscr{F}$ constructed this way, we have $L(f) = Q$, proving that L is a surjection.

Since languages are defined up to permutations of partitions and cells in partitions, several bijections f can map to the same language $L \in \mathscr{L}$. In particular, consider the group $\mathscr{G} = \left(S_2\right)^r \times \left(S_3\right)^k \times S_r \times S_k$, where S_i is a permutation group of i numbers. Define the action of group $\mathscr{G}$ on Y naturally: each group S_2 in the product permutes values of a 2-valued attribute, each group S_3 in the product permutes values of a 3-valued attribute, the group S_r permutes the indexes of the 2-valued attributes, and the group S_k permutes the indexes of the 3-valued attributes. Similarly, $\mathscr{G}$ acts on $\mathscr{F}$ accordingly via

$$(G \circ f)(x) = G \circ (f(x)) \quad \text{for } G \in \mathscr{G},\ f \in \mathscr{F},\ x \in A \tag{B.2}$$

Accordingly,

$$L(f) = L(h) \text{ for some } f, h \in \mathscr{F} \iff \text{there exists } G \in \mathscr{G} \text{ such that } h = G \circ f \tag{B.3}$$

Therefore, the languages $L \in \mathscr{L}$ are the orbits of the group $\mathscr{G}$ acting on the set (of linear orders) $\mathscr{F}$. By Burnside's lemma,

$$|\mathscr{L}| = |\mathscr{F}/\mathscr{G}| = \frac{1}{|\mathscr{G}|} \cdot \sum_{G \in \mathscr{G}} |\mathscr{F}^G| \tag{B.4}$$

where $\mathscr{F}^G$ is the set of fixed points of $G \in \mathscr{G}$. A bijection $f \in \mathscr{F}$ is a fixed point of $G \in \mathscr{G}$ if and only if $G \circ f(x) = f(x)$ for all $x \in A$. Since f is a bijection, this means that $G \circ y = y$ for all $y \in Y$. Let us show that G must then be the identity permutation $I \in \mathscr{G}$. Towards a contradiction, assume $G \neq I$. If G permutes a value of any attribute, then either $G \circ (0, ..., 0) \neq (0, ..., 0)$ or $G \circ (1, ..., 1) \neq (1, ..., 1)$ (if G permutes values 1 and 2 of a 3-valued attribute), contradicting $G \circ y = y$ for all $y \in Y$. Thus, G does not permute values of any attributes. Since $G \neq I$, G must permute at least one attribute i to another attribute $k \neq i$. Take $y \in Y$ with $y_i = 0$ and $y_k = 1$, then $G \circ y \neq y$, contradicting $G \circ y = y$ for all $y \in Y$. We conclude that $\mathscr{F}^I = \mathscr{F}$, and $\mathscr{F}^G = \varnothing$ for any $G \neq I$. Put differently, $\mathscr{G}$ acts freely on $\mathscr{F}$. Therefore,

$$|\mathscr{L}| = \frac{|\mathscr{F}|}{|\mathscr{G}|} = \frac{|S_m|}{|S_2|^r \cdot |S_3|^k \cdot |S_r| \cdot |S_k|} = \frac{m!}{(2!)^r \cdot (3!)^k \cdot r! \cdot k!} = \frac{m!}{2^{r+k} \cdot 3^k \cdot r! \cdot k!} \tag{B.5}$$

proving the lemma. □

Lemma B.1 allows us to calculate the number of adequate languages with profile $(n,(\chi_i(L))_{i=1}^n) = (K,z)$ for $m = 2,3,4,6,8$. In particular:

- $m = 2$, then $r = 1$, $k = 0$, $|\mathscr{L}| = \dfrac{2!}{2!\cdot 1\cdot 1\cdot 1} = 1$;
- $m = 3$, then $r = 0$, $k = 1$, $|\mathscr{L}| = \dfrac{3!}{1\cdot 3!\cdot 1\cdot 1} = 1$;
- $m = 4$, then $r = 2$, $k = 0$, $|\mathscr{L}| = \dfrac{4!}{(2!)^2\cdot 1\cdot 2!\cdot 1} = 3$;
- $m = 6$, then $r = 1$, $k = 1$, $|\mathscr{L}| = \dfrac{6!}{2!\cdot 3!\cdot 1\cdot 1} = 60$;
- $m = 8$, then $r = 3$, $k = 0$, $|\mathscr{L}| = \dfrac{8!}{(2!)^3\cdot 1\cdot 3!\cdot 1} = 7\cdot 6\cdot 5\cdot 4 = 840$.

To calculate this number for $m = 5,7$, it is possible to push the combinatorial/group theory arguments discussed above further. Instead of doing this, I use a direct calculation using an algorithm developed to calculate the total number of minimal adequate languages by a direct count. The corresponding algorithm $\mathscr{A}$ is described in the next section.

Calculating the number of minimal adequate languages. To calculate the number of minimal adequate languages for $2 \le m = |A| \le 7$, I develop the corresponding algorithm and execute it using the attached MATLAB code.

The algorithm uses the following representation of an adequate language. Consider $A = \{1,...,m\}$, $1 \succ 2 \succ ... \succ m$ (the exact order of items is unimportant; importantly, $\succeq$ is a linear order). Let $E = \{e_1,\dots,e_M\} = \{(x,y) \in A\times A \mid x < y\}$ be the set of pairs of items in A, where $M = m(m-1)/2$. Let $\mathscr{P}$ be the set of partitions of A without the trivial partition A. Let us encode partitions via the restricted growth strings: each partition corresponds to a unique m-dimensional vector $(d_1,...,d_m)$ with coordinates $d_i \in \mathbb{Z}_+$ such that $d_k \le \max_{j<k} d_j + 1$. Enumerate all partitions naturally: let $w(d) \in \{1,2,...,|\mathscr{P}|\}$ be the index of partition $d \in \mathscr{P}$, then $w(d) < w(f)$ if and only if there is $l \le m$ such that $d_j = f_j$ for all $j < l$ and $d_l < f_l$. A partition $d \in \mathscr{P}$ separates a pair of items $e = (j,k)$ if $d_j \ne d_k$. For every $d \in \mathscr{P}$, denote $C(d) = \{(j,k) \in E | d_j \ne d_k\}$; thus, $C(d)$ is the set of pairs of items separated by d. Therefore, a language L is adequate if $\bigcup_{d\in L} C(d) = E$. Let us describe the algorithm formally.

The algorithm $\mathscr{A}$ takes as an input a natural number $m \geq 2$ and constructs the following tree $G(t)$ recursively in steps $t = 1, 2, \ldots$, given the root vertex $v_\varnothing$ with the convention that $v_\varnothing$ is created in step $t = 0$. When a successor vertex v is added to the tree in step $t - 1$, the algorithm assigns the sets $\text{Covered}(v) \subseteq E$, $\text{Used}(v) \subseteq \mathscr{P}$, $\text{Forbidden}(v) \subseteq \mathscr{P}$, and the immediate predecessor vertex $P(v)$ in the manner described below. The algorithm also assigns a number $\text{MAdeq}(v)$ to the vertex v in the manner described below. For the root vertex $v_\varnothing$ at the start of period $t = 1$, $\text{Covered}(v_\varnothing) = \varnothing$, $\text{Used}(v_\varnothing) = \varnothing$, $\text{Forbidden}(v_\varnothing) = \varnothing$, and $\text{MAdeq}(v_\varnothing) = 0$.

At the start of each step $t = 1, 2, \ldots$ when the algorithm runs, exactly one vertex v of the tree $G(t)$ is *open*, and others are *closed*; and if all vertices are not open, that is, all vertices are *closed*, the algorithm stops. At the start of step $t = 1$, the tree $G(t)$ consists of the root vertex $v_\varnothing$, and it is open. In each step $t = 1, 2, \ldots$, for the open vertex v, there are several cases:

Case 1. $\text{Covered}(v) = E$. The algorithm $\mathscr{A}$ runs another algorithm $\mathscr{B}$ that takes as an input the language $\text{Used}(v)$ and determines if $\text{Used}(v)$ is a minimal adequate language under the assumption that $\text{Used}(v)$ is adequate. If yes, $\mathscr{A}$ assigns $\text{MAdeq}(v) = 1$ and if no, $\mathscr{A}$ assigns $\text{MAdeq}(v) = 0$. Then, if $v \neq v_\varnothing$, $\mathscr{A}$ closes the vertex v, opens its immediate predecessor vertex $P(v)$ in the tree $G(t)$, updates the value of $\text{MAdeq}(P(v))$ to $\text{MAdeq}(P(v)) + \text{MAdeq}(v)$ and goes to step $t + 1$. If $v = v_\varnothing$, $\mathscr{A}$ closes it, stops and returns $\text{MAdeq}(v_\varnothing)$.

Case 2. $\text{Covered}(v) \neq E$. In this case, $\mathscr{A}$ finds a pair $e \in E$ with the lowest index that is not in the set $\text{Covered}(v)$, that is, $e = e_i$, where $i = \arg\min_i \{e_i \in E | e_i \notin \text{Covered}(v)\}$. $\mathscr{A}$ then searches for the partition $d \in \mathscr{P} \setminus \text{Forbidden}(v)$ with the lowest index $w(d)$ such that $e \in C(d)$ if it exists. There are two cases:

(a) There is no partition $d \in \mathscr{P} \setminus \text{Forbidden}(v)$ such that $e \in C(d)$. If $v \neq v_\varnothing$, $\mathscr{A}$ closes the vertex v, opens its predecessor vertex $P(v)$, updates the value $\text{MAdeq}(P(v))$ to $\text{MAdeq}(P(v)) + \text{MAdeq}(v)$ and goes to step $t + 1$. If $v = v_\varnothing$, $\mathscr{A}$ closes it, stops and returns $\text{MAdeq}(v_\varnothing)$.

(b) There is a partition $d \in \mathscr{P} \setminus \text{Forbidden}(v)$ such that $e \in C(d)$. $\mathscr{A}$ finds a partition d with the lowest index $w(d)$ that satisfies this property. $\mathscr{A}$ creates a vertex u that is an immediate successor of the vertex v; thus, $\mathscr{A}$ assigns $P(u) = v$. Next, $\mathscr{A}$ assigns $\text{Covered}(u) = \text{Covered}(v) \cup C(d)$, $\text{Used}(u) = \text{Used}(v) \cup \{d\}$, and $\text{Forbidden}(u) = \text{Forbidden}(v) \cup \{d\}$. Then, $\mathscr{A}$, updates $\text{Forbidden}(v)$ to $\text{Forbidden}(v) \cup \{d\}$, closes vertex v, opens vertex u, and goes to step $t + 1$.

Lemma B.2. *For any $m \geq 2$, the algorithm $\mathscr{A}$ stops after finitely many steps T and returns the number of minimal adequate languages for the case when $\succeq$ is a linear order and $m = |A|$.*

Proof. Naturally, let us call a vertex u *successor* of the vertex v in $G(t)$ if v is the predecessor of the vertex u, that is, if $v = P(u)$. Each time when a new successor of a vertex v is created, the set Forbidden(v) enlarges by 1 partition. Since Forbidden(v) $\subseteq \mathscr{P}$, no vertex v can have more than $|\mathscr{P}|$ successors. Next, if u is a successor of v, then the set Covered(u) is strictly larger than the set Covered(v), since by Case 2.b, $e \in$ Covered(v) $\cup\, C(d) =$ Covered(u), and $e \notin$ Covered(v). Since Covered(v) $\subseteq E$ for any v, the depth of the tree $G(t)$, that is, the total number of times the predecessor operator P can be consecutively applied before reaching the root of the tree, cannot exceed $|E|$. Thus, the total number of vertices in $G(t)$ does not exceed $|\mathscr{P}|^{|E|} < \infty$. Next, if $\mathscr{A}$ does not create a new vertex in step t, then it must be either Case 1 or Case 2.a; if $\mathscr{A}$ does not stop, it opens a predecessor vertex. Since the depth of the tree cannot exceed $|E|$, $\mathscr{A}$ creates a new vertex at least one time every $|E|$ periods. Thus, the total number of periods that $\mathscr{A}$ runs does not exceed $|E| \cdot |\mathscr{P}|^{|E|}$, proving the first statement of the lemma.

Claim B.1. *The algorithm $\mathscr{A}$ stops in period $t > 1$ when the case is case 2.a and $v = v_\varnothing$.*

Proof of Claim B.1. Note that $\mathscr{A}$ can only stop at $v = v_\varnothing$, since otherwise, it either creates a new vertex and opens it, or opens a predecessor vertex. Next, in any period t, Covered($v_\varnothing$) $= \varnothing$, hence $\mathscr{A}$ cannot stop in case 1; thus, it can only stop at case 2.a, and it must be that $t > 1$ in this case, since if $t = 1$, Forbidden($v_\varnothing$) $= \varnothing$, and there is a partition $d \in \mathscr{P}$ that separates a pair e_1. □

Claim B.2. *If Covered(v) $= E$, then Used(v) is an adequate language.*

Proof of Claim B.2. Note that if Covered(v) $= E$, then $v \neq v_\varnothing$. Denote by $v_0, v_1, ..., v_k$, the predecessors of the vertex v, where $v_0 = v_\varnothing$, $v_k = v$, and $P(v_i) = v_{i-1}$ for $i = 1, ..., k$. Each vertex v_i has been created in some period t_i according to Case 2.b; let d^i be the associated partition. Inductively,

$$\text{Covered}(v) = \bigcup_{i=1}^{k} C(d^i) \;\text{ and }\; \text{Used}(v) = \{d^1, \dots, d^k\} \tag{B.6}$$

which proves that Used(v) is an adequate language. □

Claim B.3. *If $L = \{L_i\}_{i=1}^n$ is a minimal adequate language, then there is a vertex v of the final tree $G(T)$ such that Used(v) $= L$.*

Proof of Claim B.3. Note that once the vertex v is created and endowed with the set Used(v), the algorithm $\mathscr{A}$ never changes the set Used(v). Since $L = \{L_i\}_{i=1}^n$ is adequate, there is its partition with the lowest index w that separates the first pair e_1; without loss, this is partition L_1. By Claim B.1, $\mathscr{A}$ returns to state $v_\varnothing$ at steps $t_1, t_2, \dots$, where $t_1 = 1$. At step t_k, Forbidden($v_\varnothing$) $=$

$\{d^1,...,d^{k-1}\}$, where d^i are partitions that separate e_1 with $j < l \implies w(d^j) < w(d^l)$, and step 2.b finds partition $d^k \notin \{d^1,...,d^{k-1}\}$ with the lowest index that separates e_1. Thus, in some step t_k, $d^k = L_1$, $\mathscr{A}$ creates a vertex v with $P(v) = v_\varnothing$, Used$(v) = \{L_1\}$, Forbidden$(v) = \{L_1\}$, and Covered$(v) = C(L_1)$. If $L = \{L_1\}$, then Covered$(v) = C(L_1) = E$ and hence, Used$(v) = \{L_1\} = L$, proving the claim. Otherwise, we repeat the argument inductively in the number of predecessors of the vertex v.

Specifically, we want to show by induction on $l \le n$ that there is a vertex v and some enumeration of partitions of the language L such that Used$(v) = \{L_1,...,L_l\}$ and, in step t_1, $L_r \notin$ Forbidden(v) for all $r > l$. Assume this holds for l, and let us prove the induction step for $l+1$. A vertex v from the induction assumption has $l < n$ predecessors $P(v)$, $P^2(v) \equiv P(P(v))$, ..., $P^l(v)$ and is opened in steps $t_1 < t_2 < ... < t_k$; without loss, $k \ge 1$. Our goal is to show that there exists an appropriate enumeration of partitions of L such that there is $t \in \{t_1,...,t_k\}$ such that in period t, case 2.b is applied, $d = L_{l+1}$, and a vertex u is created such that $P(u) = v$, Used$(u) = \{L_1,...,L_l,L_{l+1}\}$, and for all $r > l+1$, $L_r \notin$ Forbidden(u).

Since L is minimal adequate, Covered$(v) = \bigcup_{i=1}^{l} C(L_i) \neq E$ for $l < n$. Accordingly, Case 2 is applied in step t_1. Let $e \in E \backslash$Covered(v) be the pair with the lowest index that the algorithm $\mathscr{A}$ considers in step t_1 in Case 2. Since in step t_1, $L_r \notin$ Forbidden(v) for any $r > l$, and the language L is adequate, there is $r > l$ such that L_r separates e, and $L_r \notin$ Forbidden(v). Choose enumeration of partitions in L such that L_{l+1} is such a partition with the lowest index $w(\cdot)$ that satisfies this condition. If $\mathscr{A}$ does not pick L_{l+1} in step t_1, it picks some other partition d with a lower index that is not in Forbidden$(v) \supseteq$ Used(v) and hence, by construction, $d \notin L$. Applying this logic inductively, $\mathscr{A}$ picks partitions $d^1, d^2, ... \notin L$ in steps $t_1, t_2, ...$, adding these partitions to the set Forbidden(v) until in some step $t_i \in \{t_1,...,t_k\}$, $\mathscr{A}$ picks the partition L_{l+1} and creates a vertex u with $P(u) = v$ and Used$(u) = \{L_1,...,L_l,L_{l+1}\}$. We then have Forbidden$(u) =$ Forbidden$(v) \cup \{d^1,...,d^{i-1},L_{l+1}\}$. Since $d^1,...d^{i-1} \notin L$, and $L_r \notin$ Forbidden(v) for any $r > l$ by the induction assumption, then $L_r \notin$ Forbidden(u) for any $r > l+1$, proving the induction step.

Note that we have already proved the induction base in the first paragraph. Therefore, the inductive argument is valid, and at some step t, a vertex v is created with Used$(v) = L$. □

Claim B.4. *There are no two distinct vertices u and v such that Used(v) = Used(u).*

Proof of Claim B.4. Towards a contradiction, assume there are vertices v, u, $u \neq v$ with Used$(u) =$ Used$(v) = L$, then it must be that $u, v \neq v_\varnothing$, since Used$(v) = \varnothing$ if and only if $v = v_\varnothing$. Let $v_0, v_1, ..., v_j$

and $u_0, u_1, ..., u_l$ be the predecessors of the vertices v and u including the vertices themselves, such that $v_0 = v_\varnothing$, $v_j = v$, $P(v_i) = v_{i-1}$ for $i = 1, ..., j$, $u_l = u$, and $P(u_i) = u_{i-1}$ for $i = 1, ..., l$. Let k be the index of the first predecessor that is different for u and v; since $u \neq v$ and $v_0 = u_0$, such an index exists and is strictly greater than 0. Thus, $u_i = v_i$ for all $i < k$ and $u_k \neq v_k$. Denote by $q = u_{k-1} = v_{k-1}$. Let R be the set of partitions equal to Forbidden(q) in the first step when the vertex q is open. Note that Used(q) = $L \cap R$, and Used(q) $\neq L$; it must be then that Covered(q) $\neq E$.

Let $e \in E$ be the pair with the minimum index such that $e \notin$ Covered(q). Let d be the partition with the minimum index $w(d)$ that satisfies $d \in L \setminus$ Used(q) and $e \in C(d)$; such a pair exists since $\mathscr{A}$ creates vertices v_k and u_k. Without loss, $\mathscr{A}$ creates vertex v_k in an earlier step than vertex u_k. There are two possibilities: either $d \in$ Forbidden(q) in the step when v_k is created, in which case $d \in$ Forbidden(q) in the later step when u_k is created, or $d \notin$ Forbidden(q) in the step when v_k is created, in which case $\mathscr{A}$ picks d when it creates v_k and puts d in Forbidden(q). Therefore, in any case, in the step when vertex u_k is created, $d \in$ Forbidden(q), and thus, d cannot be added to the set Used(u) at this or any later stages, contradicting $d \notin$ Used(q) and $d \in L =$ Used(u). □

By Claim B.4, every vertex v is associated with a unique set of partitions Used(v). By Claim B.3, every minimal adequate language L is associated with one of the vertices $v(L)$ of the final tree $G(T)$, and by the previous argument, such a vertex is unique. Note that any such vertex must be a leaf of the tree and correspond to Case 1 of $\mathscr{A}$. Let u be a leaf vertex of the final tree $G(T)$. Since Covered(u) = E, the language Used(u) is adequate; therefore, the algorithm $\mathscr{B}$ can be correctly applied. Thus, every minimal adequate language L adds 1 to the count MAdeq($P(v(L))$) of the vertex $P(v(L))$ that is the immediate predecessor of the vertex $v(L)$; this follows from the fact that every leaf vertex of the tree $G(T)$ is opened only one time, since it does not have any successors. If Used(u) is not minimal adequate, then MAdeq(u) = 0 and MAdeq($P(u)$) is not updated when the vertex u is closed. Thus, in the final step T, the sum of MAdeq(q) over all vertices that are immediate predecessors of the leaf vertices is equal to the number of the minimal adequate languages. Note that for any non-leaf vertex v of $G(T)$, there is exactly one time when $\mathscr{A}$ visits a given successor branch u with $P(u) = v$, hence by Case 2.a of $\mathscr{A}$, in the final step T, for any non-leaf vertex v of $G(T)$, MAdeq(v) = $\sum_{u:\, P(u)=v}$ MAdeq(u) and hence, inductively, MAdeq($v_\varnothing$) is equal to the total number of minimal adequate languages. ■

The algorithm $\mathscr{B}$ takes as an input an adequate language $L = \{d^1, ..., d^n\}$ and verifies if it is minimal adequate or not. To do it, $\mathscr{B}$ goes over each of the partitions $d^i \in L$ and checks if $\bigcup_{j \neq i} C(d^j) \neq E$ for all i. If yes, the language L is minimal adequate, and if no, the language L is

not minimal adequate.

Finally, I have extended the algorithm $\mathscr{A}$ in such a way that for each minimal adequate language L that it finds, $\mathscr{A}$ verifies if the profile $(n,(\chi_i(L))_{i=1}^n)$ matches the solution of minimization problem (8) in the paper. This check is implemented by verifying if the added partition is 2-valued, 3-valued, or neither, counting this way the total number of 2-valued and 3-valued partitions for the resulting minimal adequate language, and comparing these numbers with the targets for a given m given by the solution of minimization problem (8).

The number of minimal adequate languages for $m \leq 7$ is calculated using a realization of the algorithm $\mathscr{A}$ in MATLAB; the code is attached. The bound for $m = 8$ is derived from monotonicity. Indeed, take any minimal adequate language for m items and, to each of its partitions, add a singleton cell containing the $(m+1)$-th item. The resulting language is minimal adequate for $m+1$ items, and this mapping is one-to-one, so the number of minimal adequate languages is monotone in m. Moreover, since the setup is symmetric with respect to permutations of items, the set of minimal adequate languages is symmetric with respect to permutations of items as well. In particular, for $m > 2$, there are minimal adequate languages with non-singleton cells, and hence there are minimal adequate languages in which item $m+1$ belongs to a non-singleton cell; such languages do not arise from the construction above, so the number of minimal adequate languages is strictly increasing in m.

B.3. Omitted calculations for Section 5.3.

Lemma B.3. *Consider an automaton σ that starts at state $s = 1$ each time a new item is drawn, and suppose that $q_\sigma(x) > 0$ for all $x \in A$. Let $M(B)$ be the expected time of choice from a menu B, and $M(x)$ be the expected time of choice from the menu $B = \{x\}$. Then:*

$$M(x) = \mathbb{E}[\textit{investigation of } x] \cdot q^{-1}(x) \qquad \textit{and} \qquad M(B) = \sum_{x \in B} \frac{\rho^B(x)q(x)}{\sum_{y \in B} \rho^B(y)q(y)} M(x) \tag{B.7}$$

where $\mathbb{E}[\textit{investigation of } x]$ is the expected time of investigation of item x before dismissal or choice.

Proof. Decompose any search history H that ends in a choice into rounds $H = (H_1, ..., H_N)$, where each H_i with $i < N$ ends in a dismissal (voluntary or involuntary), and H_N ends in the choice. Note that H can be considered as the first N elements of the sequence $H = (H_n)_{n \in \mathbb{N}}$ where H_n is a random history of a draw and single investigation ending in either dismissal or choice. Let $l_i = |H_i|$ be the length of round i, and $q(B)$ be the probability that some item from menu B is drawn and chosen during a single investigation; accordingly, $q(B) = \sum_{x \in B} \rho^B(x)q(x)$. Since with

each dismissal, the Markov chain re-starts from the same initial distribution, $\mathbb{E}[N|B] = q^{-1}(B)$. Next note that l_i are i.i.d. random variables with expected value

$$\mathbb{E}[l_i|B] = \sum_{x\in B} \rho^B(x)\mathbb{E}[\text{investigation of } x] \tag{B.8}$$

Since N is a stopping time for the filtration $(\mathscr{F}_n)_{n\in\mathbb{N}}$, where $\mathscr{F}_n$ is the sigma-algebra induced by events $H_1,...,H_n$, then Wald's identity applies, and

$$M(B) = \mathbb{E}[T|B] = \mathbb{E}[N|B]\cdot\mathbb{E}[l_i|B] = \frac{\sum_{x\in B}\rho^B(x)\mathbb{E}[\text{investigation of } x]}{\sum_{x\in B}\rho^B(x)q(x)} \tag{B.9}$$

Applying this formula to $B = \{x\}$, we get

$$M(x) = \mathbb{E}[\text{investigation of } x]\cdot q^{-1}(x) \tag{B.10}$$

Therefore,

$$M(B) = \sum_{x\in B}\frac{\rho^B(x)q(x)}{\sum_{y\in B}\rho^B(y)q(y)}M(x) \tag{B.11}$$

proving the lemma. □

Omitted calculations for eq. (18). Let us calculate $\mathbb{E}[\text{investigation of } x]$, the expected number of periods in one investigation of item x by an adapted automaton σ. Note that σ has $K = \lceil\log_2(n)\rceil$ states. Let $P(k,x)$ be the probability that σ has investigated item x in at least k states, then

$$\mathbb{E}[\text{investigation of } x] = \sum_{k=1}^{K} P(k,x) \tag{B.12}$$

where

$$P(k,x) = \prod_{s=1}^{k-1}(1-\eta)\tau(s,s+1,x_{\iota(s)}) \tag{B.13}$$

Choosing the transition probabilities of the adapted automaton σ as in the proof of Lemma 6,

$$P(k,x) = \prod_{s=1}^{k-1}(1-\eta)\epsilon^{\bar{v}_s - v_s(x)} = (1-\eta)^{k-1}\epsilon^{\sum_{s=1}^{k-1}(\bar{v}_s - v_s(x))} \tag{B.14}$$

Let $M(x)$ be the expected time of choice if the menu consists of a single item x, then

$$M(x) = \mathbb{E}[\text{investigation of } x]\cdot q_\sigma^{-1}(x) = \tag{B.15}$$

$$= \sum_{k=1}^{K}(1-\eta)^{k-1}\epsilon^{\sum_{s=1}^{k-1}(\bar{v}_s - v_s(x))}\cdot(1-\eta)^{-(K-1)}\cdot\epsilon^{-\sum_{s=1}^{K}(\bar{v}_s - v_s(x))} = \sum_{k=1}^{K}(1-\eta)^{-(K-k)}\epsilon^{-\sum_{s=k}^{K}(\bar{v}_s - v_s(x))}$$

from which it is clear that $M(x)$ is strictly decreasing in $v_s(x)$. Therefore, $M(x)$ attains a maximum over $x \in A$ at $x = w$ where w is the worst item in A.

By Lemma B.3, $M(B)$ is a convex combination of $M(x)$ with $x \in B$. Therefore,

$$\max_{B \in 2^A \setminus \{\varnothing\}} M(B) = \max_{x \in A} M(x) = M(w) \tag{B.16}$$

Denote by $\epsilon_i = \epsilon^{\bar{v}_i - v_i(w)}$ the probability that σ transitions from state i to the next state $i+1$ if the investigated attribute is w_i, then

$$P(k) = \prod_{i=1}^{k-1} (1-\eta)\epsilon_i \tag{B.17}$$

We also have $\epsilon = \dfrac{q_\sigma(w)}{(1-\eta)^{K-1}} = \prod_{i=1}^{K} \epsilon_i$. Note that the order of questions in σ can be chosen arbitrarily, and in particular, we can also make it such that $\epsilon_i \le \epsilon_k$ for all $i < k$. In other words, the first question corresponds to the largest potential probability penalty, then the second, etc. For fixed values of ϵ_i, this arrangement minimizes $\mathbb{E}[\text{investigation of } x] = \sum_{k=1}^{K} P(k)$. With this order of questions, $\mathbb{E}[\text{investigation of } x] = \sum_{k=1}^{K} P(k)$ assumes the largest value in the worst-case scenario when $\epsilon_i = \epsilon^{\frac{1}{K}}$ for all i. Thus,

$$\mathbb{E}[\text{investigation of } w] = \sum_{k=1}^{K} P(k) = \sum_{k=1}^{K} \prod_{i=1}^{k-1} (1-\eta)\epsilon_i \le \sum_{k=1}^{K} ((1-\eta)\cdot \epsilon^{\frac{1}{K}})^{k-1} = \frac{1-z^K}{1-z} \tag{B.18}$$

where $z = (1-\eta)\cdot\epsilon^{\frac{1}{K}}$. Recall from Appendix A.6 that the worst-case error of the Logit choice is at most $\ln(n)/(-\ln(\epsilon))$; hence, choosing $\epsilon = n^{-1/\Delta}$ guarantees a worst-case error of at most Δ. This gives $\epsilon^{-1} = n^{1/\Delta}$ and $\epsilon^{1/K} = n^{-1/(\Delta K)}$. Substituting, we have

$$\mathbb{E}[T] \le \frac{1-z^K}{1-z} \cdot (1-\eta)^{-(K-1)} \epsilon^{-1} \le \frac{1}{1-z} \cdot (1-\eta)^{-(K-1)} \epsilon^{-1} = \frac{n^{1/\Delta}(1-\eta)^{-(K-1)}}{1-(1-\eta)n^{-1/(\Delta K)}} \tag{B.19}$$

Finally, since $n^{1/\log_2(n)} = 2$, we have $n^{-1/(\Delta K)} = (2^{-1/\Delta})^{\frac{\log_2(n)}{\lceil \log_2(n) \rceil}} \approx 2^{-1/\Delta}$ and $(1-\eta)^{-(K-1)} \approx (1-\eta)^{-\log_2(n)}$; substituting these into the bound above yields the approximation $\dfrac{n^{1/\Delta}(1-\eta)^{-\log_2(n)}}{1-(1-\eta)/2^{1/\Delta}}$ in eq. (18).

Omitted calculations for eq. (19). For any $x \in A$, we have

$$M(x) \le \mu(\sigma) \cdot q_\sigma^{-1}(x) \le \mu(\sigma) \cdot q_\sigma^{-1}(w) = \mu(\sigma) \cdot (1-\eta)^{-(\mu(\sigma)-1)} \cdot \epsilon^{-1} \tag{B.20}$$

By Lemma B.3, $M(B)$ is a convex combination of $M(x)$ with $x \in B$, hence

$$\max_{B \in 2^A \setminus \{\varnothing\}} M(B) = \max_{x \in A} M(x) \le \mu(\sigma) \cdot q_\sigma^{-1}(w) = \mu(\sigma) \cdot (1-\eta)^{-(\mu(\sigma)-1)} \cdot \epsilon^{-1} \tag{B.21}$$

Substituting $\epsilon^{-1} \le n^{1/\Delta}$ and $\mu(\sigma) \le n-1 \le n$ yields eq. (19).

Calculations for a general automaton depicted in Figure 3. As σ_j, one can use an automaton introduced in the proof of Theorem 2 in Appendix A.4, where instead of the vanishing transitions, if $x = a^i$ with $i \le j$, the automaton transitions to the choosing state, and if $i > j$, the item is dismissed, and the automaton σ_j transitions back to its starting state with probability $1-\epsilon$ and to a starting state of automaton σ_{j+1} if $j < n-1$ or to the choice if $j = n-1$ with probability ϵ. Each σ_i contains at most $(n-1)\cdot n$ non-vanishing transitions and at most n vanishing transitions, hence no more than $(n-1)\cdot n + n = n^2$ transitions for each σ_i. Since there are $(n-1)$ automata σ_i, the total number of transitions that are not forced by the attention shock is $n^2(n-1)$. The total number of the transitions associated with the attention shocks depends on the convention to count those. If we have no more than 1 such transition per state (i.e., the transition does not depend on the value of the attribute investigated in a given state, as in the automaton considered in Figure 3), then the total number of these transitions is no more than the number of states, which is itself no more than $(n-1)^2$. Accordingly, $\kappa(\sigma) \le n^2(n-1) + (n-1)^2 = n^3 - 2n + 1 \le n^3$.

Let us now calculate a bound on the worst expected decision time for the automaton σ depicted in Figure 3. For a fixed $\epsilon > 0$, let

$$\delta_\epsilon(B) = \max_{x \in B} U(x) - \mathbb{E}[U(\text{choice made by } \sigma)] \tag{B.22}$$

Thus, $\delta_\epsilon(B)$ is the expected mistake that σ makes in menu B. Let $\bar{i}(B) = \min\{i | a^i \in B\}$. To establish an upper bound on the mistake, suppose that the utilities of items are chosen adversarially in order to maximize $\delta_\epsilon(B)$. Thus, $U(a^{\bar{i}(B)}) = 1$ and $U(a^j) = 0$ for all $j > \bar{i}(B)$. Clearly, if B does not contain some item a^j with $j > \bar{i}(B)$, adding a^j to the menu B increases the expected mistake in such adversarial case. Accordingly, we only need to analyze n menus $B^k = \{a^i \in A | i \ge k\}$ for $k = 1, 2, \ldots, n$. For the adversarial utilities, a direct calculation yields

$$\delta_\epsilon(B^k) \le \bar{\delta}_\epsilon(B^k) = g_\epsilon(n-k+1) \tag{B.23}$$

where $g_\epsilon(1) = 0$ since there is no mistake in a one-item menu, and for $r > 1$,

$$g_\epsilon(r) = 1 - \sum_{j=1}^{r-1} \left(\frac{1}{j + (r-j)\epsilon} \cdot \prod_{l=1}^{j-1} \frac{(r-l)\epsilon}{l + (r-l)\epsilon} \right) \tag{B.24}$$

To see this, note first that σ_i does not choose any item from a menu B^k with $k > i$. When the state goes to the block S^k, since the sampling probability of each item is $1/(n-k+1)$, the probability that a^k is chosen by σ^k is $\dfrac{1}{1+(n-k)\epsilon}$, and the probability that σ transitions to block S^{k+1} is $\dfrac{(n-k)\epsilon}{1+(n-k)\epsilon}$. For a block S^{k+j-1}, there are j items $x^k, \ldots, x^{k+j-1}$ that are chosen by σ_{k+j-1} with

probability one, and only one of these choices yields utility 1, hence the factor $\frac{1}{j+(n-k-j+1)\epsilon}$ in the corresponding summand. Finally, the probability that the automaton reaches block $k+j-1$ is, therefore, $\prod_{l=1}^{j-1}\frac{(n-k-l+1)\epsilon}{l+(n-k-l+1)\epsilon}$. Summing up the contribution from choices of x^k by σ^k, $\sigma^{k+1},...,\sigma^{n-1}$ yields the formula for $\bar{\delta}_\epsilon(B^k)$ above, which is valid for $r\geq 2$.

$$g_\epsilon(1)=0\leq\frac{\epsilon}{2},\qquad g_\epsilon(2)=\frac{\epsilon}{1+\epsilon}\leq\frac{2\epsilon}{2} \tag{B.25}$$

and for $r>2$, we have

$$g_\epsilon(r)\leq\frac{(r-1)\epsilon}{1+(r-1)\epsilon}\cdot\frac{1+(r-2)\epsilon}{2+(r-2)\epsilon}=\frac{(r-1)\epsilon}{2}\cdot\left[1-\frac{r\epsilon+(r-1)(r-2)\epsilon^2}{(1+(r-1)\epsilon)(2+(r-2)\epsilon)}\right]\leq\frac{(r-1)\epsilon}{2}\leq\frac{r\epsilon}{2} \tag{B.26}$$

Accordingly,

$$\Delta=\max_{B\in 2^A\setminus\{\varnothing\}}\delta_\epsilon(B)\leq\max_{k=1,...,n}\bar{\delta}_\epsilon(B^k)\leq\frac{n\epsilon}{2} \tag{B.27}$$

B.4. Special case: binary languages.

How important are languages with non-binary attributes? Theorem 1 shows that only ternary attributes may make some problems simpler in comparison to using only binary attributes. However, the ability to use languages with more than binary attributes may save a lot of memory load. Indeed, if a near optimal automaton uses only binary questions, it must ask at least $\lceil\log_2(m)\rceil$ questions to differentiate items in m indifference classes. Let us call a language L *binary* if all its attributes are binary; that is, $\chi_i(L)=2$ for all i; call any associated automaton *binary automaton.* Given $\succeq$, call a binary language L *binary adapted* if it contains an additive binary language $Q\subseteq L$ with $\lceil\log_2(m)\rceil$ partitions; call Q a *minimal binary adapted* language in this case, and an additive automaton associated with a minimal binary adapted language a *binary adapted automaton.* In the theorem below, the word "binary" is omitted, but all languages, adapted languages, automata, and adapted automata are assumed to be binary according to the definitions above.

Theorem B.1. *Let $\succeq$ have m indifference classes. Consider only binary languages; then:*

1. *For any language Q, $\kappa(Q)\geq 3\lceil\log_2(m)\rceil$, and there exists a language Q such that $\kappa(Q)=3\lceil\log_2(m)\rceil$;*
2. *For any language Q, $\mu(Q)\geq\lceil\log_2(m)\rceil$, and there exists a language Q such that $\mu(Q)=\lceil\log_2(m)\rceil$;*
3. *If σ is a near optimal automaton, and $\kappa(\sigma)=3\lceil\log_2(m)\rceil$, then $\mu(\sigma)=\lceil\log_2(m)\rceil$;*
4. *There exist an adapted language and an adapted automaton; any adapted language (adapted near optimal automaton) has complexity $3\lceil\log_2(m)\rceil$;*
5. *If $\frac{3}{4}\cdot 2^k<m\leq 2^k$, where $k=\lceil\log_2(m)\rceil$, then a language (near optimal automaton) has complexity*

$3\lceil \log_2(m) \rceil$ *if and only if it is adapted.*

All proofs of the results from Sections 3 and 4 work for the case of binary languages without much change. Comparing $\kappa(Q) \geq 3\lceil \log_2(m) \rceil$ and $\underline{\kappa}(m)$ given in eq. (9) in the paper, we can see that using non-binary languages may save 1 or 2 transitions in some cases. With binary languages, it is also the case that near optimal automata that reach the lower transitional complexity bound also reach the lower memory load bound, since $(K, z) = (\lceil \log_2(m) \rceil, (2, ..., 2))$ solves the minimization problem (8) in the paper constrained to binary languages ($z_i \leq 2$). The condition $\frac{3}{4} \cdot 2^k < m \leq 2^k$ relaxes condition $\frac{27}{32} \cdot 2^k < m \leq 2^k$ from Theorem 3, since the latter has been used to ensure that the minimization problem (8) in the paper has solution $(K, z) = (\lceil \log_2(m) \rceil, (2, ..., 2))$ without demanding that $z_i \leq 2$. For other arguments, the condition $\frac{3}{4} \cdot 2^k < m \leq 2^k$ is sufficient. An upper bound on complexity among binary languages is linear in the number of items: for any binary language Q, $\kappa(Q) \leq 3|A| - 3$, and there is a binary language R such that $\kappa(R) \geq \frac{|A|}{2}$; for example, language R from the proof of Theorem 2.

B.5. An Extension of Section 5.1.

The next proposition investigates the trade-off between the memory load and the lowest possible transitional complexity from a different angle than in Section 5.1. In particular, suppose that the agent can use no more than n memory states, but there is no other cost of memory usage. What is the lowest transitional complexity that the agent can achieve?

Proposition B.1. *Let the preference relation $\succeq$ have m indifference classes. Then for any $n \in \mathbb{N}$, if automaton σ is near optimal, and σ has at most n memory states, then $\kappa(\sigma) \geq f(m, n)$, and there is a language Q and a near optimal automaton σ with at most n memory states such that $\kappa(\sigma) = f(m, n)$, where*

$$f(m,n) \;=\; \min_{K \leq n,\, (z_i)_{i=1}^K \in \mathbb{N}^K} \left(-K + 2 \sum_{i=1}^{K} z_i \right) \qquad \text{such that} \quad \prod_{i=1}^{K} z_i \geq m \tag{B.28}$$

and the following approximation holds:

$$n \;\leq\; \log_{2.16}(m) \qquad \Longrightarrow \qquad f(m,n) \;\geq\; 2nm^{\frac{1}{n}} - n \tag{B.29}$$

The solution of problem (B.28) is very close to the bound given by eq. (B.29) and differs from it just because $2nm^{\frac{1}{n}} - n$ solves the problem if we allow z to take non-integer values, but in problem (B.28), all z_i should be natural numbers. If the complexity cost of an agent is a function of both the number of memory states and the number of transitions, the formulas above allow us

to analyze the lowest complexity cost, provided that the agent can use any questions (languages). One can see that in this case, the required number of transitions falls rapidly as the number of states increases from one to several.

Proof of Proposition B.1.. The proof of Proposition B.1 follows the same steps as the proof of Theorem 1.

To prove the approximation formula, note that if we relax the assumption that $z_i \in \mathbb{N}$ and allow for $z_i \in \mathbb{R}_{++}$ in the minimization problem (B.28), then the solution of the relaxed problem is K, $z_i = m^{\frac{1}{K}}$ where $K \in \mathbb{N}$ minimizes the function $h(K) = 2 \cdot K \cdot m^{\frac{1}{K}} - K$. Taking derivatives with respect to K gives us $h'(K) = 2 \cdot m^{\frac{1}{K}} \left(1 - \frac{\ln(m)}{K}\right) - 1$, $h''(K) = 2m^{\frac{1}{K}} \cdot \frac{\ln^2(m)}{K^3} > 0$. Hence, $h'(\cdot)$ is strictly increasing, and it suffices to show that $h'(n^*) \leq 0$ for $n^* = \log_{2.16}(m)$ to establish that $K = n$ achieves the minimum for the relaxed problem for all $n \leq n^*$. Substituting, we get $h'(n^*) = 2 \cdot 2.16 \cdot (1 - \ln(2.16)) - 1 = -0.0068... < 0$. Accordingly, n, $z_i = m^{\frac{1}{n}}$ solves the relaxed minimization problem with value $2nm^{\frac{1}{n}} - n$. The original minimization problem (B.28) has additional integer constraints and hence, $f(m,n) \geq 2nm^{\frac{1}{n}} - n$. ■