NBER WORKING PAPER SERIES

STRATEGYPROOFNESS-EXPOSING MECHANISM DESCRIPTIONS

Yannai A. Gonczarowski
Ori Heffetz
Clayton Thomas

Working Paper 31506
http://www.nber.org/papers/w31506

NATIONAL BUREAU OF ECONOMIC RESEARCH
1050 Massachusetts Avenue
Cambridge. MA 02138
July 2023

The authors thank Itai Ashlagi, Tilman Börgers, Eric Budish, Robert Day, Benjamin Enke, Xavier Gabaix, Nicole Immorlica, David Laibson, Jacob Leshno, Irene Lo, Kevin Leyton-Brown, Shengwu Li, Paul Milgrom, Michael Ostrovsky, Assaf Romm, Al Roth, Fedor Sandomirskiy, Ilya Segal, Ran Shorrer, Pete Troyan, Matt Weinberg, Leeat Yariv, Huacheng Yu, and participants in HUJI's JESC, Cornell's BERG, the 2020/2021 DIMACS Matching Reading Group, Harvard's Economic Theory, Behavioral Economics, and Computer Science Theory Seminars, Stanford GSB's Economics Seminar, WALE 2022, Noam Nisan's 60th Birthday Conference, the INFORMS Workshop on Market Design at EC 2022, MATCH-UP 2022, UMich's Economic Theory Seminar, INFORMS 2022, the 2022 NBER Market Design Working Group Meeting, the UTMD Rising Stars in Market Design workshop, UCI's ACO Seminar, UCSD's Economic Theory/Behavioral/Experimental Seminar, the Bellairs Workshop on Multi-Agent Systems, EC 2023, and the 2023 WZB Matching Market Design conference for helpful comments and discussions. We especially thank Guy Ishai for insights and invaluable assistance in preparing the wording of the experiments, and Itamar Bellaiche for his work on graphic design and general UI of the experiment. Tal Asif, Yehonatan Caspi, Ronny Gelman, Zoe Hitzig, and Yonatan Rahimi provided valuable research assistance. Gonczarowski gratefully acknowledges research support by the Harvard FAS Inequality in America Initiative; part of his work was carried out while at Microsoft Research. Heffetz gratefully acknowledges research support by the Israel Science Foundation (grant No. 2968/21) and Cornell's Johnson School; part of his work was carried out while visiting Princeton's School of Public and International Affairs. Thomas gratefully acknowledges the support of NSF CCF-1955205, a Wallace Memorial Fellowship in Engineering, and a Siebel Scholar award.The views expressed herein are those of the authors and do not necessarily reflect the views of the National Bureau of Economic Research.

Strategyproofness-Exposing Mechanism Descriptions
Yannai A. Gonczarowski, Ori Heffetz, and Clayton Thomas
NBER Working Paper No. 31506
July 2023
JEL No. D47,D82

ABSTRACT

A menu description presents a mechanism to player i in two steps. Step (1) uses the reports of other players to describe i's menu: the set of i's potential outcomes. Step (2) uses i's report to select i's favorite outcome from her menu. Can menu descriptions better expose strategyproofness, without sacrificing simplicity? We propose a new, simple menu description of Deferred Acceptance. We prove that—in contrast with other common matching mechanisms—this menu description must differ substantially from the corresponding traditional description. We demonstrate, with a lab experiment on two elementary mechanisms, the promise and challenges of menu descriptions.

Yannai A. Gonczarowski
Harvard University
Department of Economics and
Department of Computer Science
Cambridge, MA
yannai@gonch.name

Clayton Thomas
Princeton University
claytont@princeton.edu

Ori Heffetz
S.C. Johnson Graduate School of Management
Cornell University
324 Sage Hall
Ithaca, NY 14853
and The Hebrew University of Jerusalem
and also NBER
oh33@cornell.edu

1 Introduction

Strategyproof social-choice rules are often considered desirable. Under standard assumptions about players' preferences, these rules eliminate the need for players to strategize, since straightforward play is a dominant strategy.[1] In practice, however, real participants in strategyproof mechanisms often do not play these theoretically dominant strategies, suggesting they may not perceive them as dominant.[2]

Recent work takes a more refined approach, and suggests that the way a social-choice rule is *implemented* can influence straightforward play. This line of work proposes new extensive-form implementations that are dynamic and interactive, helping players avoid contingent-reasoning failures (Li, 2017), limited foresight (Pycia and Troyan, 2023), and expectations-based loss aversion (Dreyfuss et al., 2022b; Meisner and von Wangenheim, 2023). While already making a seminal contribution, this approach has limitations. For example, it may be theoretically restrictive (i.e., provably impossible for many desirable social-choice rules[3]) or impractical (e.g., calling upon each participant many times, each time halting the market until she makes a choice).

In this paper, we take a different approach: instead of searching for alternative implementations of a fixed social-choice rule, we investigate different *descriptions* of a fixed implementation of a fixed social-choice rule. We consider static, direct-revelation mechanisms, and propose a general framework—that we call *menu descriptions*—for presenting a mechanism to one player at a time in a way that may make strategyproofness easier to see. We motivate this framework with experiments, propose new descriptions, and prove theorems bounding the simplicity of such descriptions (according to context-relevant formal notions of simplicity).

Our main application is to Deferred Acceptance (henceforth DA) (Gale and Shapley, 1962). This matching mechanism is widely used in practice (Roth, 2002) to assign students to schools (Abdulkadiroğlu and Sönmez, 2003), medical residents to hospitals, and beyond. The traditional description of DA to participants is via the deferred

[1] Throughout this paper, we use the term "straightforward" to describe the strategy an agent would play under classic economic assumptions. While often referred to in past research as the "truthtelling" strategy, we avoid this morally laden term. Indeed, from the point of view of behavioral mechanism design, deviations from this classically optimal strategy should not be thought of as dishonesty.

[2] Evidence of such non-straightforward behavior comes both from the field (Hassidim et al., 2017, 2021; Shorrer and Sóvágó, 2017; Rees-Jones, 2018) and the lab (Kagel and Levin, 1993; Li, 2017; Chen and Sönmez, 2006; Hakimov and Kübler, 2021). It persists (somewhat ameliorated) even when the participants are explicitly informed of the strategyproofness of the mechanism (Guillen and Hakimov, 2018; Masuda et al., 2022).

[3] Of the five canonical social-choice rules we discuss in this introduction, only two have implementations in the theoretical framework of Li (2017): the second-price auction, and serial dictatorship.

acceptance algorithm.[4] However, this description may not intuitively expose the strategyproofness of DA, since showing its strategyproofness conventionally requires a delicate and technical mathematical proof. We present a novel, simple description of DA that makes its strategyproofness easy to see. Specifically, in our new description, strategyproofness holds by a short, one-sentence proof, while the simplicity of the description itself is comparable to that of the traditional one. Nonetheless, this new description obscures some properties that the traditional description of DA exposes. We prove that this is unavoidable by investigating a broad class of simple descriptions of DA, and proving formal tradeoffs between the different properties they can expose.

Table 1: Two pairs of descriptions of two strategyproof direct-revelation mechanisms.

(a) **Median Voting:** Two descriptions of the median voting mechanism over an ordered set of candidates, with three voters with single-peaked preferences.

Traditional Description:	Menu Description:
The three votes will be sorted from lowest to highest, and the *middle vote* of the three will be elected.	The "obtainable candidates" will be the votes of the other two players, and all candidates between them. Out of these "obtainable candidates," the one *closest to your own vote* will be elected.

Notes: For each voter, voting for her favorite candidate is a dominant strategy. Each voter's menu consists of all candidates between (and including) the votes of the other two voters.

(b) **Second-Price Auction:** Two descriptions of a single-item, sealed-bid, second-price auction.

Traditional Description:	Menu Description:
The player who placed the *highest bid* will win the item. She will pay a price equal to the *second highest bid.*	Your "price to win" the item will be set to the *highest bid* placed by any *other* player. If your bid is higher than this "price to win," then you will win the item and pay this price.

Note: Each bidder's menu consists of exactly two options: winning the item for a price equal to the highest bid of the other players, or winning nothing and paying nothing.

To make our framework concrete, Table 1 presents, for each of two classic mechanisms, two equivalent descriptions: a commonly used "Traditional Description" and a novel "Menu Description." While the former conveys the outcome in a direct way, the latter attempts to expose and emphasize strategyproofness. It does so by explaining the mechanism to a single player at a time, using the classic notion of a *menu* (Hammond, 1979). Formally, a player i's menu is defined as the set of all outcomes

[4] While "deferring acceptance" refers to properties of this algorithm, we follow popular usage and use DA to refer to the mechanism whose outcome is defined by this algorithm, i.e., to the mechanism that gives the one-side-optimal (e.g., student-optimal) stable matching (regardless of how this mechanism is described).

that i might possibly receive, given the reports of all other players. We define a *menu description* for player i as a description meeting the following two-step outline, which both examples from Table 1 follow:

Step (1) uses only the reports of other players to describe the set of outcomes player i might receive (i's menu).

Step (2) describes how to award player i her favorite outcome (according to her report) from her menu.

The first main premise of this paper is that menu descriptions provide a way to expose strategyproofness. While strategyproofness might be hard to infer from traditional descriptions of some mechanisms, it always holds for menu descriptions via a one-sentence proof: player i's menu in Step (1) cannot be affected by her report, and in Step (2), straightforward reporting guarantees her favorite outcome from the menu.

To begin the study of menu descriptions, we observe that *every* strategyproof mechanism has a menu description (Hammond, 1979). To see this, consider any description D of (the outcome of) the mechanism, and consider the following "brute force" menu description for player i:

Step (1): Iterate over all possible reports t_i' of player i, and let M denote the set of all outcomes for player i of the form $D(t_i', t_{-i})$.

Step (2): Award player i her favorite outcome (according to t_i) from M.

However, we suspect such descriptions may be impractical. We speculate that many participants would find them complicated, implausible, or confusing, precluding their real-world use.

Given the above, the second main premise of this paper is that *simplicity* of menu descriptions is paramount. What counts as a simple description is naturally subjective, multi-faceted, and context-dependent. As a guiding principle, we strive for menu descriptions that are not dramatically more complex than the corresponding traditional descriptions (which are typically the simplest known way to describe the outcome). In the main results of this paper, we first present new menu descriptions that we view as being nearly as simple as traditional ones. Second, we propose simplicity conditions that traditional descriptions (and our menu descriptions) meet, and use these conditions to prove theorems that rule out the existence of other possible simple (according to our conditions) menu descriptions beyond the ones that we present.

To empirically motivate using menu descriptions, we turn to the two elementary settings (with arguably simple descriptions) from Table 1, and explore how real participants respond to the descriptions in that table. We conduct a preregistered (between-subjects) lab experiment using descriptions similar to those in Table 1. For the (three player) median-voting mechanism, in a menu-description treatment we find higher rates of straightforward behavior (80%; $N = 100$) than in a traditional-description treatment (70%; $N = 100$; equality-of-means $p = 0.01$). Furthermore, we find that in the menu treatment, straightforward behavior is highly correlated with participants' *comprehension* of the mechanism, while in the traditional treatment it is not. This may suggest that for the menu description of this mechanism—but not for its traditional description—understanding how the outcome is calculated helps understand strategyproofness. In contrast, for the (five player) second-price auction we find no difference in straightforward behavior between the two treatments. This in turn may suggest that for some mechanisms, strategyproofness may be equally apparent from traditional and menu descriptions.

Our paper begins by providing preliminaries in Section 2. Next, we turn our attention to matching mechanisms, a common setting in which complicated mechanisms are used in practice to match *applicants* to *institutions*.[5] We start by considering our main application, DA (i.e., the applicant-optimal stable matching mechanism). In contrast to the mechanisms in Table 1, it is initially far from clear how to characterize the menu in DA.

In Section 3, we present our main positive result: a novel menu description of DA, summarized in Table 2. We view this description as being nearly as simple as the corresponding traditional description. In fact, the menu is calculated directly from a single run of the "flipped-side-proposing" deferred acceptance procedure (among all participants except some player i, to whom the mechanism is described). In sharp contrast with the traditional description of DA, strategyproofness can be readily seen from this menu description in the same way as from the menu descriptions in Table 1. We emphasize that it was previously unknown that the menu in DA can be characterized as in Table 2, or, indeed, in any other way that does not require *multiple* runs of deferred acceptance while going over different hypothetical reports by player i.

In Section 4, we consider two additional popular matching mechanisms, Serial Dictatorship (henceforth SD) and Top Trading Cycles (henceforth TTC). For SD, the traditional description *already is* a menu description. Namely, for each player i, the traditional description of SD can be written in the following *three* steps, the first

[5] In all matching mechanisms that we consider, the only strategic players are the applicants. The institutions are not strategic, and thus their preferences over the applicants are by convention called *priorities*.

Table 2: **Deferred Acceptance:** Two descriptions of the *applicant-optimal* stable matching mechanism.

Traditional Descr.:	Menu Description:
The applicants will be matched to institutions according to *applicant-proposing deferred acceptance*.	Imagine running *institution-proposing deferred acceptance* with all institutions and all applicants *except you*, to obtain a hypothetical matching. You "earn admission" at every institution that ranks you higher than its hypothetically matched applicant. You will be matched to the institution that you *ranked highest* out of those at which you will have earned admission.

Notes: Both descriptions use the traditional deferred acceptance procedure as a building block. We emphasize that in the menu description, the hypothetical match of other applicants is not necessarily their match in the actual outcome of DA.

two of which are a menu description:

(1) Using only the reports t_{-i} of other players, calculate i's menu.

(2) Match i to her top-ranked institution on her menu, according to i's report t_i.

(3) Using t_i and t_{-i}, proceed to calculate the rest of the matching (for all other applicants).

We call a description with this outline, i.e., any description of the full matching that contains a menu description for player i, an *individualized dictatorship* (see Table 3). Our main result in Section 4 shows that a simple individualized dictatorship for TTC exists, and in particular can be constructed via only a slight modification of the traditional description (namely, by specializing the order of the steps performed by the traditional description). The existence of this description for TTC is somewhat remarkable: a small tweak to the traditional description of TTC suffices to produce an individualized dictatorship, i.e., an alternative description for TTC whose strategyproofness is in principle as easy to see as that of SD.

We find the description presented in Section 3 (for DA) as appealing as that presented in Section 4 (for TTC). However, it is quite different in character; one may wonder whether, like with TTC, an alternative menu description of DA exists embedded within (a small tweak of) its traditional description. To examine this question, and the limits of simple descriptions more broadly, we define in Section 5 a general formal model of mechanism descriptions, particularly the algorithmic descriptions commonly used for matching mechanisms. While our models are informed by ideas from computer science, the complexity notions that we reason about are not standard computer-science complexity notions concerned with running algorithms on

Table 3: Different outlines for mechanism descriptions.

Menu description:	
Definition:	Description of one player's outcome fitting Steps (1) and (2) on page 3.
Application:	Describe one player's outcome while exposing strategyproofness for that player.
Motivation:	Strategyproofness holds by a simple proof: the player's report cannot affect her menu, and straightforward reporting gets the player her favorite outcome from the menu.
Individualized dictatorship description:	
Definition:	Description of the full outcome fitting Steps (1), (2), and (3) on page 5. (the first two of which constitute a menu description).
Application:	Describe the *full* outcome while exposing strategyproofness for one player.
Motivation:	Quantify (using simplicity conditions) when traditional descriptions can or cannot be tweaked to expose strategyproofness through menus.

computers. Rather, they are novel notions specifically tailored to capture attributes of algorithms that are used in practice to help explain the algorithms to non-experts.

In Section 6, we prove our main impossibility theorem: in contrast to SD and TTC, no individualized dictatorship for DA is similar to its traditional description. To formalize this claim, we delineate two fundamental properties of the traditional description of DA.[6] First, it is applicant-proposing (querying applicants' preferences in favorite-to-least-favorite order). Second, it describes the outcome by iteratively modifying a single tentative matching (without performing more complicated bookkeeping or rote memorization; formally, using a roughly linear amount of memory). We prove that any applicant-proposing individualized dictatorship description of DA must keep track of vastly more information than a single tentative matching; in fact, such a description must remember essentially all preferences of all applicants throughout its run (roughly equivalent to keeping track of n full matchings; formally, requiring a quadratic amount of memory). We thus interpret this result as showing that no small tweak of the traditional description of DA has a menu description embedded within it. This provides a formal sense in which (in contrast to SD and TTC) it is difficult to recognize that DA "has a menu"—and hence is strategyproof—from its traditional description.

In Section 7, we examine simple one-side-proposing descriptions of DA more broadly. We search for three classes of descriptions: First, menu descriptions, which expose strategyproofness but may obscure the fact that the matching collectively

[6] These properties are also shared by the descriptions of virtually any other popular matching mechanism.

described by all players' menu descriptions are feasible (i.e., that no two applicants infeasibly get the same assignment); second, descriptions of the full matching, which may make feasibility easy to see but may obscure strategyproofness; third, individualized dictatorships, which in these senses expose both properties. Our findings are summarized in Table 5 (on page 30). While we uncover certain additional descriptions, our most significant finding is an additional impossibility theorem: any conceivable description in our classification—other than traditional DA and our new menu description—must in a formal sense be a delicate and technical algorithm.[7] For simple descriptions of DA in our framework, there is thus a tradeoff between conveying feasibility and conveying strategyproofness. All traditional descriptions of DA have long been at one corner solution of this tradeoff. Our new menu description is at the other corner solution, exposing perhaps the most relevant consideration for the applicants: strategyproofness.

Table 4: Classification of main results.

Description Type		Interpretation of Description Type	Result
Outline	Simplicity Constraint		
Menu description	Informal$^{(*)}$: comparable in simplicity to the traditional description	Alternative, potentially practically simple description that exposes strategyproofness	Provide new descriptions for both DA (Sec. 3) & TTC (Sec. 4)
Individualized dictatorship	Formal: applicant-proposing & low-memory	Menu description embedded in (potentially) small tweak of the traditional description	Show possible for TTC (Sec. 4); Prove impossible for DA (Sec. 6)
Any outline type for DA	Formal: one-side-proposing & local bookkeeping	Broad class of simple descriptions exposing strategyproofness (menu descr.s), feasibility (outcome descr.s), or both (individ. dictatorships)	For DA: prove can convey strategyproofness or feasibility, but not both (Sec. 7)

(*) Note: As we discuss in Section 6 and Section 7, our new descriptions of TTC and DA also satisfy formal simplicity constraints (one-side-proposing, low-memory, local bookkeeping).

Table 4 summarizes our main results for matching mechanisms. In Section 8 we briefly consider an extension of our framework to multi-item auctions where bidders have additive or unit-demand valuations over items. We detail our experiment in Section 9, review related work in Section 10, and conclude in Section 11, where we also discuss limitations of our framework.

[7] Roughly speaking, such a description must update bookkeeping concerning some participants when making queries that seem unrelated to them, for example, learning that institution a is on the menu when the most recent preference query concerned institution b.

2 Preliminaries

2.1 Environments and Mechanisms

This paper studies strategyproof mechanisms, i.e., dominant-strategy direct-revelation implementations of social-choice rules, in different economic environments. We define all these terms fully in Appendix B. Briefly, an *environment* is defined by a set A of possible *outcomes*, a number n of *players* (sometimes referred to as agents), and a set of possible *types* $t_i \in \mathcal{T}_i$ for each player i. The type t_i of each player i completely defines her preferences $\succ_i^{t_i}$ over the outcomes.

Definition 2.1. A *social-choice rule* in an environment $\big(A, n, (\mathcal{T}_i)_{i=1}^n\big)$ is any mapping $f : \mathcal{T}_1 \times \ldots \times \mathcal{T}_n \to A$ from the types of all players to an outcome. A social choice function is *strategyproof* if, for every $t_i, t_i' \in \mathcal{T}_i$ and[8] $t_{-i} \in \mathcal{T}_{-i}$, we have

$$f(t_i, t_{-i}) \succeq_i^{t_i} f(t_i', t_{-i}).$$

We study *descriptions* of mechanisms, which are ways to convey the mechanism ex ante. To start, we consider *outcome descriptions*. An outcome description is a precise and unambiguous way of conveying the function from players' types to the outcome of the mechanism. In this paper, we typically consider outcome descriptions that provide an explicit set of instructions that one could use to calculate the outcome, i.e., an algorithm.

In matching environments, we call each player an *applicant*, and outcomes are matchings between applicants and *institutions* (such that no applicant is matched to two institutions, and vice versa). For our main mechanisms of interest, DA and TTC, we denote applicants by d (mnemonic: doctor), and institutions by h (mnemonic: hospital); for other mechanisms and environments we denote players (applicants or otherwise) by i. The type of each applicant is any strict ordinal preference over the institutions. We denote the type of applicant d as $\succ_d$, and write $h_1 \succ_d h_2$ if d prefers institution h_1 to institution h_2. Applicants may have partial preference lists, indicating that they prefer to remain unmatched over being matched to any institution not on their list. Only the applicants are strategic, whereas the institutions have fixed *priorities*, which are exogenously given strict ordinal preferences over applicants.

We study the canonical mechanisms of Serial Dictatorship (SD), Top Trading Cycles (TTC), and Deferred Acceptance (DA), defined as follows:

[8] As is standard, we write $\mathcal{T}_{-i}$ to denote the set $\mathcal{T}_1 \times \ldots \times \mathcal{T}_{i-1} \times \mathcal{T}_{i+1} \ldots \mathcal{T}_n$, and for $t_i \in \mathcal{T}_i$ and $t_{-i} \in \mathcal{T}_{-i}$, we write (t_i, t_{-i}) for the element of $\mathcal{T}_1 \times \ldots \times \mathcal{T}_n$ that naturally corresponds to t_i along with t_{-i}.

Definition 2.2. Serial Dictatorship (SD) is defined with priority order π (i.e., π is any bijection between $\{1, \dots, n\}$ and the applicants). The matching is produced as follows. In order $i = \pi(1), \pi(2), \dots, \pi(n)$, applicant i selects and is permanently matched to her favorite institution that has not already been selected by some preceding (in the order π) applicant.

Definition 2.3. Top Trading Cycles (TTC) is defined with respect to a profile of priority orders $\{\succ_h\}_h$, one for each institution h, over applicants. The matching is produced as follows. Repeat the following until everyone is matched (or have exhausted their preference lists): each remaining (i.e., not-yet-matched/exhausted) applicant points to her favorite remaining institution, and each remaining institution points to its highest-priority remaining applicant. There must be some cycle in this directed graph (as there is only a finite number of vertices).[9] Permanently match each applicant in this cycle to the institution to which she is pointing. (And these applicants and institutions do not participate in later iterations.)

Definition 2.4. Deferred Acceptance (DA) is defined with respect to a profile of priority orders $\{\succ_h\}_h$, one for each institution h, over applicants. The matching is produced as follows. Repeat the following until every applicant is matched (or has exhausted her preference list): A currently unmatched applicant is chosen to *propose* to her favorite institution which has not yet *rejected* her.[10] The institution then rejects every proposal except for the *top priority applicant* who has proposed to it thus far. Rejected applicants become (currently) unmatched, while the top priority applicant is tentatively matched to the institution. This process continues until no more proposals can be made, at which time the tentative allocations become final.

Note that DA refers to the (direct-revelation) mechanism defined by applicant-proposing DA (i.e., outputting the applicant-optimal stable matching); when confusion might arise, we use APDA (and for institution-proposing, we use IPDA). For additional formal discussion on these definitions, see Appendix B.

For each of SD, TTC, and DA, we refer to the algorithm in the above definition as the *traditional description* of the mechanism; for one real-world example conveying this description for DA, see Section 6.1.

In matching environments, applicants are assumed to only care about their own match, and thus the preferences of applicant d over matchings μ depend only on $\mu(d)$.

[9] The final outcome of TTC is independent of which cycle is chosen at every step.

[10] The final outcome of DA is independent of which unmatched applicant proposes at every step.

In many other environments (especially those without externalities), each player similarly does not care about the entire outcome, but only about some part of it. We thus define the *i-relevant outcome sets* (or *i-outcomes* for short) of an environment as the equivalence classes of outcomes for which i is indifferent between these outcomes for all possible types of player i. (For a formal definition, see Definition B.2.) We denote the i-outcome corresponding to an outcome a by $[a]_i$, and the set of i-outcomes by A_i. For example, in matching environments, $[\mu]_d = \{\mu' \mid \mu'(d) = \mu(d)\}$.

2.2 Menu Descriptions

The central notions of our paper are the definition of a player's *menu* in a mechanism, and the alternative definition of strategyproofness that the concept of a menu provides. Briefly, player i's menu with respect to $t_{-i} \in \mathcal{T}_{-i}$ is the set of outcomes that player i might receive when other players have types t_{-i}, and a mechanism is strategyproof if and only if each player always gets her favorite outcome from her menu.[11]

Definition 2.5. For any social choice rule f, the *menu* $\mathcal{M}_{t_{-i}}$ of player i with respect to types $t_{-i} \in \mathcal{T}_{-i}$ is the subset of all i-outcomes $a_i \in A_i$ for which there exists some $t_i \in \mathcal{T}_i$ such that $f(t_i, t_{-i}) \in a_i$. That is,

$$\mathcal{M}_{t_{-i}} = \{\ [f(t_i, t_{-i})]_i \mid t_i \in \mathcal{T}_i\} \subseteq A_i.$$

Theorem 2.6 (Hammond, 1979)**.** *A social choice rule f is strategyproof if and only if each player i always receives (one of) her favorite i-outcomes from her menu. That is, for every $t_{-i} \in \mathcal{T}_{-i}$ and $t_i \in \mathcal{T}_i$, it holds that $f(t_i, t_{-i}) \succeq_i^{t_i} x$ for any $x \in \mathcal{M}_{t_{-i}}$.*

Proof. Suppose f is strategyproof and fix $t_{-i} \in \mathcal{T}_{-i}$. For every $t_i \in \mathcal{T}_i$, it holds by definition that player i will always prefer $[f(t_i, t_{-i})]_i$ at least as much as any other i-outcome $[f(t'_i, t_{-i})]_i$ on the menu. On the other hand, if player i always receives her favorite i-outcome from her menu, then she always prefers reporting t_i at least as much as any t'_i, so f is strategyproof. □

A *menu description* of f for player i first calculates the menu of player i using t_{-i},

[11] Related or equivalent versions of Definition 2.5 have been considered under many different names in many different contexts (e.g., *sets that decentralize the mechanism* in Hammond (1979); *option sets* in Barberà et al. (1991); *proper budget sets* in Leshno and Lo (2021); *feasible sets* in Katuščák and Kittsteiner (2020); and likely others). We follow the "economics and computation" literature (Hart and Nisan, 2017; Dobzinski, 2016; and follow-ups) in calling these sets "menus." This definition is distinct from many other definitions of menus such as those in (Mackenzie and Zhou, 2022; Bó and Hakimov, 2023, and many others).

and then selects i's favorite i-outcome in the menu using t_i.[12] The first premise of this paper is that menu descriptions are one way to expose strategyproofness. This is because any menu description can be immediately proven strategyproof, by a simple, one-sentence proof: Player i's report cannot affect her menu, and straightforward reporting ("truthtelling") gets player i's favorite outcome from the menu.

In matching mechanisms, the menu of an applicant is simply the subset of institutions she can get given other applicants' preferences (and all institutions' priorities). For this domain, there is a formal sense in which menu descriptions are the *only* way for strategyproofness to be proven via the simple proof outline above.

Proposition 2.7. *Consider some applicant i. Fix t_{-i}, let S denote some set of institutions, and suppose that for every possible t_i, the institution in S that t_i ranks highest is $[f(t_i, t_{-i})]_i$. Then $S = \mathcal{M}_{t_{-i}}$.*

Proof. Since $[f(t_i, t_{-i})]_i \in S$ for each $t_i \in \mathcal{T}_i$, we have $\mathcal{M}_{t_{-i}} \subseteq S$. Now, if there exists some institution $a \in S \setminus \mathcal{M}_{t_{-i}}$, then for any type t_i that ranks a first, t_i's favorite institution from S cannot possibly be $[f(t_i, t_{-i})]_i \in \mathcal{M}_{t_{-i}}$. So no such a can exist, and hence $S \subseteq \mathcal{M}_{t_{-i}}$. □

Thus, for any two-step description that calculates a set S of institutions in Step (1) and matches i to one of these institutions in Step (2), if strategyproofness follows from the simple proof outline above, then the description must be a menu description. For domains other than matching, an analogous result holds, except that the set S can also contain "dominated" i-outcomes that no type prefers to all elements of $\mathcal{M}_{t_{-i}}$.

This paper seeks menu descriptions of matching mechanisms that are as simple as traditional descriptions. To start, we give a *non-simple* menu description as a baseline:[13]

Example 2.8 (A *non*-simple menu description for any strategyproof matching mechanism)**.** Consider any strategyproof matching mechanism f for n applicants and n institutions, and fix an applicant i. For each institution h, let $\{h\}$ denote the preference list of applicant i that ranks only h (indicating that all other institutions are unacceptable). Let D be any outcome description of f; for concreteness, suppose that D is an algorithm for calculating the outcome matching of f given all applicants' types. Then, the following is a menu description for applicant i:

[12] In some domains, more than one i-outcome can be tied for i's favorite according to some t_i (for example, in a single item auction, a bidder is indifferent between not receiving the item and receiving the item for a price equal to their value). In this case, the menu description must still assign player i to $[f(t_i, t_{-i})]_i$. In other words, the menu description must follow the same tiebreaking rules as f.

[13] An equivalent description was independently given by Katuščák and Kittsteiner (2020).

(1) Using $\succ_{-i}$, evaluate D on each type profile of the form $(\{h\}, \succ_{-i})$ for each institution h separately. Let $\mathcal{M}$ be the set of all institutions h such that i is matched to h at the end of some evaluation of D.

(2) Using $\succ_i$, match applicant i to her highest-ranked institution in $\mathcal{M}$.

By strategyproofness, h will be included in M in Step (1) if and only if h is on the menu. Thus, Example 2.8 provides a menu description of f. However, we believe there are disadvantages to using such a description in practice. Namely, we speculate that real people would find this description far less natural and plausible than traditional descriptions, and possibly even confusing. In Example 2.8, any information about i's menu is acquired separately for each institution by completely restarting description D. This stands in contrast to traditional descriptions of DA and TTC, which (as discussed further in Section 6.1) incorporate each part of the preferences of each applicant at most once. All in all, we consider a description similar to that in Example 2.8 "complex," and we do not recommend that real-life clearinghouses adopt an approach as in Example 2.8. Instead, the second main premise of this paper is to look for simpler menu descriptions, and in particular, menu descriptions that seem nearly as simple as the corresponding traditional descriptions (which by their nature are typically the simplest currently known way to describe the mechanism to participants).

3 A Simple Menu Description of DA

We start with our main positive result: a menu description of (applicant-optimal) Deferred Acceptance (DA), presented in Description 1 (and also in Table 2 in the introduction). We view Description 1 as being nearly as algorithmically simple as the traditional description of DA; in fact, it only adds an easy-to-state "menu calculation and matching" step on top of a traditional description of DA.

Description 1 A menu description of DA for applicant d

(1) Run *institution*-proposing DA with applicant d removed from the market, to get a matching μ_{-d}. Let M be the set of institutions h such that $d \succ_h \mu_{-d}(h)$.

(2) Match d to d's highest-ranked institution in M.

Crucially, Description 1 uses *institution*-proposing DA to calculate an applicant's menu in the applicant-optimal DA outcome (traditionally described via applicant-proposing DA). To get some intuition for why this is the case, consider a market with three applicants d_*, d_1, d_2 and two institutions h_1, h_2. Applicants have preferences $d_1 : h_1 \succ h_2$ and $d_2 : h_2 \succ h_1$, and institutions have priorities $h_1 : d_2 \succ d_* \succ d_1$ and $h_2 : d_1 \succ d_* \succ d_2$. Running applicant-proposing DA on these preferences without d_* gives matching $\{(d_1, h_1), (d_2, h_2)\}$, and both h_1 and h_2 prefer d_* to their match. However, neither h_1 nor h_2 are on d_*'s menu, since having d_* propose to any $h_i \in \{h_1, h_2\}$ (after running applicant-proposing DA without d_*) causes a "rejection cycle" that results in h_i rejecting d_*. In contrast, institution-proposing DA outputs a matching that has no potential applicant-proposing rejection cycles (sometimes also referred to as institution-improving rotations, see Irving and Leather, 1986).

Formally, the following theorem establishes the correctness of Description 1:

Theorem 3.1. *Description 1 is a menu description of DA. In particular, if every applicant is assigned to an institution according to this description, then the result is the applicant-optimal stable matching (i.e., the matching output by applicant-proposing DA).*

Proof. We denote applicant-proposing (resp., institution-proposing) DA when run with preferences P by $APDA(P)$ (resp., $IPDA(P)$). Fix an applicant d_*, fix preferences P for applicants, fix priorities for institutions, and let h be an institution. We denote by $P|_{d_*:\emptyset}$ the preference profile obtained by altering P so that d_* reports an empty preference list (i.e., marking all institutions as unacceptable), and by $P|_{d_*:\{h\}}$ the preference profile obtained by altering P so that d_* reports a preference list consisting only of h (i.e., marking all other institutions as unacceptable). We then observe the following chain of equivalences:

h is in the menu of d_* in $APDA$ with respect to the reports P_{-d_*} of other applicants

$$\iff \big(\text{By strategyproofness of } APDA\text{, see Theorem E.9}\big)$$

d_* is matched to h by $APDA(P|_{d_*:\{h\}})$

$$\iff \big(\text{By the Lone Wolf / Rural Hospitals Theorem, see Theorem E.6}\big)$$

d_* is matched to h by $IPDA(P|_{d_*:\{h\}})$

$$\iff \big(IPDA(P|_{d_*:\{h\}}) \text{ and } IPDA(P_{d_*:\emptyset}) \text{ coincide until } h \text{ proposes to } d_*\big)$$

h proposes to d_* in $IPDA(P_{d_*:\emptyset})$

$$\iff \big(IPDA(P_{d_*:\emptyset}) \text{ and } IPDA(P_{-d_*}) \text{ produce the same matching, ignoring } d_*; \text{ in } IPDA,\ h \text{ proposes in favorite-to-least-favorite order}\big)$$

h prefers d_* to its match in $IPDA(P_{-d_*})$ (in the market without d_*). □

Theorem 3.1 provides an appealing characterization of the menu of DA. It can also provide an alternative approach to defining DA, or towards proving the strategyproofness of (traditionally described) DA. To facilitate the latter, in Appendix C we prove Theorem 3.1 from first principles, without relying on the strategyproofness of DA as in the above proof.[14] One can also consider menu descriptions of DA in many-to-one markets and markets with contracts. In Remark C.2, we observe that the same arguments as in the proof of Theorem 3.1 above show that a natural generalization of Description 1 (provided in Description A.1) characterizes the menu of DA in many-to-one markets with substitutable priorities, and even in many-to-one markets with contracts where institutions have substitutable priorities that satisfy the law of aggregate demand (Hatfield and Milgrom, 2005).

We also remark that—even disregarding the goal of describing DA—Theorem 3.1 can serve as a useful lemma for reasoning about the properties of DA. For example, if one applicant's priorities increase at some set of institutions, then (all other things being equal) the match of that applicant in DA can only improve (Balinski and Sönmez, 1999); this property is immediate from Description 1. As another example, a short argument using Description 1, which we provide in Remark C.3, suffices to show that in a market with $n{+}1$ applicants, n institutions, and uniformly random full length preference lists, applicants receive in DA roughly their $n/\log(n)$th choice in expectation—rather lower than in the case with n applicants, where they receive their $\log(n)$th choice—re-proving results from Ashlagi et al. (2017); Cai and Thomas (2022).

3.1 Practical Considerations Regarding Description 1

While our main focus in the theoretical sections of this paper is searching for simple menu descriptions, we make some remarks on potential practical applications of Description 1 (and its variant for many-to-one markets, Description A.1). From the point of view of an applicant, Description 1 is qualitatively different from traditional descriptions. While the traditional description of DA cleanly explains that the matching will be feasible—i.e., that every applicant will be matched to a different institution (or, in a many-to-one market, that no institution will exceed its capacity)—its strategyproofness requires a complex mathematical proof (such as that presented in Section E.2).[15]

[14] Description 1 could also be used to give an inductive definition of DA that does not reference the traditional DA algorithm, where the induction base is that if there are no applicants then the matching is empty. Theorem 3.1 proves that this inductive definition defines a feasible matching rule (and in particular, that it defines the DA social choice rule).

[15] While clearinghouses typically encourage straightforward preference reporting, they rarely try to elucidate precisely *why* DA is strategyproof. One common approach is to instead rely on

In Description 1, the situation reverses: strategyproofness is easy to observe, but seeing why this procedure will produce a feasible matching (let alone a stable matching) becomes complex and delicate. (For example, if institution-proposing DA is replaced with student-proposing DA in Description 1, the resulting mechanism would still be strategyproof, yet might match many applicants to the same institution.) If a clearinghouse adopts Description 1, they could make strategyproofness easier to see but feasibility harder to see (a tradeoff we investigate formally in Sections 6 and 7). In real-world settings, applicants' primary concern may be their own match and the question which preference list they should report, while feasibility and stability may be the concerns of only the policymakers. Thus, adopting Description 1 may have tangible benefits.[16]

A clearinghouse could adopt Description 1 by changing their internal algorithms calculating each applicant's match to work according to Description 1. But, even if the algorithm used to calculate the matching remains unchanged, Description 1 provides a complete and accurate description of one's match in the DA mechanism (regardless of how that match is calculated). In addition to demonstrating strategyproofness, Description 1 may afford participants a better understanding of what strategyproofness means. Absent any training in economics or game theory, advice such as "it is always best to report your true ranking" may be unclear (e.g., what does best mean?).[17] Instead, Description 1 exposes strategyproofness as a simple, concrete, and tangible property: each applicant is matched to their highest-ranked attainable institution (where the set of attainable institutions cannot be influenced by one's own ranking).[18]

appeals to authority. As reported by Dreyfuss et al. (2022b), an informative video published by the National Resident Matching Program (NRMP) was formerly introduced with the text:

> Research on the algorithm was the basis for awarding the 2012 Nobel Prize in Economic Sciences. To make the matching algorithm work best for you, create your rank order list in order of your true preferences, not how you think you will match.

[16] That said, we believe Description 1 is unlikely to mitigate—and may even increase—the amount of trust applicants must place in any clearinghouse's descriptions (in contrast to, e.g., the framework of credible mechanisms, Akbarpour and Li, 2020).

[17] This advice is sometimes phrased so as to instruct applicants that they "should not strategize," an often borderline moral command. Abdulkadiroğlu et al. (2011) report a case of one parent who wanted to submit the best possible list and, frustrated by the language the school district used to describe misreporting one's true preference over the schools, said "you call this strategizing as if strategizing is a dirty word...".

[18] We speculate that there may be additional psychological framing benefits to our description. First, it may be much harder for an applicant to confuse Description 1 with the non-strategyproof Boston mechanism, where seats can indeed fill up for one, depending on her own submitted list. Second, instead of framing DA in terms of applicants getting repeatedly rejected, Description 1 frames DA in terms of choosing the best institution out of a set, which may both sound generally more positive from the point of view of real-world applicants and have specific benefits from the

4 SD, TTC, and Individualized Dictatorships

In this section, we consider the mechanisms Serial Dictatorship (SD) and Top Trading Cycles (TTC) through the lens of menu descriptions.

First, consider SD. Suppose applicants are indexed by $i \in \{1, 2, \ldots, n\}$ with a respective priority order $1, 2, \ldots, n$. Observe that from the point of view of applicant i, the traditional description of SD can be divided into three steps as follows:

(1) Each applicant $1, \ldots, i-1$, in order, is matched to her top-ranked remaining institution.

(2) Applicant i is matched to her top-ranked remaining institution.

(3) Each applicant $i+1, \ldots, n$, in order, is matched to her top-ranked remaining institution.

Since Steps (1) and (2) above are a menu description for applicant i, the traditional description of SD already contains a menu description (for each applicant simultaneously) embedded within it.

Are menu descriptions embedded in the traditional descriptions of other mechanisms? Inspired by SD, we consider a generalization of the above 3-step description. Our generalized outline applies to any environment, any strategyproof mechanism, and any player i. This outline describes the full outcome (e.g., the entire matching or allocation as opposed to only player i's match or allocation), and emphasizes strategyproofness for player i by starting with a menu description. This outline is:

(1) Using only $t_{-i} \in \mathcal{T}_{-i}$, the menu $\mathcal{M}_{t_{-i}}$ of player i with respect to t_{-i} is calculated.

(2) Using $t_i \in \mathcal{T}_i$, player i is guaranteed her favorite i-outcome from $\mathcal{M}_{t_{-i}}$.

(3) Using both t_i and t_{-i}, the full outcome $f(t_i, t_{-i})$ is calculated.

We call such a description an *individualized dictatorship description* for player i. Note that being an individualized dictatorship simply means being a description that contains a menu description for player i while calculating the full outcome.

For SD, we can now state the following observation:

Fact 4.1. *For each applicant i, the traditional description of SD is an individualized dictatorship description for applicant i.*

point of view of models of news utility, disappointment aversion, and/or ego utility.

We now turn to TTC. Perhaps surprisingly, we find that this mechanism has an individualized dictatorship description that is quite similar to its traditional description. Consider any applicant d in TTC. In contrast with SD, this description only contains a menu description for d, the applicant reading the description.[19] Our individualized dictatorship for TTC is presented in Description 2. Briefly, this new description modifies the traditional description (from Definition 2.3) only by delaying matching applicant d as long as possible.[20] This accurately describes the full matching due to the well-known fact that TTC is independent of the order in which cycles of applicants and institutions are chosen to be matched.

Description 2 An individualized dictatorship description of TTC for applicant d

(1) Using only t_{-d}, iteratively match as many cycles not involving applicant d as possible. Let M denote the set of remaining institutions.

(2) Using t_d, match d to her highest-ranked institution in M. Call this institution h.

(3) Match the cycle created when d points to h, then continue to interatively match cycles until all applicants are matched (or have exhausted their preference lists).

Theorem 4.2. *Description 2 is an individualized dictatorship description of TTC. In particular, the set M in Step (1) is applicant d's menu, and the matching produced at the end of Step (3) is the outcome of TTC.*

Proof. We use the fact that by Lemma B.4, TTC is independent of the order in which we choose to match trading cycles, and proceed by showing that Description 2 is a valid run of the traditional description of TTC (with a specially chosen order for matching cycles). Description 2 begins in Step (1) by running TTC, and as long as this is possible, only matching cycles that do not include applicant d. (By definition, Step (1) uses only t_{-d}, as needed.)

Observe that (by Lemma B.4) any institution matched during this Step (1) of Description 2 is not on d's menu. To see that each remaining institution is on d's menu, note that when d now points to her favorite remaining institution (whatever it is), since the next executed cycle *must* involve d, she gets this institution. (See

[19] At a technical level, it is easy to see that a menu description for all players at the same time is strongly OSP (Pycia and Troyan, 2023). Since virtually all mechanisms beyond SD are not strongly OSP implementable (Pycia and Troyan, 2023), we must specialize the menu description to each player separately to describe mechanisms such as TTC.

[20] This can also be thought of as running TTC, with a twist: during the first stage, applicant d does not point to any institution.

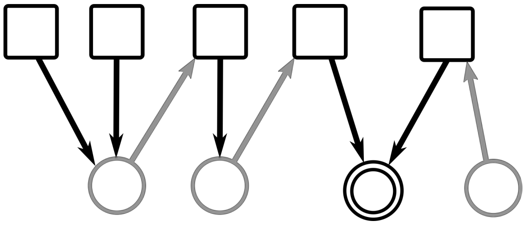


Figure 1: An illustration of the menu calculation in Description 2.

Notes: Circles are applicants; squares are institutions; each participant except applicant i (the double circle) points to her favorite remaining institution. When all possible trading cycles not involving applicant i have been performed, all remaining institutions directly or indirectly point at applicant i, who can therefore complete a cycle by pointing to any remaining institution. Thus, every remaining institution is on i's menu.

Figure 1 for an illustration.) Thus, M is exactly applicant d's menu. Finally, Step (3) correctly matches the cycle containing applicant d, and then continues running TTC as usual (matching cycles in any order) using the rest of the applicants' preferences to construct the final matching. This entire process constitutes a valid run of TTC, and thus correctly computes the final matching. □

5 Formal Model of Mechanism Descriptions

We now introduce a general framework for reasoning about descriptions of mechanisms. This framework helps capture ways in which traditional descriptions of matching mechanisms, as well as Description 1 and Description 2, may be simple. Most significantly, we use this framework to prove impossibility results that rule out the existence of additional types of simple descriptions beyond those we present.

Mechanisms can be described to players in a variety of ways. The present paper alone contains a range of options, including textual descriptions and algorithmic pseudocode. To abstract over all these possibilities, we introduce the definition of an *extensive-form description*. At a technical level, an extensive-form description is similar to an extensive-form mechanism in which different branches may "merge," i.e. an extensive-form mechanism in which the underlying game tree is actually a directed acyclic graph (DAG).[21] However, the semantic interpretation is very different from that of an extensive-form mechanism: Rather than modeling an interactive process where the players may act multiple times, an extensive-form description spells out the steps used to calculate the outcome (or the i-outcome for some player i) as a function of the (directly reported) types of the players.

[21] Alternatively, extensive-form descriptions can be viewed as finite automata where state transitions are given by querying the types of players, or (borrowing terminology from the computer science literature) as branching programs.

We are interested in three special cases of extensive-form descriptions: an *extensive-form outcome description* models traditional descriptions such as Definitions 2.2, 2.3, and 2.4. An *extensive-form menu description* models menu descriptions such as Description 1 and those from Table 1. An *extensive-form serial individualized description* models individualized dictatorship descriptions such as Description 2.

Definition 5.1 (Extensive-Form Descriptions)**.**

- An *extensive-form description* in some social choice environment is defined by a directed graph on some set of vertices V.[22] There is a (single) root vertex $s \in V$, and the vertices of V are organized into *layers* $j = 1, \ldots, L$ such that each edge goes between layer j and $j+1$ for some j. For a vertex v, let $S(v)$ denote the edges outgoing from v. Each vertex v with out-degree at least 2 is associated with some player i, whom the vertex is said to *query*, and some *transition function* $\ell_v : \mathcal{T}_i \to S(v)$ from types of player i to edges outgoing from v. (It will be convenient to also allow vertices with out-degree 1, which are not associated with any player.) For each type profile $(t_1, \ldots, t_n)$, the *evaluation path* on $(t_1, \ldots, t_n)$ is defined as follows: Start in the root vertex s, and whenever reaching any non-terminal vertex v that queries a player i and has transition function ℓ_v, follow the edge $\ell_v(t_i)$.

- An *extensive-form outcome description* of a social choice function f is an extensive-form description in which each terminal vertex is labeled by an outcome, such that for each type profile $t \in \mathcal{T}$, the terminal vertex reached by following the evaluation path on $t \in T$ is labeled by the outcome $f(t_1, \ldots, t_n)$.

- An *extensive-form menu description* of a social choice function f for player i is an extensive-form description with $k+1$ layers, such that (a) each vertex preceding layer k queries some player other than i, (b) each vertex v in layer k queries player i and is labeled by some set $\mathcal{M}(v) \subseteq A_i$, such that if v is on the evaluation path on a type profile $t \in \mathcal{T}$, then $\mathcal{M}(v) = \mathcal{M}_{t_{-i}}$ is the menu of player i with respect to t_{-i} in f, and (c) each (terminal) vertex v in the final layer $k+1$ is labeled by an i-outcome, such that if v is reached by following the evaluation path on a type profile $t \in \mathcal{T}$, then v is labeled by the i-outcome $[f(t_i, t_{-i})]_i$.

- An *extensive-form individualized dictatorship description* of f for player i is

[22] Formally, a directed graph G on vertices V is some set of ordered pairs $G \subseteq V \times V$. An element $(v, w) \in G$ is called an *edge* from v to w. A *source* (resp., *sink*) vertex is any v where there exists no vertex w with an edge from w to v (resp., from v to w).

an extensive-form outcome description of f with some layer $k \in \{1, \dots, L\}$ such that: (a) each vertex preceding layer k queries some player other than i, and (b) each vertex v in layer k queries player i and is labeled by some set $\mathcal{M}(v) \subseteq A_i$, such that if v is on the evaluation path on a type profile $t \in \mathcal{T}$, then $\mathcal{M}(v) = \mathcal{M}_{t_{-i}}$ is the menu of player i with respect to t_{-i} in f.

Any precise algorithmic description—whether it is an outcome, menu, or individualized dictatorship description—induces an extensive-form description in a natural way: the vertices in layer j are the possible states of the algorithm after querying the types of different players altogether j times. For example, consider the menu description of a second price auction given in Table 1(b). This description calculates the menu of one bidder as a function of (only) the highest bid placed by any other bidder. This can be made precise via an extensive-form description that queries the other bidders one-by-one, while keeping track of only the highest bid placed by any of them. Figure 2 provides an illustration. We sometimes refer to an extensive-form description simply as a "description" for brevity. For the mechanisms SD, TTC, and DA, we use the term "traditional description" to refer to an extensive-form outcome description that formalizes the canonical (algorithmic) description of the mechanism, as in Definitions 2.2, 2.3, and 2.4.

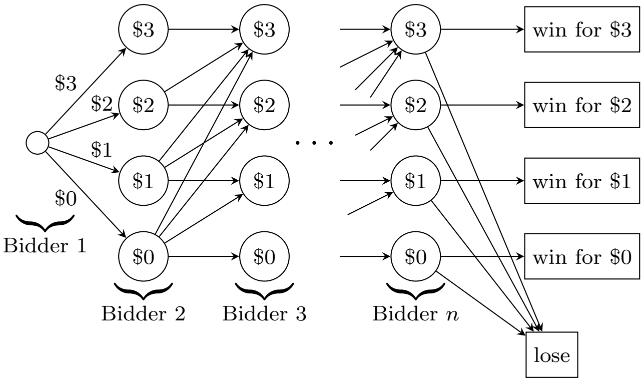


Figure 2: An extensive-form menu description for bidder n in a second-price auction.

Note: The second-to-last layer is labeled with bidder n's menu, abbreviated in the figure by the price she must pay to win the item.

6 Main Impossibility Result for DA

6.1 Applicant-Linear Descriptions

We now apply our general framework from Section 5 to matching mechanisms. We start by identifying two crucial properties that the traditional descriptions of SD, TTC, and DA (and virtually all other popular matching mechanisms) share. First, they only consider the preferences of applicants once, in a specific, natural order—from favorite to least favorite. Second, they require a small amount of bookkeeping as they run—little more than the bookkeeping required to remember a single matching. We formalize the latter requirement using a memory requirement: a general quantitative measure of the amount of bookkeeping a description uses.

Definition 6.1 (Applicant-Linear Descriptions)**.**

- In a matching environment, an extensive-form description D is *applicant-proposing* if it satisfies the following: For every applicant i and every possible evaluation path through D, let $v_1, v_2, \ldots, v_k$ denote the vertices along the evaluation path that query applicant i. Then, for $j = 1, \ldots, k$, the transition function $\ell_{v_j} : \mathcal{T}_i \to S(v_j)$ (which determines which edge to follow in the evaluation path) depends only on the jth institution on applicant i's preference list (possibly an "empty institution" if this list consists of fewer than j institutions).[23]

- The *memory requirement* of an extensive-form description is the logarithm, base 2, of the maximum number of vertices in any layer of the graph. (This is precisely the number of bits required to store the vertex number of the current vertex within a layer; intuitively, this is the amount of bookkeeping or "scratch paper" required by the description.)

- In a matching environment with n applicants and n institutions, an extensive-form description is *applicant-linear* if it is applicant-proposing and uses at most $\widetilde{\mathcal{O}}(n)$ memory.[24,25]

Each of SD, TTC, and DA is traditionally defined using applicant-linear descriptions.

Observation 6.2. *Each of SD, TTC, and DA has an applicant-linear outcome description.*

[23] While we call this property "applicant-*proposing*," it also applies to the "applicant-pointing" TTC description, as well as to any other description that uses applicant preferences (one time only) in favorite-to-least-favorite order.

[24] The standard computer-science notation $\widetilde{\mathcal{O}}(n)$ means $O(n \log^{\alpha} n)$ for some constant α. That is, for large enough n, memory is upper-bounded by $cn \log^{\alpha} n$ for some constants c, α that do not depend on n. Using $\widetilde{\mathcal{O}}(n)$ memory means using only nearly constant bookkeeping per applicant.

[25] We remark that the name "linear" refers to two things in the interest of brevity: the linear order in which the description reads preferences (i.e., being applicant-proposing), and the nearly-linear amount of bookkeeping used (i.e., $\widetilde{\mathcal{O}}(n)$).

We start by making two technical remarks. First, note also that $\widetilde{\mathcal{O}}(n)$ is exactly (up to the precise logarithmic factors) the number of bits of memory required to describe a single matching (or a single applicant's menu). To see this formally, note that there are $n! = 2^{O(n \log n)}$ distinct matchings involving n applicants and n institutions (and exactly 2^n possible menus). Intuitively, this simply formalizes the fact that the number of letters it takes to write down a single matching with n applicants and n institutions (or, a subset of the n institutions) is roughly proportional to n. Thus, $\widetilde{\mathcal{O}}(n)$ is the minimal possible memory requirement of any description that calculates a matching (or one applicant's menu). Another reason that $\widetilde{\mathcal{O}}(n)$-memory descriptions are particularly natural for matching mechanisms is that on average only a small amount of information about each applicant's type (namely, $O\big(\log^k(n)\big)$-many bits) need be remembered at any point throughout the evaluation.[26]

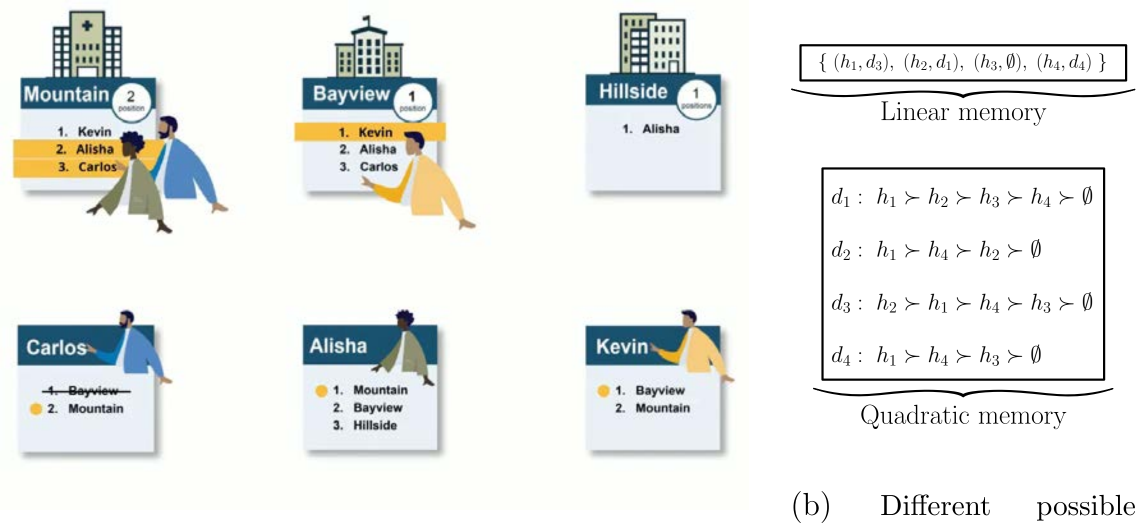
Second, note that assuming only that a description is applicant-proposing (with no bound on the memory) places no restrictions on what the description can compute. To see this, pick any matching mechanism and consider an extensive-form description that (a) queries applicants one-by-one for their entire preference list while remembering that list in its entirety (formally, this is done by constructing the directed graph of the extensive-form description to be a tree), and then (b) outputs the outcome of the matching rule (formally, each leaf of the tree is labeled with the outcome of the matching mechanism for the types queried on the path to that leaf). This description shows that the maximum memory requirement for describing any matching mechanism is $\widetilde{O}(n^2)$, matching the memory required to store all $(n!)^n = 2^{O(n^2 \log(n))}$ possible preference profiles for all applicants. When we prove our main impossibility result (Theorem 6.4) below, we show that a certain class of descriptions requires memory $\Omega(n^2)$, matching (up to the precise logarithmic factors) this as-high-as-possible solution.[27]

To demonstrate why we interpret applicant-linearity also as a simplicity notion for descriptions of matching mechanisms, consider a description of DA given in one of its most celebrated applications: matching medical doctors to residencies in the US National Residency Matching Program (NRMP). This description is in a form of a video that describes DA by applying it to an example small matching market; see Figure 3(a). The explanation in the video is aided by two visual elements: crossing off institutions from applicants' lists as the description progresses, and keeping track of a

[26] Moreover, applicant-proposing descriptions have the natural property that they can read each part of each applicants' preference list only once, so information that is read but not recorded (in the small amount of memory available) cannot matter for the rest of the evaluation.

[27] The standard computer-science notation $\Omega(n^2)$ means that, for large enough n, memory is lower-bounded by cn^2 for some constant c that does not depend on n.

"current tentative matching" illustrated by the yellow-highlighted names. We observe that these two simple visual elements are enabled precisely by the two desiderata of applicant-linearity.


Mountain
2
position
1. Kevin
2. Alisha
3. Carlos
Bayview
1
position
1. Kevin
2. Alisha
3. Carlos
Hillside
1
positions
1. Alisha
Carlos
~~1. Bayview~~
2. Mountain
Alisha
1. Mountain
2. Bayview
3. Hillside
Kevin
1. Bayview
2. Mountain
$\{ (h_1, d_3), (h_2, d_1), (h_3, \emptyset), (h_4, d_4) \}$
Linear memory
$d_1 : h_1 \succ h_2 \succ h_3 \succ h_4 \succ \emptyset$
$d_2 : h_1 \succ h_4 \succ h_2 \succ \emptyset$
$d_3 : h_2 \succ h_1 \succ h_4 \succ h_3 \succ \emptyset$
$d_4 : h_1 \succ h_4 \succ h_3 \succ \emptyset$
Quadratic memory

(a) An illustration of the traditional description of DA.

(b) Different possible memory requirements.

Figure 3: (a): Screenshot of a video illustrating the traditional description of DA through an example. (b): A graphical illustration of the contrast between roughly linear (i.e., $\widetilde{O}(n)$) memory and quadratic (i.e., $\Omega(n^2)$) memory.

Notes: Screenshot taken from NMS (2020), a video produced by National Matching Services, the company that provides matching software to the National Residency Matching Program.

First, the fact that the description is applicant-proposing is necessary for the video to cross off institutions from applicants' lists as the description progresses. This would not have been possible for a description that is not applicant-proposing, i.e., a description that reads applicant preferences in an order that is not favorite-to-least-favorite, or reads these preferences multiple times.

Second, the linear memory requirement of the description is necessary for the yellow highlighting in the video, which illustrates one tentative match for each applicant, to capture the entire required bookkeeping. This would not have been possible for a description that requires much more than linear memory. For example, any description that requires *quadratic* memory (i.e., $\Omega(n^2)$, matching the bound achieved in our main impossibility result, Theorem 6.4 below) relies on an amount of bookkeeping roughly equivalent to all of the n preference lists of each applicant simultaneously—the same bookkeeping requirement as remembering n disjoint matchings, which would require some much bulkier, more verbose illustration. See Figure 3(b) for an illustration of the stark difference between linear and quadratic memory.

While some non-applicant-proposing and/or non-linear-memory algorithms may or may not be simple in other senses, we contend that such algorithms are not simple in the same way that traditional descriptions of DA are simple. In particular, such non-applicant-linear algorithms could not leverage the properties facilitating the simple description in the video.

We finally note that while applicant-linearity captures a way in which the traditional descriptions of SD, TTC, and DA are simple, we do not view *every* applicant-linear description as simple. Rather, applicant-linearity aims to be a necessary, but not sufficient, condition for being simple in a particular sense shared by traditional descriptions. Indeed, we examine a different type of simplicity in Section 7, and additional notions of simplicity may be investigated by future work.

In the next section, we use applicant-linearity to investigate the relationship between menu descriptions and and traditional descriptions. Our main result provides a strong sense in which uncovering a menu description within a small tweak of the traditional description of DA is impossible, using applicant-linearity to capture (a necessary condition for being) a small tweak of the traditional description of DA.

6.2 Individualized Dictatorships for DA: A Stark Contrast to SD and TTC

Recall that for both SD and TTC, we constructed menu descriptions within (small tweaks of) the traditional descriptions in Section 4. In particular, we constructed individualized dictatorships, which expose strategyproofness to one applicant while calculating the entire matching. Like the corresponding traditional descriptions, the individualized dictatorships in Section 4 are applicant-linear:

Corollary 6.3. *For any applicant, SD and TTC each have an applicant-linear individualized-dictatorship description.*

We now turn to DA, and present our main impossibility result. We prove that nothing like Corollary 6.3 is possible for DA. That is, under our simplicity condition from Section 6.1, no simple individualized dictatorship for DA exists. Formally:

Theorem 6.4. *For any applicant d, there exist priorities of the institutions such that no applicant-linear individualized dictatorship description of DA exists. In fact, any applicant-proposing individualized dictatorship extensive-form description for DA requires $\Omega(n^2)$ memory.*

In addition to the literal interpretation of Theorem 6.4—ruling out a class of simple individualized dictatorships for DA—we view Theorem 6.4 as showing that a menu description cannot be found within a small tweak of the traditional description of DA. First, we contend that any small tweak of the traditional (applicant-linear) description should still be applicant-linear, since (as discussed in Section 6.1) any description that is not applicant-proposing must query preferences in a fundamentally different way, and any description with dramatically higher memory requirements must have dramatically different bookkeeping. Second, a small tweak should still calculate the same overall matching. Combined with the requirement that the menu be calculated (within the tweaked description) without querying applicant d's type (as for all menu descriptions, to expose strategyproofness), this means that the tweaked description must be an individualized dictatorship (Definition 5.1). Thus, Theorem 6.4 rules out the possibility that a small tweak of the traditional description of DA can describe one applicant's menu in a way that exposes strategyproofness.

Before proving Theorem 6.4 we make one technical remark. Recall that in Section 4, we used the fact that the traditional description of TTC is independent of the execution order that the mechanism chooses (i.e., the order in which cycles/proposals are chosen) to find an appealing individualized dictatorship. Since applicant-proposing deferred acceptance is independent of execution order as well, one may wonder why a similar approach does not work for DA (as such a result is precluded by Theorem 6.4). To see why this is the case, recall that as shown by an example in Section 3, in DA allowing all other applicants to propose before applicant d proposes does *not* suffice to calculate the menu (because applicant d proposing at that point to an institution might result in a rejection cycle that leads to the rejection of d from that institution).

The proof of Theorem 6.4 constructs (for a carefully chosen, fixed set of institution priorities) a very large set of applicant preferences such that: (a) to learn the menu of applicant d, a large fraction of *every other applicant's* preference list must be read, and (b) to correctly compute the final matching, all of the information from these applicants' lists must be remembered in full. The $\Omega(n^2)$ lower bound comes from this large amount of information that must be remembered.

Proof. Fix an applicant d_* and let D be any applicant-proposing individualized dictatorship extensive-form description of DA for d_*.

We now describe a set $\mathcal{S}$ of possible inputs to DA, illustrated in Figure 4. For simplicity, let n be a multiple of 4. There are $n/2$ total "2-cycles" containing two applicants and two institutions each. Cycle i has applicants d_i and d_i' and institutions

h_i and h'_i. The cycles are divided into two classes, "top" cycles (for $i = 1, \ldots, n/4$) and "bottom" cycles (for $i = n/4+1, \ldots, n/2$).

The institutions' priorities are fixed, and defined as follows:

For top 2-cycles $(i \in \{1, \ldots, n/4\})$:	For bottom 2-cycles $(i \in \{n/4+1, \ldots, n/2\})$:
$h_i: \quad d'_i \succ d_* \succ d_i$	$h_i: \quad d'_i \succ d_1 \succ d_2 \succ \ldots \succ d_{n/4} \succ d_i$
$h'_i: \quad d_i \succ d'_i$	$h'_i: \quad d_i \succ d'_i$

For the top cycle applicants d_i with $i \in \{1, \ldots, n/4\}$, the preferences vary (in a way we will specify momentarily). Other applicant preference are fixed, as follows:

For bottom 2-cycles $(i \in \{n/4+1, \ldots, n/2\})$:	$d_i: \quad h_i \succ h'_i$
For all 2-cycles $(i \in \{1, \ldots, n/2\})$:	$d'_i: \quad h'_i \succ h_i$

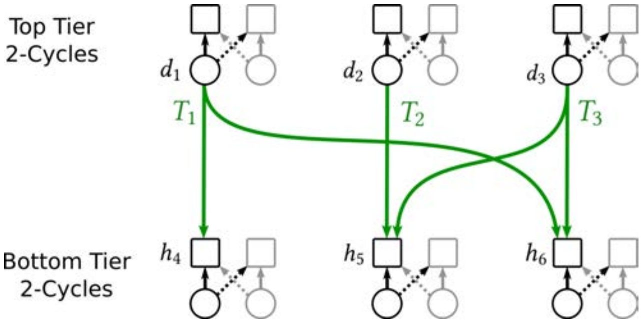


Figure 4: Illustration of the set of preferences used in Theorem 6.4.

Notes: Dark nodes represent d_i or h_i for some i, and grey nodes represent d'_i or h'_i. The arrows directed outwards from a top-tier d_i represent the sets T_i. We show that these sets T_i must be fully remembered by any applicant-proposing individualized dictatorship for DA.

Let $\mathcal{S}$ denote the set of preference profiles where, in addition to all the above, for all top cycle d_i with $i \in \{1, \ldots, n/4\}$, we have

$$d_i: \qquad h_i \succ T_i \succ h'_i$$

where T_i is an arbitrary subset of $\{h_j \mid j \in \{n/4+1, \ldots, n/2\}\}$ (the set of h_i from bottom cycles) placed in an arbitrary fixed order. Any such collection of $\{T_i\}_{i=1,\ldots,n/4}$ uniquely defines a preference profile in $\mathcal{S}$. Note that $|\mathcal{S}| = 2^{(n/4)^2}$.

We additionally define a set of inputs $\mathcal{S}' \supseteq \mathcal{S}$. Specifically, let $\mathcal{S}'$ denote the set containing every element of $\mathcal{S}$, and additionally any top cycle applicant d_i may or may not truncate the final institution h'_i off her list. In other words, in addition to the sets $\{T_i\}_{i=1,\ldots,n/4}$, an element of $\mathcal{S}'$ is defined by bits $\{b_i\}_{i=1,\ldots,n/4}$, such that whenever $b_i = 0$, we set d_i's preference to $h_i \succ T_i \succ h_i'$, and whenever $b_i = 1$, we set d_i's preference is $h_i \succ T_i$.[28]

We now proceed with two lemmas formally establishing that each set T_i must be read during Step (1) of D, and each set T_i must be remembered during Step (3) of D.

Lemma 6.5. *In order to correctly calculate the menu of d_* on $\mathcal{S}'$, description D must query the entire preference list of each top cycle applicant d_i (up to the position of h_i').*

To prove this lemma, consider the rejection chain that occurs when d_* submits a list containing only h_i. First, d_i is rejected, then she proposes to every institution $h_j \in T_i$. This "rotates" the bottom cycle containing h_j. That is, h_j will accept the proposal from d_i, then d_j will propose to h_j', then d_j' with propose to h_j, so d_i will then be rejected from h_j. This will occur for every $h_j \in T_i$, so d_i will not match to any h_j with $j \in \{n/4 + 1, \dots, n/2\}$.

Finally, after getting rejected from each institution in T_i, d_i may or may not propose to h_i'. If she does not, then h_i will be on d_*'s menu. If she does, then h_i' will reject d_i', who will propose to h_i, which will reject d_*, so h_i will not be on d_*'s menu. Thus, any applicant-proposing description must read all of d_i's list before being able to correctly learn the menu of d_*. This proves Lemma 6.5.

Lemma 6.6. *For each distinct set of preference profiles t_{-*} in $\mathcal{S}$, the induced function $APDA(\cdot, t_{-*}) : \mathcal{T}_* \to A$ from types of d_* to matchings is distinct.*

To prove this lemma, consider two preference profiles in $\mathcal{S}$, one profile t_{-*} defined by $\{T_i\}_{i \in \{1,\dots,n/4\}}$, and the other profile t'_{-*} defined by $\{T_i'\}_{i \in \{1,\dots,n/4\}}$. As these are distinct collections of sets, without loss of generality there is some i and j such that $h_j \in T_i \setminus T_i'$. Suppose now that d_* proposes just to h_i. Then consider the rejection chain under t_{-*} and under t'_{-*}. Under t_{-*}, the bottom tier 2-cycle containing h_j will be "rotated," i.e. h_j will match to d_j' in the final matching. However, this is not the case under t'_{-*}. Thus, t_{-*} and t'_{-*} produce different final matchings under the same preference list of d_*, and thus induce distinct functions from $\mathcal{T}_*$ to outcomes. This proves Lemma 6.6.

We can now prove Theorem 6.4. Together, Lemma 6.5 and Lemma 6.6 show that at the layer where D presents the menu to d_*, the description must be in a distinct state for each possible way of assigning $T_i \subseteq \{h_i | i \in \{n/4 + 1, \dots, n/2\}\}$ for $i \in \{1, \dots, n/4\}$. There are $2^{(n/4)^2} = 2^{\Omega(n^2)}$ possible ways to set the collection $\{T_i\}_i$, so the description requires at least this number of states, and thus uses space $\Omega(n^2)$. □

[28] This collection of preferences can also be constructed with full preference list by adding some unmatched institution $h_\emptyset$ to represent truncating d_i's list.

Theorem 6.4 is the main impossibility result of our paper: it gives a robust and precise sense in which it is hard to infer from the traditional description of DA that DA "has a menu"—i.e., that an applicant gets her highest-ranked institution from a set that her report cannot influence—a property equivalent to strategyproofness. It shows that identifying the menu requires vastly more bookkeeping—in fact, nearly the maximum possible amount of bookkeeping (as discussed in Section 6.1)—than performed by the traditional description.

Theorem 6.4 rules out a broad class of individualized dictatorship descriptions resembling traditional ones, namely, all those that are applicant-proposing and use low memory. The theorem is tight in the sense that none of these three requirements (individualized dictatorship, applicant-proposing, and low memory) can be dropped. First, the traditional description is applicant-proposing and uses low memory, but does not compute the menu (so is thus not an individualized dictatorship). Second, as discussed in Section 6.1, an applicant-proposing individualized dictatorship that uses $\widetilde{O}(n^2)$ memory can be constructed for any matching mechanism. Finally, we can show that an $\widetilde{O}(n)$ memory individualized dictatorship description of DA exists that makes *just two* passes through the preference list of each applicant (one pass before the menu is computed, and one pass afterward).[29]

All told, there is a stark three-leveled contrast in our framework between SD, TTC, and DA. The strategyproofness of SD is already clear, simultaneously for all applicants, from its traditional description. To expose the strategyproofness of TTC, the traditional description of the matching must be slightly tweaked and specialized to each individual applicant. However, once this is done, strategyproofness is easy to see.[30] For DA, in contrast with both other mechanisms, no small tweak of the traditional description suffices to expose strategyproofness through menus.

[29] This follows from a result in Section D.2, which we discuss in Section 7 below, that shows that an applicant-linear menu description of DA exists. A two-pass individualized dictatorship can then use this description to compute the menu of one applicant, and then "restart" and use the traditional description of DA to compute the rest of the matching. We informally observe that the use of two passes significantly obscures the connection and consistency between the menu and the resulting matching.

[30] Obvious strategyproofness (OSP) can be interpreted as one formal sense in which strategyproofness can be exposed to all players simultaneously. Indeed, it is not hard to see that any description that is a menu description for all players simultaneously, such as the traditional description of SD, is OSP-implementable. However, TTC is not OSP-implementable (Li, 2017). This gives a formal sense in which specializing the description of TTC to different players is necessary.

7 The Landscape of Descriptions of DA

7.1 Additional Descriptions of DA

Description 1, our positive result from Section 3, is an institution-proposing description of DA (specifically, a menu description).[31] In contrast, Theorem 6.4, our impossibility result from Section 6, is for *applicant*-proposing descriptions of DA (specifically, for individualized dictatorships). This difference leaves open the question of whether other types of descriptions are feasible. For example, could there be an appealing institution-proposing individualized dictatorship? To begin this search, we utilize the simplicity condition in Theorem 6.4, and ask whether there exist additional descriptions with low (i.e., $\widetilde{O}(n)$) memory.

Perhaps surprisingly, in Appendix D we construct low-memory extensive-form descriptions of (applicant-optimal) DA of every type not ruled out by Theorem 6.4. That is, we construct for DA each of:

- An institution-proposing, $\widetilde{O}(n)$-memory outcome description (Section D.1, adapted from an algorithm used by Ashlagi et al. (2017)).
- An applicant-proposing, $\widetilde{O}(n)$-memory menu description (Section D.2).
- An institution-proposing, $\widetilde{O}(n)$-memory individualized dictatorship (Section D.3).

Unfortunately, except for the traditional description of DA and for Description 1, every description we construct is a delicate and technical algorithm; as one can see in Appendix D, each of these algorithms uses careful, and likely unintuitive, bookkeeping to maintain low-memory, and requires many sub-routines to define. Thus, these algorithms seem impractical. However, this does not necessarily imply that there are no other, more attractive, descriptions of these types. We address this (im)possibility in the next section.

7.2 Local One-Side-Proposing Descriptions of DA

In this section, we give a precise sense in which any descriptions of the types discussed in Section 7.1 *must* be convoluted. Briefly, any such description must have a property we call *non-locality*: it must update bookkeeping concerning some participants when

[31] While we have not formally defined institution-proposing descriptions, whenever we use these terms we mean the analogous definitions to Definition 6.1, in which sides are interchanged (and in particular, for purposes of analysis, the vertices of the extensive-form description query the institutions' priorities).

Table 5: Full classification of one-side-proposing descriptions of the applicant-optimal stable matching mechanism.

	Menu Description	Outcome Description	Individualized Dictatorship
Applicant proposing	In Section D.2. Necessarily **non-local** by Theorem 7.2.	Traditional DA algorithm	**Completely impossible** by Theorem 6.4.
Institution proposing	Description 1 in Section 3	In Section D.1 / Ashlagi et al. (2017). Necessarily **non-local** by Theorem 7.3.	In Section D.3. Necessarily **non-local** by Theorem 7.3.

Notes: We look for descriptions of DA that use at most $\widetilde{O}(n)$ memory. Descriptions either read preferences in an applicant-proposing manner or read priorities in an institution-proposing manner. Descriptions either compute the menu (exposing strategyproofness), compute the outcome matching (exposing feasibility), or compute both the menu and the matching in an individualized dictatorship (exposing both).

making queries that seem unrelated to them. See Table 5 for an overview all our descriptions and impossibility results for DA.

Technically, we call an applicant-proposing outcome description (resp., menu description for applicant d) *local* if (a) in addition to any global bookkeeping, it also maintains local bookkeeping for each institution, and this bookeeping is only updated when that institution is read from any applicant's list, and (b) the calculated match of an institution (resp., whether the institution is on applicant d's menu) only depends on the final state of the local bookkeeping for this institution. The definition of local bookkeeping for institution-proposing descriptions interchanges the roles of applicants and institutions, and locality analogously requires that the calculated part of the output relevant to an applicant d depend only on the local bookkeeping for d. Formally:

Definition 7.1.

- In a matching environment, *local bookkeeping* for an applicant-proposing extensive-form description D is a label $\big(L_1(v), \ldots, L_n(v)\big)$ for each vertex v in D such that the following holds for every internal vertex v in D. Let i be the applicant queried at v. Since D is applicant-proposing, recall that the transition function $\ell_v : \mathcal{T}_i \to S(v)$ depends only on the jth institution (possibly an empty institution $\emptyset$) on i's preference list, for some j. Abusing notation, we therefore consider ℓ_v to be a function from institutions to $S(v)$. Then, for every institution k, the labels of v and of $\ell_v(k)$ may only differ in their kth coordinate, L_k (and in particular, if $k = \emptyset$, the labels of v and $\ell_v(k)$ must be the same).

- An applicant-proposing extensive-form outcome description D is *local* if there exists local bookkeeping for it such that for every terminal vertex v in D and institution k, the determination of the match of k at v depends only on $L_k(v)$.

- An applicant-proposing extensive-form menu description D for applicant i is *local* if there exists local bookkeeping for it such that for every terminal vertex v in D and institution k, the determination of whether k is on i's menu at v depends only on $L_k(v)$.

- The definition of local bookkeeping for an institution-proposing extensive-form description is completely analogous to that of an applicant-proposing one, interchanging the roles of applicants and institutions. In particular, each vertex of such a description queries an institution's priorities, and if applicant i is read, only the label $L_i(v)$ that concerns applicant i can be updated.

- Accordingly, an institution-proposing extensive-form outcome description D is *local* if there exists local bookkeeping for it such that for every terminal vertex v in D and applicant i, the determination of the match of i at v depends only on $L_i(v)$.

- An institution-proposing extensive-form menu description D for applicant i is *local* if there exists local bookkeeping for it such that for every terminal vertex v in D, the determination of i's menu at v depends only on $L_i(v)$ (and in particular, the labels for other applicants are not used by this definition).

The traditional description of DA is a local applicant-proposing outcome description, and Description 1 is a local institution-proposing menu description. In contrast, each of the (convoluted, yet $\widetilde{O}(n)$ memory) descriptions that we present in Appendix D (corresponding to the other description types from Table 5) is non-local.

Like applicant-linearity, locality captures one possible sense in which the traditional description of DA is simple (e.g., the description depicted in Figure 3 is a local applicant-proposing outcome description). Whereas low-memory restricts the *amount* of bookkeeping used, locality restricts the *manner* in which the the bookkeeping is updated and used. While low-memory and locality are not formally comparable (for either applicant- or institution-proposing descriptions), locality seems like a more specialized simplicity condition. For example, the applicant-proposing, low-memory menu description of TTC given in Description 2 is non-local, and yet seems nearly as simple as the traditional description of TTC.[32] For descriptions of DA, however, non-locality seems to capture one aspect that we find unintuitive in all of the descriptions that we construct in Appendix D. By ruling out local descriptions of the same types, we believe our impossibility theorems below provide good evidence that there are no major simplifications to these delicate descriptions. This suggests that practical descriptions of their types are not likely to exist. Formally, our results are:

Theorem 7.2. *If there are at least three applicants and three institutions, then for every applicant i there exist priorities of the institutions such that any applicant-proposing menu description of DA for applicant i is non-local.*

Theorem 7.3. *If there are at least three applicants and two institutions, then there exist preferences of the applicants such that any institution-proposing outcome description of DA is non-local.*

The proofs are in Appendix C. A direct corollary of these results is that no local one-side-proposing individualized dictatorship exists for DA, as such a description contains either an applicant-proposing menu description, or an institution-proposing outcome description. All told, our results say that simple one-side-proposing descriptions of DA face a formal tradeoff between conveying feasibility (as in the traditional description) and conveying strategyproofness (as in Description 1).

8 Menu Descriptions of Auctions

As a secondary application of our theoretical framework, we briefly explore the possibility of simple menu descriptions for multi-item welfare-maximizing auctions, and draw parallels with our results for matching mechanisms. We study the VCG mechanism for different classes of bidder valuations; see Appendix B for exposition and preliminaries for this environment.

Thinking about VCG through the lens of menus is perhaps particularly natural: a common way to explain the strategyproofness of VCG is to note that the price that bidder i pays when winning any bundle of items S is independent of bidder i's report; rather, bidder i's report is used only to determine which bundle bidder i wins. Indeed,

[32] This also illustrates why, in contrast to applicant-linearity, locality likely does not give a flexible enough definition to capture all "small tweaks" of the traditional description of a matching mechanism.

specifying a bidder's menu is equivalent to specifying—for each bundle S—the price that she would pay if she wins S (which might be different from the price paid by a different bidder who actually wins S).

In this section, we show that for multi-item VCG in some settings, a menu description can be given via a separate menu description regarding each item separately, while for other settings this is impossible. We first formalize this.

Definition 8.1.

- Consider an auction environment with n bidders and m items. An extensive-form description of a mechanism in this environment is *item-read-once* if along each evaluation path $v_1, v_2, \ldots, v_k$ and for each item j, there exists an interval $v_u, v_{u+1}, \ldots, v_{u+p}$ of vertices along the path such that (a) for each vertex v in this interval, the transition function ℓ_v (which determines which edge to follow in the evaluation path) only depends on some bidder's valuation for item j, and (b) for each vertex w outside of this interval, the function ℓ_w *does not* depend on any bidder's valuation for j.
- An *item-linear* description is an item-read-once description that uses at most $\widetilde{O}(m)$ memory.

In close parallel with the discussion in Section 6.1 for matching mechanisms, the bidder-read-once condition only rules out certain descriptions if complemented with an additional restriction, such as on the memory used by the algorithm. Moreover, note that $\widetilde{O}(m)$ is (up to logarithmic factors) the number of bits required to describe a "tentative allocation" of the items, or a single price for each item, and therefore is as low as possible.

We now consider auctions where bidders' valuations are additive over items. That is, each bidder's valuation for a bundle of items is the sum of her valuations for the individual items in the bundle.

Theorem 8.2. *The VCG auction with additive bidders has an item-linear outcome description. For any bidder i, it also has an item-linear menu description for bidder i.*

Proof. The item-linear outcome description of the auction simply goes through the items one by one, queries each bidder for her valuation, and keeps track of the highest bidder, her bid, and the second-highest bid. The menu description is even simpler: it keeps track of only the highest bid on each item; see Figure 2 on Page 20 for an illustration with a single-item. □

Thus, the situation for additive bidders is similar to that of TTC: there is a menu description that is a fairly simple modification to the traditional description, specializing the order of steps to compute the menu of player i. (Additionally, it is not hard to construct an individualized dictatorship for player i, provided that Definition 8.1 is modified to allow player i's values to be queried after all other players.) Moreover, this gives some sense in which the mechanism's strategyproofness can be understood in terms of the (separate) strategyproofness of m separate single-item auctions. In fact, this demonstrates a general property of our framework: it allows for the "composition" of simple menu descriptions when players are additive over the sub-mechanisms.[33]

When bidders have unit-demand valuations over items, concrete descriptions become harder. This holds both for menu descriptions and for descriptions of the outcome. In this sense, descriptions of auctions for unit-demand bidders are even harder than those of DA, even though the structure of any menu is as simple as in auctions for additive bidders (just a single price for each item). We prove this result in Appendix C.

Theorem 8.3. *No item-linear description of an auction with unit-demand bidders exists. In fact, any item-read-once description for unit-demand bidders requires memory* $\Omega(m^2)$*. This holds both for outcome descriptions and for menu descriptions.*

9 Experiment

When a mechanism is presented using a menu description, strategyproofness is always easy to formally show; indeed, a proof follows almost immediately from the description. But do mechanism participants intuitively see this—and act on it? In practical terms, do real people increase straightforward play under menu descriptions? Our experiment explores this question in two elementary mechanism-design settings.

9.1 Experiment Flow

Our experiment consists of two parts in a fixed order: median voting in an election with three single-peaked voters (henceforth, Median); and bidding in a single-item

[33] This attribute is not always shared by other simplicity notions such as OSP, which captures the simplicity of a single-item (ascending price) auction (Li, 2017), but where no OSP auction exists with even two bidders who have additive valuations over two items (Bade and Gonczarowski, 2017). In fact, this attribute is not even satisfied by communication-efficient dominant-strategy mechanisms: Rubinstein et al. (2021) construct a composed mechanism where all dominant-strategy implementations require exponentially more communication than the sub-mechanisms.

second-price auction with five bidders (henceforth, Auction). Respondents are randomly assigned, in each part independently, into either a Traditional (T) or a Menu (M) treatment. The two treatments differ from each other only in the way the mechanism is presented to participants.[34]

After informed-consent and introductory screens, for each mechanism, participants walk through four "setup" screens: (1) instructions, (2) a practice round, (3) practice-round results, and (4) further examples and comprehension questions. See the online experimental materials for screenshots. Respondents who fail to correctly answer a comprehension question within three attempts are given the answer; they must enter it to proceed. (We analyze comprehension results below, and provide additional analysis in Appendix A.)

The setup screens work together to convey, and confirm understanding of, the workings of the environment (i.e., the way outcomes affect the participant's earnings) and the mechanism (i.e., the way that the participant's vote or bid, and the other randomized votes/bids, determine the outcome). Because our goal is to test how changes to the description affect behavior, the materials are careful to not give any form of strategic advice.

In each of the two parts of the experiment, after completing the setup screens, participants participate in ten rounds of voting (in Median) or bidding (in Auction). After completing both parts, respondents fill out an exit questionnaire that includes demographic questions, an informal numeracy test, and (for each mechanism) open-ended questions on their understanding, strategies, and thoughts. They are paid the sum of their earnings in all (twenty) rounds.[35] The experiment is programmed on Otree (Chen et al., 2016).

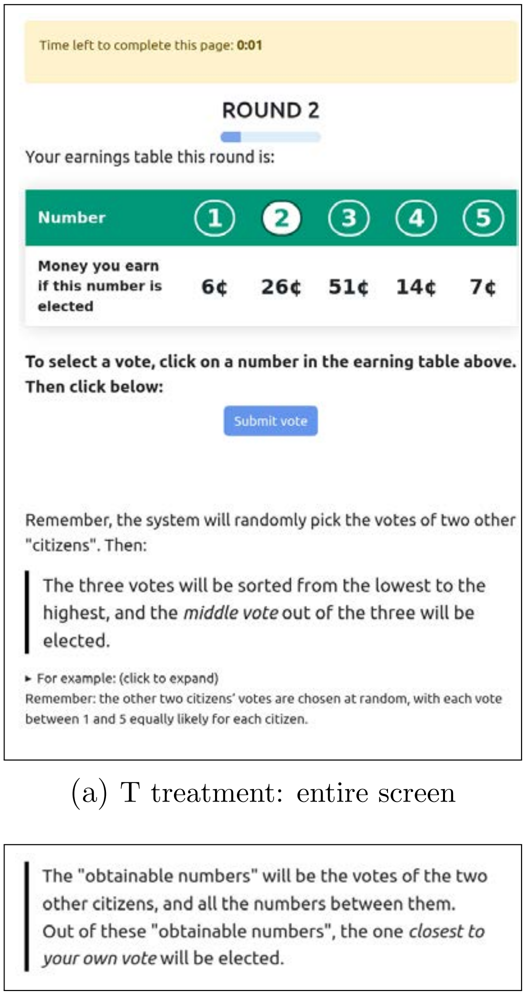
9.2 Traditional (T) and Menu (M) Treatments

Figure 5 and Figure 6 reproduce, for Median and Auction respectively, the main screen that participants face in each round. Each figure's panel (a) shows an entire screen in T, while panel (b) shows the only changed part of the screen in M. In both treatments, participants see their private information—their single-peaked value ta-

[34] Presentation differences include the language describing the mapping from votes or bids to outcomes, the language used in examples and comprehension questions that accompany the description, and, in Median, graphic illustrations that accompany the examples. (The examples themselves depict situations that are identical across the treatments.)

[35] Participants can lose money in rounds of Auction where they overbid. They are informed that if their cumulative earnings are negative at the end of all rounds, they will be set to \$0.00. (This happened to two participants, with −\$1.46 and −\$0.44.)

ble in Median and a private value in Auction—and have 30 seconds to submit their vote/bid.[36] They are reminded (1) how the mechanism works and (2) that the system randomly picks the votes of other two voters (in Median) or bids of other four bidders (in Auction). After submitting their vote or bid, participants see the result of the round, including their earnings.



Time left to complete this page: **0:01**

ROUND 2

Your earnings table this round is:

Number	1	2	3	4	5
Money you earn if this number is elected	6¢	26¢	51¢	14¢	7¢

To select a vote, click on a number in the earning table above. Then click below:

Remember, the system will randomly pick the votes of two other "citizens". Then:

> The three votes will be sorted from the lowest to the highest, and the *middle vote* out of the three will be elected.

▸ For example: (click to expand)

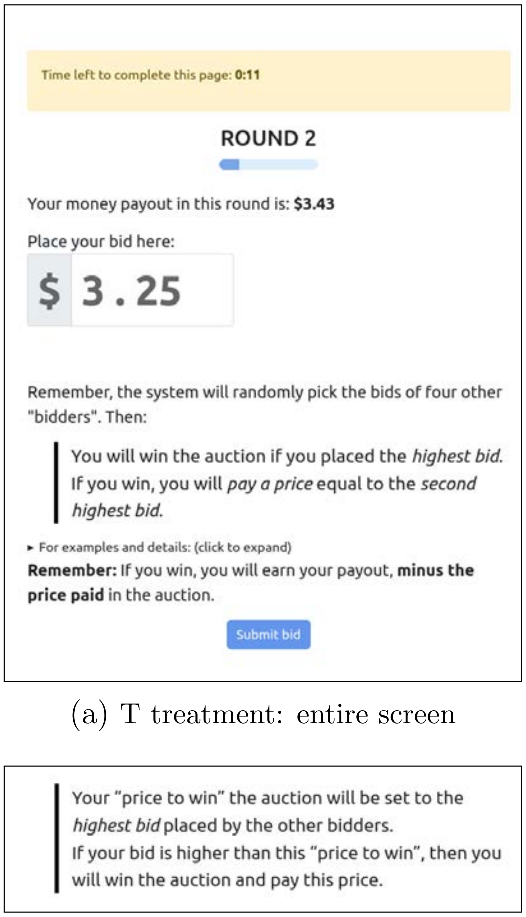
Remember: the other two citizens' votes are chosen at random, with each vote between 1 and 5 equally likely for each citizen.

(a) T treatment: entire screen

> The "obtainable numbers" will be the votes of the two other citizens, and all the numbers between them. Out of these "obtainable numbers", the one *closest to your own vote* will be elected.

(b) M treatment: changed text

Figure 5: Median Voting Screen. The participant chose number 2, but has not yet submitted their vote.



Time left to complete this page: **0:11**

ROUND 2

Your money payout in this round is: **$3.43**

Place your bid here:

$ 3.25

Remember, the system will randomly pick the bids of four other "bidders". Then:

> You will win the auction if you placed the *highest bid.* If you win, you will *pay a price* equal to the *second highest bid.*

▸ For examples and details: (click to expand)

Remember: If you win, you will earn your payout, **minus the price paid** in the auction.

(a) T treatment: entire screen

> Your "price to win" the auction will be set to the *highest bid* placed by the other bidders. If your bid is higher than this "price to win", then you will win the auction and pay this price.

(b) M treatment: changed text

Figure 6: Second-Price Auction Screen. The participant entered $3.25, but has not yet submitted their bid.

As the figures show, in Median participants vote for one of five candidate numbers (1–5) by clicking on their chosen number. In Auction, participants enter a bid be-

[36] For both mechanisms, the distribution from which the private information is drawn is tailored to provide a consistent and reasonably high expected monetary reward for straightforward behavior; see the online experimental materials for details.

tween $0.00 and $5.00. In both mechanisms, the only main-screen difference between T and M is the larger-font text to the right of the thick black vertical line (and the way the outcome is described in the examples—which participants can see again using a "click to expand" link).

9.3 Data

The experiment was conducted on February 4–5, 2022. We recruited US 18+ participants on the Prolific platform (https://www.prolific.co). 229 participants clicked on the experiment's link; 220 progressed beyond the informed-consent and (pre-treatment) introductory screens; 200 completed the experiment.[37] Subjects received $2 for participation and earned an additional $6.33 on average through their voting and bidding. Median respondent age is 29; 51% are female; and median completion time is 20.6 minutes.

9.4 Results

Table 6 presents our main results. In Median, participants vote for their highest-earning candidate 70 percent of the time under T, and 80 percent of the time under M. A two-sided t-test for equality of means yields a p-value of 0.01. This difference is arguably large: it corresponds to a reduction by one-third in non-straightforward behavior (from 30 to 20 percent). The difference is larger still when comparing the fraction of participants who play straightforwardly *in every round*: 26 percent in T versus 52 percent in M; $p = 0.0003$. (Figure 7 shows the full distributions.) Participants also earn 6 percent more in M ($3.00) than in T ($2.83), however this result is not statistically strong ($p = 0.10$).

In Auction, we find essentially no difference across the treatments in either straightforward bidding or earning. This (non-)result holds regardless of the distance d between bids and private values that we consider acceptable.[38]

In both mechanisms, the data may suggest that respondents play more SF in

[37] Of the 20 incompletes who progressed beyond these introductory screens, 16 dropped during Median setup screens (5 in T, 11 in M), 3 during Auction setup screens (2 in T, 1 in M), and one during Auction rounds (1 in T). Our (preregistered) intention was to recruit a total of 200 participants. Due to miscommunication, our RA kept recruiting until we had 200 *completes*. Excluding the additional completes (beyond the first 200 participants who started the experiment) does not affect our results more than trivially, but the % Straightforward p-value reported in Table 6 below for Median changes from 0.01 to 0.02. See Appendix A for additional analysis of the dropouts.

[38] Table 6 reports the results for Auction with a stringent definition of straightforward play, resulting in low straightforward percentages. See Appendix A for more lenient definitions, which result in straightforward percentages closer to those in Median.

Table 6: Straightforward Play and Earning by Treatment.

	Median Voting			**Second-Price Auction**		
	Trad. (T)	Menu (M)	p-value	Trad. (T)	Menu (M)	p-value
% Straightforward	70 (3)	80 (3)	0.01	37 (4)	34 (3)	0.55
% All Straightforward	26 (4)	52 (5)	0.0003	13 (3)	10 (3)	0.66
$ Earning	2.83 (0.07)	3.00 (0.07)	0.10	3.40 (0.26)	3.40 (0.24)	0.98
N Participants	100	100		100	100	

Notes: "% Straightforward": participants' average fraction of the ten rounds with straightforward play (in Median, voting for the ideal number; in Auction, bidding within $0.10 of the private value). "% All Straightforward": share of participants with straightforward play in all ten rounds. "$ Earnings": participants' average dollar amount earned across all rounds. Standard errors are in parentheses. p-values for Straightforward and Earnings: two-sample, two-sided equality-of-means t-test (Welch's t-test); p-values for All Straightforward: two-sample, two-sided equality-of-proportions test.

last-five than in first-five rounds, perhaps more so in T than in M and in Auction than in Median, but our main findings remain essentially the same (see Table A.1 on page A.1). We also compare differences in rates of straightforward play in Auction based on treatment in *Median* (recall, participants play Median before they play Auction), and find no effect (see Table A.2 on page A.1).

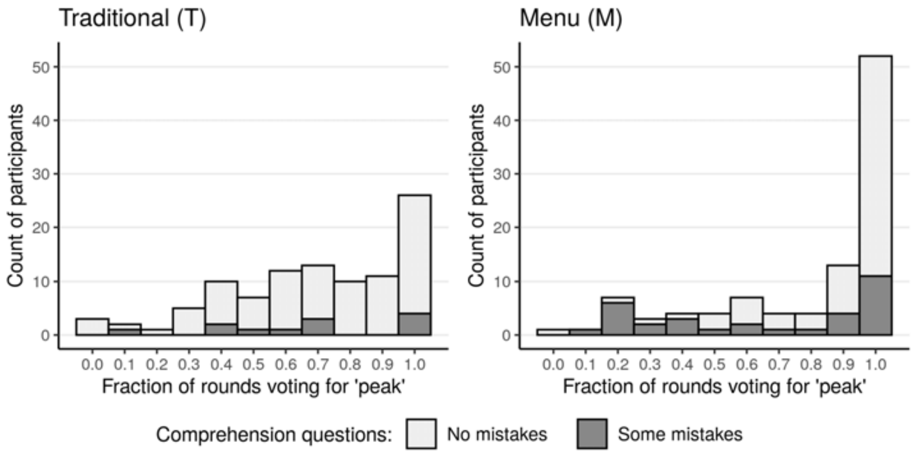


Figure 7: Median Voting. Distributions of participants' fraction of rounds with straightforward play. "No/Some mistakes" patterns divide participants by performance on the comprehension questions. $N = 100$ in each of T and M.

To further investigate our strong results in Median, Figure 7 and Table 7 report on the relationship between straightforward play and participants' understanding of the mechanism's description, as measured by the comprehension questions. The table highlights three findings. First, in M, answering all comprehension questions correctly on first attempt is associated with significantly more straightforward play: 87 percent in the "No Mistakes" column versus 65 percent in the "Some Mistakes" column, $p = 0.002$. Further subdividing participants with some mistakes by the number of examples in which they made mistakes—in the three rightmost columns—similarly suggests a monotonically decreasing trend (although the subsamples are small). In contrast, in T we find no such differences, with 70 versus 67 percent straightforward play in the No versus Some Mistakes columns ($p = 0.79$). Second, the comprehension questions seem harder to correctly answer on first attempt in M than in T: N Participants = 68 versus 88, respectively, in the "No mistakes" column. Third, straightforward play is rather similar among participants in M with at least one mistake ($N = 32$, 65 percent straightforward play) and participants in T—either those with mistakes ($N = 12$, 67 percent) or those without them ($N = 88$, 70 percent). Taken together, these three findings suggest that while comprehension may be more challenging for some in M than in T, M significantly increases straightforward play for quick comprehenders while not decreasing it for the others.

Table 7: Median Voting. Straightforward Play by Treatment and Comprehension Mistakes.

		No Mistakes	Some Mistakes	p-value	# of examples w/ mistakes: 1	2	3
T	% Straightforward	70 (3)	67 (8)	0.79	66 (11)	50 (9)	100 (NA)
	N Participants	88	12		7	3	2
M	% Straightforward	87 (3)	65 (6)	0.002	69 (11)	67 (8)	50 (14)
	N Participants	68	32		10	17	5

Notes: p-value: two-sided Welch's test between participants who correctly answer all questions on first try ("No Mistakes") and those who do not ("Some Mistakes"). Standard errors are in parentheses. Those who made at least one mistake in the comprehension questions are further subdivided by the number of examples (each of which contained two questions) out of three where they made mistakes.

While we can only speculate why, unlike in Median, in Auction we find no difference in straightforward play between M and T, one common comprehension-question mistake in Auction may provide some insight. Several participants appear to mistake the mechanism for a first-price auction: they mistakenly write their own bid (which is given to them in the question) as the price they will pay if they win the auction.

In M versus T, 18 versus 8 participants make this mistake (equality-of-proportions $p = 0.06$). Since this misconception about a second-price auction is something we specifically hoped that a menu description could help dispel, this (ex post) finding may hint at a fundamental problem with our menu description in Auction—a problem that future research can explore, but that our theory does not shed light on. For additional discussion and analysis (and samples of the comprehension pages), see Appendix A.

Overall, the experimental results in this section suggest that while our first-attempt designs of menu descriptions still need work, they already show promise. On the one hand, in both Median and Auction, our menu descriptions—which are conspicuously longer than traditional descriptions—appear more difficult to grasp. On the other hand, our menu descriptions do not meaningfully reduce average straightforward play in Auction, while significantly increasing it in Median.[39] Furthermore, in Median, our menu description does not on average dramatically decrease straightforward play relative to the traditional description even when focusing only on those who do not easily understand it—but it dramatically increases such play among those who quickly understand it. We return to these findings in our concluding remarks, where we also discuss their potential distributional implications, and propose future directions for needed additional empirical work.

10 Related work

Our paper is most directly inspired by the contemporary "strategic simplicity" program in mechanism design. A cornerstone of this literature is Li (2017), which introduces OSP mechanisms as a refinement of strategyproofness that might explain why certain interactive mechanisms might be easier to recognize as strategyproof. Unfortunately, many desirable mechanism do not have OSP implementations. This is the case for TTC (Li, 2017) and DA (Ashlagi and Gonczarowski, 2018) (in both of these, OSP implementations are possible only in rare special cases of institutions' priorities, Troyan, 2019; Mandal and Roy, 2021; Thomas, 2021), as well as for Median Voting and two-item welfare-maximizing auctions for bidders with additive valuations (Bade and Gonczarowski, 2017; Arribillaga et al., 2020).[40]

[39] One hypothesis, yet untested, as to why we find a difference across M and T in Median but not in Auction is that in Auction, M and T are simply too similar. Indeed, they essentially just switch the order of two sentences and fix the grammar accordingly. In contrast, in Median there is a genuine algorithmic change from T to M.

[40] A different line of work also considers notions of strategic simplicity that are weaker than strategyproofness (Börgers and Li, 2019; Fernandez, 2020; Troyan and Morrill, 2020; Chen and Möller, 2021; Mennle and Seuken, 2021).

The empirical paper by Breitmoser and Schweighofer-Kodritsch (2022) takes a deeper look at OSP vs. static implementations of second-price auctions. That paper experiments with describing/framing a static, direct-revelation auction as the result of an OSP ascending auction that is simulated using participants' directly-reported bid. They show that this alternative framing can improve the rate of straightforward behavior. Our paper can be viewed in this context as observing that once a direct-revelation mechanism is described as an extensive-form mechanism whose run is simulated using participants' reports, one can in fact describe it as a *different* extensive-form mechanism to each participant. Explaining strategyproofness to only one player (the player reading the description) then becomes possible for all strateghproof mechanisms; in fact, menu descriptions describe any strategyproof rule to one player at a time as a mechanism that is OSP, and even *strongly* OSP (Pycia and Troyan, 2023), for that player. We note, however, an important conceptual difference between our framework and earlier ones (discussed as the second premise of our paper in Section 1). Every OSP mechanism is intended to be simple to play, so the challenge within that framework is finding any OSP mechanism. In contrast, not every menu description is simple (e.g., we view the "brute force" approach in Example 2.8 as complex and undesirable), so the challenge lies in finding appealing menu descriptions.

We also contribute to the literature on structural properties of matching mechanisms. Different notions of a player's budget set have been defined in this literature, especially in matching markets with a continuum of agents (Azevedo and Leshno, 2016; Leshno and Lo, 2021; Immorlica et al., 2020). In finite markets, budget sets are different sets from the player's menu.[41] Nonetheless, for some mechanisms, budget sets coincide with the player's menu in continuum or limit markets, which makes them useful tools in such markets to reason about approximate strategyproofness (Azevedo and Budish, 2019) and to guide information acquisition (Immorlica et al., 2020). With respect to these papers, our menu characterizations could also be conceptually seen as showing how to remove large-market assumptions to derive precise rather than approximate properties of mechanisms in markets of any size. The budget equilibria

[41] For the specific mechanism of DA, the budget set of applicant i is typically defined as the set of institutions j such that i has at least as high priority at j as $\mu(j)$, where μ is the outcome of DA. (Note that this set depends on both t_i and t_{-i}.) In finite markets, the menu in DA is not given in a simple way by this budget set or the priorities. For example: Let institutions h_1, h_2, h_3, and h_4 have priorities $h_1 : d_1 \succ d_2$; $h_2 : d_4 \succ d_3 \succ d_2 \succ d_1$; $h_3 : d_3$; $h_4 : d_2 \succ d_4$. Let applicants d_1, d_2, d_3, and d_4 have preferences $d_1 : h_1 \succ \ldots$; $d_2 : h_1 \succ h_2 \succ h_4 \succ \ldots$; $d_3 : h_3 \succ \ldots$; $d_4 : h_4 \succ h_2 \succ \ldots$. Then DA pairs h_i to d_i for each $i = 1, \ldots, 4$, and h_2 is in the budget set of applicants d_2, d_3 and d_4. However, h_2 is in the menu of applicants d_1, d_2, and d_4. So, despite d_3 being higher priority than d_2 at h_2, h_2 is not on d_3's menu; despite d_1 being lower priority than d_2 at h_2, h_2 *is* on d_1's menu.

defined in Segal (2007) are also different from players' menus. There is some conceptual connection, though: where menus can explain ex ante why strategyproofness holds, budget equilibria can help verify ex post that (for example) the matching is stable. The menu in DA is also a distinct notion from the set of stable partners of an applicant. Even though each applicant gets her favorite choice out of her set of stable partners (Gale and Shapley, 1962), this set cannot help to explain strategyproofness in the way that an applicant's menu can, because an applicant's report can affect this set.

Leshno and Lo (2021), in their Proposition 2, give a characterization of the menu of TTC in finite markets (by embedding such markets in continuum markets), though this characterization does not seem targeted towards an alternative description of TTC. That paper mentions that the fact that this menu is independent of an applicant's type (which, by Hammond, 1979, is true for any strategyproof mechanisms) could help explain strategyproofness, but does not discuss how or whether the menu relates to any concrete description. A vast literature develops techniques for analyzing DA by incrementally modifying submitted preference lists (e.g., Gale and Sotomayor, 1985; Immorlica and Mahdian, 2005; Hatfield and Milgrom, 2005; Gonczarowski, 2014; Ashlagi et al., 2017; Cai and Thomas, 2022, to name a few)—the direct proof in Appendix C of Theorem 3.1 builds upon such techniques. We are not aware of any prior characterizations of the menu in DA.[42] We also do not know of any other paper that analyzes different ways to describe multi-player mechanisms in terms of menus, seeks simpler menu descriptions, or provides a formalism of the required trade-offs.

Our Section 9 contributes to the experimental literature on behavioral mechanism design. The most closely related experimental paper to ours is the recent working paper of Katuščák and Kittsteiner (2020), who also, independently, suggest describing mechanisms via menu descriptions.[43] That paper runs an experiment on TTC using a menu description that is very close to that of Example 2.8, which essentially calculates the menu by iterating over possible reports and running the standard TTC description for each report to determine which outcomes are possible.

Additional related experimental papers explore behavior across different social choice rules (Kagel and Levin, 1993; Chen and Sönmez, 2006; Pais and Pintér, 2008,

[42] Certain other properties of DA (e.g., in Blum et al., 1997; Adachi, 2000) and of unit-demand auctions (e.g., in Gul and Stacchetti, 2000; Alaei et al., 2016), despite not being studied with relation to menus, bear some technical similarity to the menu calculation in Description 1. However, the proofs seem unrelated.

[43] There is no intersection between our paper and theirs beyond this suggestion and Example 2.8 (the description we use to illustrate non-simple menu descriptions), both of which we had before learning of their paper. When we learned of their paper, the only main result of our paper that was not completed was our experiment.

among others). Other papers explore *advice* that explicitly informs participants of the strategyproofness of the mechanisms, including Masuda et al. (2022) (for auctions) and Guillen and Hakimov (2018) (for TTC). Somewhat relatedly, Danz et al. (2020) study the empirical effect of providing or withholding information on the exact workings of belief elicitation mechanisms. There is also a vast literature of empirical papers studying or guiding real-world implementations of matching mechanisms. Most related to our paper are those that study how real-world participants understand or interface with the mechanisms—see Pathak and Sönmez (2008); Arteaga et al. (2022); Grenet et al. (2022), among others, as well as the review article Pathak (2017).

Our paper is also technically inspired by the literature within computer science studying menus. These works largely focus on single-player mechanisms, particularly in the context of the revenue-maximizing monopolist problem with one buyer. The most common object of study is the structural complexity of the menu, i.e., how many different entries are offered on the menu (Hart and Nisan, 2019; Daskalakis et al., 2017; Babaioff et al., 2022; Saxena et al., 2018; Gonczarowski, 2018). Most relatedly within computer science are Dobzinski (2016); Dobzinski et al. (2022), who consider the properties of menus in multi-buyer auctions in some detail, largely as a tool for analyzing mechanisms and bounding communication complexity.[44] Because our viewpoint lies on defining (rather than analyzing) mechanisms in terms of menus, many technical distinctions arise. (For instance, our bounds are on the memory requirements of algorithms and/or on how many times they can read their input, rather than on their communication complexity.) We are not aware of prior algorithmic work on finding the menu in matching mechanisms (be it DA or otherwise), for which we give multiple optimal (in terms of complexity) algorithms, as well as impossibility results.

11 Conclusion

Strategyproofness has long been proposed as a way to make mechanisms fair by theoretically leveling the playing field for players who do not strategize well (Pathak and Sönmez, 2008). We warmly embrace this agenda. We however observe that if disparities remain in participants' *understanding* of strategyproofness, then the participants remain on uneven footing.[45] These disparities may be hard to avoid, as

[44] Brânzei and Procaccia (2015); Golowich and Li (2022) study the computational complexity of checking whether a mechanism, given its extensive- or normal-form representation, is strategyproof.

[45] Robertson et al. (2021) report a case of one parent saying "It's definitely convoluted. It's definitely multilayered, it's complex. And that favors people who have the time and the wherewithal to figure it out. [. . . T]he complexity invites accusations of [corruption] and does not inspire trust."

people appear to differ dramatically in their intuitive grasp of strategyproofness. For example, many people seemingly fail to realize, even when given carefully constructed explanations, that a second-price auction is *not* a real-life haggling process where the bidder can influence their price, and that Deferred Acceptance is *not* a real-life job hunt where one can miss out on otherwise-obtainable satisfactory positions because of time wasted applying to reach positions. Menu descriptions may help clarify such distinctions in these and other real-world strategyproof mechanisms.

At present, menu descriptions still have several limitations, both empirical and theoretical. Empirically, as our experimental results for Second-Price Auction suggest, menu descriptions alone may not be sufficient to convey the crucial distinctions discussed above. Moreover, our first attempt at writing them for real participants suggests that menu descriptions too may not always be equally easily understood by everybody. Eliminating disparities in understanding remains an important challenge, especially as ease of understanding strongly predicts straightforward play in our Median Voting data.

Theoretically, a framework for mechanism descriptions is not a concrete behavioral model of how participants make decisions: our theory provides no quantitative behavioral predictions (e.g., of rates of straightforward behavior) under menu versus non-menu descriptions. (In this aspect, our paper is analogous to Li (2017), which also begins by theoretically motivating certain ways of presenting mechanisms that may make strategyproofness easy to see without predicting how much this could affect behavior; but stands in contrast to Dreyfuss et al. (2022b,a), which begin from behavioral models, and use quantitative predictions derived from these models to seek desirable implementations of mechanisms.) Closely related, our theory provides no guidance on what to do if, e.g., due to issues of trust, the description of the mechanism cannot be personalized for each individual reading it.[46] A more general theory that orders mechanisms by the cognitive difficulty of finding one's dominant strategy could provide guidance on which description to use when, for whatever reason, some descriptions are ruled out.

Menu descriptions provide other avenues for investigation. Since a mechanism is strategyproof if and only if each player always gets her favorite outcome from her menu, the menu also provides a natural way to *define* strategyproofness. Inspired by this, one could consider a "black box" menu description that simply tells players that *some* menu will be calculated using only other players' types, without specifying any

[46] However, we remark that traditional descriptions of mechanisms also require trust, e.g., that the description is accurate (Akbarpour and Li, 2020).

details. However, beyond conveying strategyproofness, this description conveys no additional information about the mechanism. Thus, this (partial) description does not meet the standard to which we hold all descriptions in the present paper: unambiguously describing a player's outcome in a mechanism. Still, it would be interesting to investigate the behavior of real-world participants under such descriptions. We are currently developing such experiments, along with an experiment on our new menu description of Deferred Acceptance.

This paper applies theoretical tools from computer science to social science. Traditionally, computer science asks whether a given algorithm can be easily run on a computer, after all inputs are known. In contrast, the simplicity conditions in this paper are intended to capture whether a given algorithm can be easily conveyed to humans, before any input is known. Using these simplicity conditions, we approach the problem of explaining strategyproofness, try to address it by phrasing descriptions of matching mechanisms in terms of menus, and examine tradeoffs between describing the matching and describing the menu. As more parts of modern life are affected by algorithms and mechanisms, their interpretability may be of increasing importance.[47] Future work may study other properties one might wish to expose (for example, fairness or optimality), find context-specific methods to expose these properties, and study sets of available options and tradeoffs in a variety of different settings.

References

A. Abdulkadiroğlu and T. Sönmez. School choice: A mechanism design approach. *American Economic Review*, 93(3):729–747, 2003.

A. Abdulkadiroğlu, Y.-K. Che, and Y. Yasuda. Resolving conflicting preferences in school choice: The "Boston mechanism" reconsidered. *American Economic Review*, 101(1): 399–410, 2011.

H. Adachi. On a characterization of stable matchings. *Economics Letters*, 68(1):43–49, 2000.

M. Akbarpour and S. Li. Credible auctions: A trilemma. *Econometrica*, 88(2):425–467, 2020.

S. Alaei, K. Jain, and A. Malekian. Competitive equilibria in two-sided matching markets with general utility functions. *Operations Research*, 64(3):638–645, 2016.

[47] The White House's recent "Blueprint for an AI Bill of Rights" (The White House Office of Science and Technology Policy, 2022) underlines the following as one of their five key principles:

> **Notice and Explanation**: You should know when an automated system is being used and understand how and why it contributes to outcomes that impact you.

R. P. Arribillaga, J. Massó, and A. Neme. On obvious strategy-proofness and single-peakedness. *Journal of Economic Theory*, 186:104992, 2020.

F. Arteaga, A. J. Kapor, C. A. Neilson, and S. D. Zimmerman. Smart matching platforms and heterogeneous beliefs in centralized school choice. *Quarterly Journal of Economics*, 137(3):1791–1848, 2022.

I. Ashlagi and Y. A. Gonczarowski. Stable matching mechanisms are not obviously strategy-proof. *Journal of Economic Theory*, 177:405–425, 2018.

I. Ashlagi, Y. Kanoria, and J. D. Leshno. Unbalanced random matching markets: The stark effect of competition. *Journal of Political Economy*, 125(1):69 – 98, 2017. Abstract in Proceedings of the 14th ACM Conference on Electronic Commerce (EC 2013).

E. M. Azevedo and E. Budish. Strategy-proofness in the large. *The Review of Economic Studies*, 86(1):81–116, 2019.

E. M. Azevedo and J. D. Leshno. A supply and demand framework for two-sided matching markets. *Journal of Political Economy*, 124(5):1235–1268, 2016.

M. Babaioff, Y. A. Gonczarowski, and N. Nisan. The menu-size complexity of revenue approximation. *Games and Economic Behavior*, 134:281–307, 2022. Extended abstract in Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2017).

S. Bade and Y. A. Gonczarowski. Gibbard-satterthwaite success stories and obvious strategyproofness. In *Proceedings of the 18th ACM Conference on Economics and Computation (EC)*, page 565, 2017.

M. Balinski and T. Sönmez. A tale of two mechanisms: student placement. *Journal of Economic theory*, 84(1):73–94, 1999.

S. Barberà, H. Sonnenschein, and L. Zhou. Voting by committees. *Econometrica: Journal of the Econometric Society*, pages 595–609, 1991.

Y. Blum, A. E. Roth, and U. G. Rothblum. Vacancy chains and equilibration in senior-level labor markets. *Journal of Economic theory*, 76(2):362–411, 1997.

I. Bó and R. Hakimov. Pick-an-object mechanisms. *Management Science*, 2023.

T. Börgers and J. Li. Strategically simple mechanisms. *Econometrica*, 87(6):2003–2035, 2019.

S. Brânzei and A. D. Procaccia. Verifiably truthful mechanisms. In *Proceedings of the 6th Conference on Innovations in Theoretical Computer Science (ITCS)*, page 297–306, 2015.

Y. Breitmoser and S. Schweighofer-Kodritsch. Obviousness around the clock. *Experimental Economics*, 25:483–513, 2022.

L. Cai and C. Thomas. The short-side advantage in random matching markets. In *Proceedings of the 5th SIAM Symposium on Simplicity in Algorithms (SOSA)*, pages 257–267, 2022.

D. L. Chen, M. Schonger, and C. Wickens. oTree—an open-source platform for laboratory, online, and field experiments. *Journal of Behavioral and Experimental Finance*, 9(C): 88–97, 2016.

Y. Chen and M. Möller. Regret-free truth-telling in school choice with consent. Mimeo, 2021.

Y. Chen and T. Sönmez. School choice: An experimental study. *Journal of Economic Theory*, 127(1):202–231, 2006.

D. Danz, L. Vesterlund, and A. J. Wilson. Belief elicitation: Limiting truth telling with information on incentives. Working paper 27327, National Bureau of Economic Research, 2020.

C. Daskalakis, A. Deckelbaum, and C. Tzamos. Strong duality for a multiple-good monopolist. *Econometrica*, 85(3):735–767, 2017.

S. Dobzinski. Computational efficiency requires simple taxation. In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2016.

S. Dobzinski, S. Ron, and J. Vondrák. On the hardness of dominant strategy mechanism design. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 690–703, 2022.

B. Dreyfuss, O. Glicksohn, O. Heffetz, and A. Romm. Deferred acceptance with news utility. In preparation, 2022a.

B. Dreyfuss, O. Heffetz, and M. Rabin. Expectations-based loss aversion may help explain seemingly dominated choices in strategy-proof mechanisms. *Forthcoming in American Economic Journal: Microeconomics*, 2022b.

M. A. Fernandez. Deferred acceptance and regret-free truth-telling. Mimeo, 2020.

D. Gale and L. S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–14, 1962.

D. Gale and M. Sotomayor. Ms. Machiavelli and the stable matching problem. *American Mathematical Monthly*, 92(4):261–268, 1985.

L. Golowich and S. Li. On the computational properties of obviously strategy-proof mechanisms. Mimeo, 2022.

Y. A. Gonczarowski. Manipulation of stable matchings using minimal blacklists. In *Proceedings of the 15th ACM Conference on Economics and Computation (EC)*, page 449, 2014.

Y. A. Gonczarowski. Bounding the menu-size of approximately optimal auctions via optimal-transport duality. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 123–131, 2018.

J. Grenet, Y. He, and D. Kübler. Preference discovery in university admissions: The case for dynamic multioffer mechanisms. *Journal of Political Economy*, 130(6):1427–1476, 2022.

P. Guillen and R. Hakimov. The effectiveness of top-down advice in strategy-proof mechanisms: A field experiment. *European Economic Review*, 101:505–511, 2018.

F. Gul and E. Stacchetti. The english auction with differentiated commodities. *Journal of Economic theory*, 92(1):66–95, 2000.

R. Hakimov and D. Kübler. Experiments on centralized school choice and college admissions: A survey. *Experimental Economics*, 24:434–488, 2021.

P. J. Hammond. Straightforward individual incentive compatibility in large economies. *Review of Economic Studies*, 46(2):263–282, 1979.

S. Hart and N. Nisan. Approximate revenue maximization with multiple items. *Journal of Economic Theory*, 172:313–347, 2017. Abstract in Proceedings of the 13th ACM Conference on Electronic Commerce (EC 2012).

S. Hart and N. Nisan. Selling multiple correlated goods: Revenue maximization and menu-size complexity. *Journal of Economic Theory*, 183:991–1029, 2019. Abstract ("The menu-size complexity of auctions") in Proceedings of the 14th ACM Conference on Electronic Commerce (EC 2013).

A. Hassidim, D. Marciano, A. Romm, and R. I. Shorrer. The mechanism is truthful, why aren't you? *American Economic Review*, 107(5):220–224, 2017.

A. Hassidim, , A. Romm, and R. I. Shorrer. The limits of incentives in economic matching procedures. *Management Science*, 67(2):951–963, 2021.

J. W. Hatfield and P. R. Milgrom. Matching with contracts. *American Economic Review*, 95(4):913–935, 2005.

N. Immorlica and M. Mahdian. Marriage, honesty, and stability. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 53–62, 2005.

N. Immorlica, J. Leshno, I. Lo, and B. Lucier. Information acquisition in matching markets: The role of price discovery. Mimeo, 2020. URL https://ssrn.com/abstract=3705049.

R. W. Irving and P. Leather. The complexity of counting stable marriages. *SIAM Journal on Computing*, 15(3):655–667, 1986.

J. H. Kagel and D. Levin. Independent private value auctions: Bidder behaviour in first-, second-and third-price auctions with varying numbers of bidders. *Economic Journal*, 103(419):868–879, 1993.

P. Katuščák and T. Kittsteiner. Strategy-proofness made simpler. Mimeo, 2020.

J. D. Leshno and I. Lo. The cutoff structure of top trading cycles in school choice. *Review of Economic Studies*, 88(4):1582–1623, 2021.

S. Li. Obviously strategy-proof mechanisms. *American Economic Review*, 107(11):3257–87, 2017.

A. Mackenzie and Y. Zhou. Menu mechanisms. *Journal of Economic Theory*, 204:105511, 2022.

P. Mandal and S. Roy. Obviously strategy-proof implementation of assignment rules: A new characterization. *International Economic Review*, 63(1):261–290, 2021.

T. Masuda, R. Mikami, T. Sakai, S. Serizawa, and T. Wakayama. The net effect of advice on strategy-proof mechanisms: An experiment for the Vickrey auction. *Experimental Economics*, 25:902–951, 2022.

V. Meisner and J. von Wangenheim. Loss aversion in strategy-proof school-choice mechanisms. *Journal of Economic Theory*, 207:105588, 2023.

T. Mennle and S. Seuken. Partial strategyproofness: Relaxing strategyproofness for the random assignment problem. *Journal of Economic Theory*, 191:105144, 2021.

NMS. The matching algorithm - explained, 2020. URL https://www.youtube.com/watch?v=kVTwXNawpbk. Video produced by National Matching Services.

J. Pais and Á. Pintér. School choice and information: An experimental study on matching mechanisms. *Games and Economic Behavior*, 64(1):303–328, 2008.

P. A. Pathak. What really matters in designing school choice mechanisms. *Advances in Economics and Econometrics*, 1:176–214, 2017.

P. A. Pathak and T. Sönmez. Leveling the playing field: Sincere and sophisticated players in the boston mechanism. *American Economic Review*, 98(4):1636–52, 2008.

M. Pycia and P. Troyan. A theory of simplicity in games and mechanism design. *Econometrica*, 2023. Abstract ("Obvious Dominance and Random Priority") at Proceedings of the 20th ACM Conference on Economics and Computation (EC 2019).

A. Rees-Jones. Suboptimal behavior in strategy-proof mechanisms: Evidence from the residency match. *Games and Economic Behavior*, 108(C):317–330, 2018.

S. Robertson, T. Nguyen, and N. Salehi. Modeling assumptions clash with the real world: Transparency, equity, and community challenges for student assignment algorithms. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, 2021.

A. E. Roth. The economist as engineer: Game theory, experimentation, and computation as tools for design economics. *Econometrica*, 70(4):1341–1378, 2002.

A. Rubinstein, R. R. Saxena, C. Thomas, S. M. Weinberg, and J. Zhao. Exponential communication separations between notions of selfishness. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 947–960, 2021.

R. R. Saxena, A. Schvartzman, and S. M. Weinberg. The menu complexity of "one-and-a-half-dimensional" mechanism design. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2026–2035, 2018.

I. Segal. The communication requirements of social choice rules and supporting budget sets. *Journal of Economic Theory*, 136(1):341–378, 2007.

R. I. Shorrer and S. Sóvágó. Obvious mistakes in a strategically simple college admissions environment. Discussion Paper 2017-107/V, Tinbergen Institute, 2017.

The White House Office of Science and Technology Policy. Blueprint for an AI bill of rights: Making automated systems work for the american people, 2022. URL https://www.whitehouse.gov/ostp/ai-bill-of-rights/.

C. Thomas. Classification of priorities such that deferred acceptance is OSP implementable. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, page 860, 2021.

P. Troyan. Obviously strategy-proof implementation of top trading cycles. *International Economic Review*, 60(3):1249–1261, 2019.

P. Troyan and T. Morrill. Obvious manipulations. *Journal of Economic Theory*, 185: 104970, 2020.

A Additional Experimental Analysis

In this appendix, we provide additional analysis of our experimental data.

To begin, we check for learning from experience, fatigue, boredom, and other potential behavior changes over time by comparing rates of straightforward play across the first-five vs. last-five rounds for each Mechanism and Treatment (Table A.1). We also check for cross-effects of the treatment from Median on outcomes in Auction (Table A.2).

Table A.1: Straightforward Play by Treatment, First vs. Last Five Rounds.

	Median Voting				**Auction**			
	T	M	p	M+T	T	M	p	M+T
Rounds 1-5 (%)	67	79	0.008	73	31	31	0.87	31
Rounds 6-10 (%)	72	81	0.04	76	46	41	0.39	43
p	0.26	0.65		0.27	0.007	0.05		0.0008
N Participants (#)	100	100		200	100	100		200

Notes: "T" (resp., "M"): Traditional (resp., Menu) treatment for the relevant mechanism; "T+M": all participants. Table displays the average fraction of rounds with straightforward play, measured as in Table 6 on page 38. For comparisons between T and M, p values are two-sample, two-sided equality-of-means t-test between subjects; for comparisons between the first five and last 5 rounds, the t-tests are within subjects.

Table A.2: Effect of treatment in Median on Straightforward Play in Auction.

Median Treatment	Auction Treatment			
	T	M	p	M+T
T (%)	36	32	0.57	34
M (%)	39	37	0.78	38
p	0.60	0.47		0.38
N Participants (#)	100	100		200

Notes: "T" (resp., "M"): Traditional (resp., Menu) treatment for the relevant mechanism; "T+M": either treatment in Auction. Table displays the average fraction of rounds with straightforward play in Auction, measured as in Table 6 on page 38. All p values are two-sample, two-sided equality-of-means t-test between subjects.

We also remark on one slight difference between our paper and our preregistration. Namely, due to a miscommunication with our RA, we did not record the length of time participants spent on different screens throughout the treatment. (The only other deviation from our preregistration was mentioned in footnote 37 on page 37).

A.1 Median

What do participants vote for? One can ask: *when participants do not play straightforwardly, what strategies are they following?* Table A.3 reports how votes were placed, conditional on the location of the peak in the earnings table. We observe that there is a clear trend towards voting for numbers adjacent to the peak. In T and M respectively, only 12 and 14 percent of all non-peak votes are for a number more than 1 farther from the peak. If non-peak votes were uniformly distributed over all non-peak numbers, this statistic should be 60 percent. There may be a slight trend for non-peak votes for the second-highest valued number.

One might conjecture that participants may be confusing the mechanism with another. However, if the confused-with mechanism takes the (rounded) *average* of the three participants votes, then all of the earnings tables in our distribution with a peak at 1 or 2 would maximize their utility by voting for 1 (and similarly, all those with a peak at 4 or 5 would want to vote for 5). Thus, confusion with an averaging mechanism cannot explain the large fraction of votes that are placed closer to the middle (i.e., to 3) than the peak.

All told, it is hard to identify systematic non-straightforward behavior beyond the finding that people generally vote close to their optimal vote.

Table A.3: The marginal tables of voting across all rounds of all participants.

Traditional Treatment

Ideal Num.	Total Count	Percent (%) Voted For:					
		1	2	3	4	5	2nd
1	234	68	27	4	1	0	–
2	186	16	70	12	2	1	17
3	208	1	14	77	8	0	17
4	190	1	1	13	69	17	19
5	182	0	1	8	27	64	–

Menu Treatment

Ideal Num.	Total Count	Percent (%) Voted For:					
		1	2	3	4	5	2nd
1	200	72	24	2	1	1	–
2	201	5	84	9	1	0	8
3	208	1	7	84	8	0	11
4	195	1	2	11	85	2	6
5	195	1	1	4	19	75	–

Notes: When the Ideal Number (the peak of the single-peaked earnings table) is not 1 or 5, the 2nd column shows the fraction of rounds where the participant voted for their *second*-highest valued number. A single participant in M failed to submit a vote within the 30-second window in one single round.

Comprehension questions. Figure A.1 provides samples of the comprehension-questions page. The comprehension questions in T appear noticeably easier than those in M—12 of the 100 T participants failed to answer all the comprehension questions on first attempt;[1] the same number is 32 in M.[2] While we did not expect this, in hindsight we speculate that it is due to the increased complexity of the menu description, namely, the fact that the menu description (and the corresponding comprehension questions) necessarily compute the menu, an object not found in the traditional description. (Our menu descriptions are also conspicuously longer than their traditional counterparts, and their comprehension questions are phrased in a way that is less self-contained, i.e. they require remembering how the menu is calculated.)

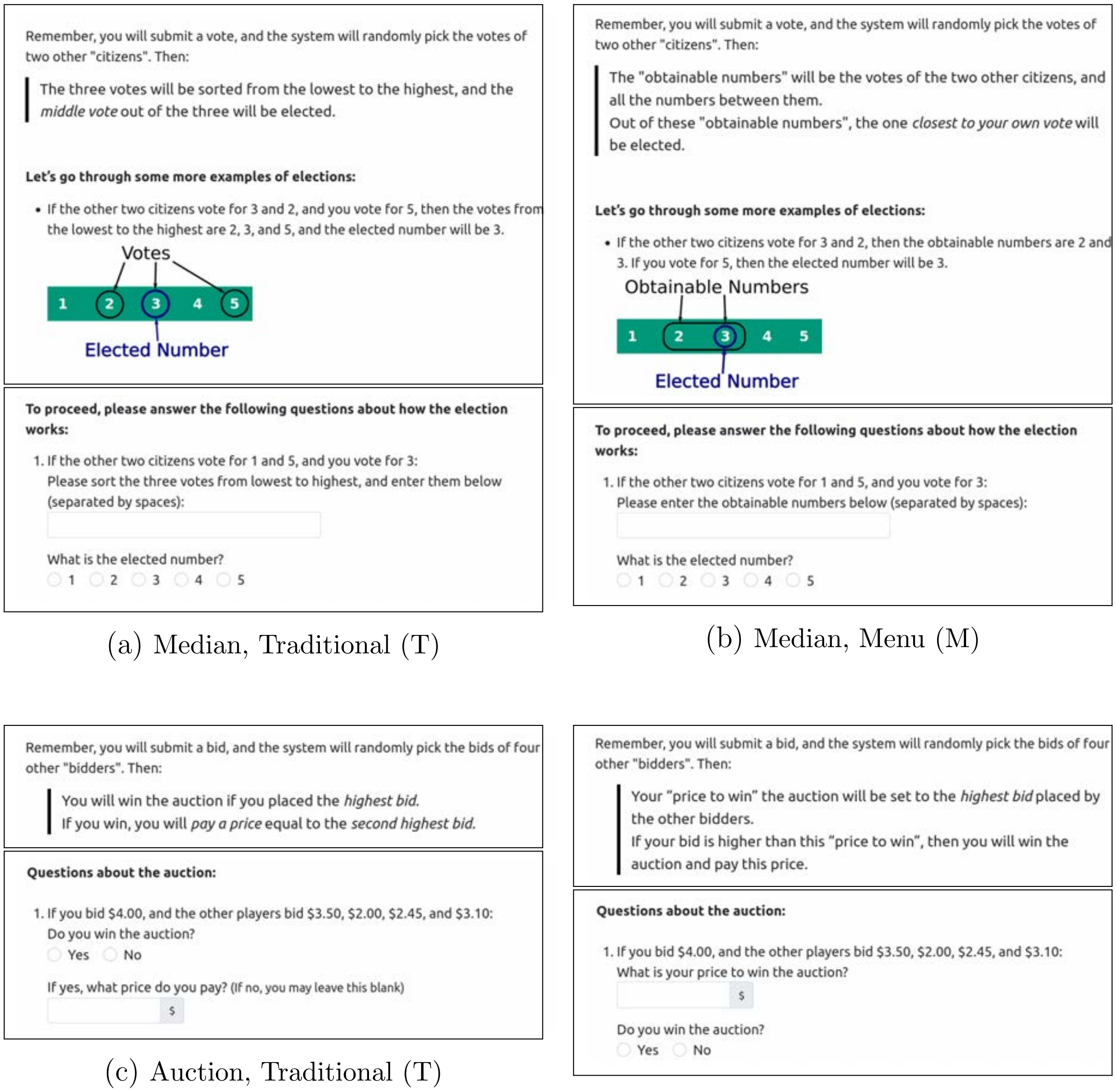
Recall that as Table 7 shows, completing all comprehension questions on the first try is associated with a sharp increase in the rate of straightforward behavior under M but not under T. On one hand, in retrospect, this may result from the comprehension questions being more difficult in M, as they may more accurately classify the participants according to their numeracy or how well they understand the mechanism. (For this reason, we do not investigate the correlation between comprehension and straightforwardness across treatments, but only within T or within M.) On the other hand, it may reinforce the natural notion that menu descriptions are only worth using when participants can understand them well.

Dropouts. 20 participants dropped out of the experiment after the informed consent and introductory screens: 16 during Median, and 4 during Auction. They are not included in our analysis. Including the 4 Auction dropouts could not effect our findings and conclusions (for either Auction or Median) more than trivially. While including the 16 Median dropouts—5 in T and 11 in M—*could* affect our findings, under reasonable assumptions they would not affect our conclusions much.[3] For example, if all dropouts in both T and M play i.i.d. uniformly randomly—yielding straightforward play in 20 percent of the rounds, and essentially never in all rounds—then

[1] Some (arbitrarily chosen) examples of mistakes in T include: not writing a repeated vote twice; sorting the numbers high-to-low; (seemingly) confusing which bubble corresponded to which elected number; writing the numbers by their listed order instead of their sorted order; or entering the elected number in the blank for the sorted numbers.

[2] Most of the relevant mistakes which occurred in T also sometimes occurred in M. Some of the most common mistakes specific to M included: thinking that a number is obtainable if it is between *any* pair of votes (instead of between the other citizens' votes); thinking the obtainable numbers *couldn't* include the votes of the other two citizens; or thinking the obtainable numbers were *only* the votes of the other two citizens.

[3] Of the 16, 5 dropped on the first screen describing the mechanism (1 in T, 4 in M), 1 dropped on the practice round (in T), and 10 dropped on the comprehension screen (3 in T, 7 in M).



Remember, you will submit a vote, and the system will randomly pick the votes of two other "citizens". Then:

> The three votes will be sorted from the lowest to the highest, and the *middle vote* out of the three will be elected.

Let's go through some more examples of elections:

- If the other two citizens vote for 3 and 2, and you vote for 5, then the votes from the lowest to the highest are 2, 3, and 5, and the elected number will be 3.

To proceed, please answer the following questions about how the election works:

1. If the other two citizens vote for 1 and 5, and you vote for 3:
   Please sort the three votes from lowest to highest, and enter them below (separated by spaces):

   What is the elected number?
   ○ 1 ○ 2 ○ 3 ○ 4 ○ 5

(a) Median, Traditional (T)

Remember, you will submit a vote, and the system will randomly pick the votes of two other "citizens". Then:

> The "obtainable numbers" will be the votes of the two other citizens, and all the numbers between them.
> Out of these "obtainable numbers", the one *closest to your own vote* will be elected.

Let's go through some more examples of elections:

- If the other two citizens vote for 3 and 2, then the obtainable numbers are 2 and 3. If you vote for 5, then the elected number will be 3.

To proceed, please answer the following questions about how the election works:

1. If the other two citizens vote for 1 and 5, and you vote for 3:
   Please enter the obtainable numbers below (separated by spaces):

   What is the elected number?
   ○ 1 ○ 2 ○ 3 ○ 4 ○ 5

(b) Median, Menu (M)

Remember, you will submit a bid, and the system will randomly pick the bids of four other "bidders". Then:

> You will win the auction if you placed the *highest bid.*
> If you win, you will *pay a price* equal to the *second highest bid.*

Questions about the auction:

1. If you bid $4.00, and the other players bid $3.50, $2.00, $2.45, and $3.10:
   Do you win the auction?
   ○ Yes ○ No

   If yes, what price do you pay? (If no, you may leave this blank)
   $

(c) Auction, Traditional (T)

Remember, you will submit a bid, and the system will randomly pick the bids of four other "bidders". Then:

> Your "price to win" the auction will be set to the *highest bid* placed by the other bidders.
> If your bid is higher than this "price to win", then you will win the auction and pay this price.

Questions about the auction:

1. If you bid $4.00, and the other players bid $3.50, $2.00, $2.45, and $3.10:
   What is your price to win the auction?
   $

   Do you win the auction?
   ○ Yes ○ No

(d) Auction, Menu (M)

Figure A.1: Excerpts of the comprehension pages, by mechanism and treatment.

Notes: Each comprehension page starts with a reminder of the mechanism description, followed by several examples, then comprehension questions. See the online experimental materials.

the "% Straightforward" row in Table 6 would be 67 in T and 74 in M, p-value = 0.11; and the "% All Straightforward" row would be 25 and 47, $p = 0.001$. Under the potentially more plausible assumption that the dropouts play similarly to participants who had some comprehension-question mistakes—after all, a large fraction of the dropouts occur on the comprehension screen—our findings are still less affected. In particular, if dropouts in T and M, respectively, are assumed to each play straightforward in 67 and 65 percent of rounds (see Table 7's "Some Mistakes" column), then the "% Straightforward" row in Table 6 would be 69 in T and 79 in M, $p = 0.01$; if dropouts are assumed to play all rounds straightforward 33 and 34 percent of the time (matching the data for "% All Straightforward" among those who made comprehension mistakes), then the "% All Straightforward" row (after rounding to integer numbers of participants) would be 30 in T and 50 in M, $p = 0.004$. Indeed, it would take an extreme—and extremely unrealistic—assumption to make the "% Straightforward" row (almost) identical across the treatments: if dropouts in T and M would have played straightforwardly, respectively, 100 and 0 percent of the time, then the "% Straightforward" row would be 71 in T and 72 in M. However, even under this unlikely worst-case assumption, the "% All Straightforward" row would still be very different across the treatments: 30 in T and 47 in M, $p = 0.01$.

A.2 Auction

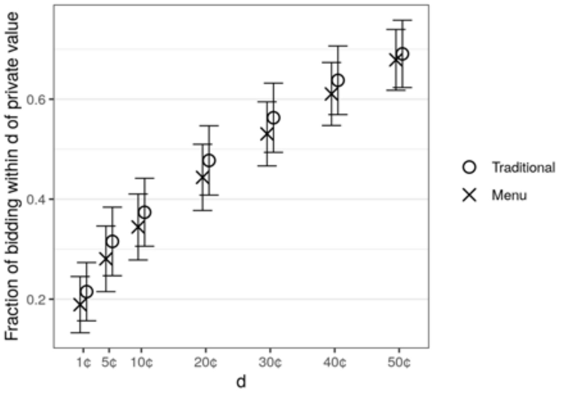
Overall, Auction appears more difficult for the participants than Median. Few bids follow the exact dominant strategy of submitting a bid equal to one's private value (or, due to tie-breaking, bidding one cent more). Indeed, only 21 and 19 percent of bids, respectively in T and M, are within 1 cent of the private value; and only 7 and 1 percent of participants, respectively, bid within 1 cent in every round.

On the other hand, the number of possible strategies is far larger in Auction than in Median, as any bid (of a whole number of cents) between \$0.00 and \$5.00 is possible. Taking a "range of strategies" consisting of 1/5 of all possible bids—the same overall fraction of possible strategies that constitute straightforward play in Median—yields results much more in line with Median. Namely, setting d (the distance from the private value which is considered "straightforward play") to \$0.50 results in 69 and 67 percent straightforward in T and M.

It is not clear which value of d should be interpreted as playing the (approximate) dominant strategy. However, Table A.4 indicates that many reasonable values of d produce the (non-)result of no difference across T and M. Other (perhaps more standard) measures of how far participants deviate from “bidding their value” also yield non-results. For example, using mean absolute deviation (MAD), as in Li (2017) and prior work, yields \$0.51 and \$0.55 in T and M, $p = 0.58$. In all of our results reporting straightforward behavior, we exclude all rounds where the private value is within d of the maximum or minimum possible bid (because in these rounds, not all possible bids within plus or minus d of the private value are possible). Including these rounds has essentially no impact on our (non-)results.

Table A.4: Straightforward rates by d.

d (\$)	Trad. (T) ($N = 100$)	Menu (M) ($N = 100$)	p
0.01	0.21 (0.03)	0.19 (0.03)	0.53
0.05	0.32 (0.04)	0.28 (0.03)	0.47
0.10	0.37 (0.04)	0.34 (0.03)	0.55
0.20	0.48 (0.04)	0.44 (0.03)	0.50
0.30	0.56 (0.04)	0.53 (0.03)	0.50
0.40	0.64 (0.03)	0.61 (0.03)	0.56
0.50	0.69 (0.03)	0.67 (0.03)	0.79



Notes: d (\$): the distance in dollars from the private value that bidding within which is considered straightforward play. Standard errors are reported in parentheses. p: two-sided, Welch’s test, weighted by the number of rounds such that all bids within d of the private value are possible. In the plot, capped bars show 95% confidence intervals.

Table A.5 considers how straightforward play relates to performance on the comprehension questions in Auction. We see that participants who make no mistakes in the comprehension question play straightforwardly at higher rates. However, we see no difference in this trend between the Traditional and Menu treatments.

We close this appendix section with some additional, speculative thoughts inspired by further analysis of the data. Despite our experimental results, we believe that menu descriptions make strategyproofness mathematically more apparent even for a second price auction. In our case, this hypothesized mathematical ease did not induce experimental results. While we do not know why, the comprehension questions may suggest that some participants were simply confused by our menu description. In particular, one of the main misconceptions we hoped that menu descriptions could help dispel is that *your price to pay if winning the auction* = *your bid.* Indeed, a defining property of menu descriptions is that one cannot influence the price one will pay. Our menu description did not achieve this goal. Table A.6 shows that 8 and 18 participants, respectively in T and M, submitted answers to comprehension questions where they

Table A.5: Straightforward rates by comprehension mistakes and d.

d (\$)		No Mistakes	Some Mistakes	p
0.05	T	0.33	0.25	0.33
	M	0.31	0.23	0.23
	T+M	0.32	0.24	0.11
0.20	T	0.50	0.39	0.18
	M	0.47	0.39	0.07
	T+M	0.49	0.39	0.05
0.50	T	0.73	0.59	0.10
	M	0.70	0.63	0.06
	T+M	0.72	0.61	0.04

Notes: 37 of 100 participants make mistakes in T; 41 of 100 in M. The p-values are two sided (Welch's t-test, weighted by the number of rounds such that all bids within d of the private value are possible).

Table A.6: Number of participants making different categories of comprehension mistakes in Auction.

	T	M	p
Qs 1, 2, or 3	24	34	
Qs 1 or 3	23	30	0.34
Qs 1, 2, or 3, price = bid	8	18	
Qs 1 or 3, price = bid	8	18	0.06
Qs 4, 5, or 6	21	20	1.00
Any question	37	41	

Notes: Each row concerns a subset of the questions ("Qs") in which the participant may have mistakes. Questions 1, 2, and 3 concerned how the auction worked. Of these, questions 1 and 3 asked for precisely the same information in both treatments (in question 2, some information was optional in T). Questions 4, 5, and 6 concerned how the result of the auction causes the participant to gain or lose money (and these were precisely the same in both treatments). The "price = bid" mistake indicates that (in some relevant question) the participant wrote their own bid as their price. In rows where both treatments ask for the same information, a p-value is reported (two-sided test for equality of proportions).

wrote their own bid as their price ($p = 0.06$). It may be that the added complexity (in terms of word length and overall conceptual difficulty) in M overshadowed any emphasis that the menu description was intended to put on how the price was calculated.[4] Future work may refine the menu descriptions we used, or (like Breitmoser and Schweighofer-Kodritsch (2022)) explore alternative approaches to framing auctions.

B Additional Preliminaries

B.1 Environments

First, we formally define an *environment.*

Definition B.1. An *environment* with n players (or agents) consists of a set A of outcomes, and sets $\mathcal{T}_1, \ldots, \mathcal{T}_n$ of types of the n players. For *ordinal environments*, each $t_i \in \mathcal{T}_i$ induces a weak order[5] $\succeq_i^{t_i}$ over the outcomes in A. For *cardinal environments*, each t_i induces a utility function $u_i(t_i, \cdot) : A \to \mathbb{R}$ (in these environments, we additionally define $a \succeq_i^{t_i} b$ to mean $u_i(t_i, a) \geq u_i(t_i, b)$).[6]

We remark that all of the mechanisms we consider are *direct revelation*, in that the players are simultaneously asked to report their types to the mechanism. For a strategyproof social choice function, this direct mechanism is dominant-strategy incentive compatible (i.e., truthful reporting is always a dominant strategy). Thus, we sometimes do not notationally distinguish between the social choice function and the corresponding (direct) mechanism.

Now, we define the i-relevant outcome sets, which partition the outcomes according to "what i cares about."

Definition B.2. We write $a \sim_i b$ when both $a \succeq_i^{t_i} b$ and $b \succeq_i^{t_i} a$ for all $t_i \in \mathcal{T}_i$. That is, $a \sim_i b$ when all types of player i are indifferent between a and b. The *i-relevant outcome sets* (or *i-outcomes*) in some environment are the equivalence classes of A under $\sim_i$, that is, the partition of A given by $\big\{\{b \in A : a \sim_i b\} \bigm| a \in A\big\}$. We let $[a]_i = \{b \in A : a \sim_i b\}$, and denote the collection of i-outcomes by A_i.[7]

[4] This could also stem from fine-grained details of the exact wording of the descriptions. For example, in the menu description, if the participants stop reading the first sentence of the description early, they may come away with the understanding that "Your 'price to win' the auction will be set to the *highest bid*" (in which case, the mistaken price = bid answer would be correct for questions 1 and 3).

[5] A weak order $\succeq$ is a binary relation that is reflexive (i.e. $a \succeq a$ for all $a \in A$), transitive (i.e. if $a \succeq b$ and $b \succeq c$, then $a \succeq c$), and total (i.e. for all $a, b \in A$, we have either $a \succeq b$ or $b \succeq a$).

[6] When no confusion can arise, we also write $\succeq^{t_i}$ in place of $\succeq_i^{t_i}$. We also write $a \succ^{t_i} b$ when $a \succeq^{t_i} b$, but we do not have $b \succeq^{t_i} a$.

[7] When no confusion can arise, we sometimes do not distinguish between an outcome a and

B.2 Matching mechanisms

Our primary domain of interest is (ordinal) *matching mechanisms*, specifically the three canonical strategyproof mechanisms of Serial Dictatorship, Top Trading Cycles, and Deferred Acceptance. Each of these mechanisms is common in practice, and each has its own advantages and disadvantages.

Definition B.3. A *matching environment* is one in which each outcome is some partial matching (i.e., one-to-one pairing that may leave some players unmatched) of players (also referred to as *applicants*) to *institutions.* A type is some strict ordering over some subset of the institutions (the subset that the type views as "acceptable"). We refer to a tuple of types $(t_1, \ldots, t_n)$ for each applicant as a "preference profile."

Note that for each player i, the i-outcome that corresponds to a matching is completely determined by i's partner in that matching. That is, each i-outcome in a matching environment consists of all matchings μ such that $\mu(i) = h$ for some institution h, and all such matchings are indistinguishable from the point of view of i's preferences. When no confusion can arise, we identify each i-outcome in a matching environment with the institution to which applicant i matches in that i-outcome.

Note that we assume that the preferences of applicants are strict, i.e., that no applicant type is indifferent between two institutions that it finds acceptable. All of our results apply equally well in environments with and without "outside options," i.e. regardless of whether applicants rank all institutions or just some subset (and regardless of whether the outcome is always a perfect matching or might only match a subset of applicants). Note that we do not treat the institutions as strategic in any matching environment.

Serial Dictatorship (SD, Definition 2.2) is a strategyproof mechanism that always produces a Pareto-optimal[8] matching, and seems simple to understand and play according to intuitive notions. In practice, the priority order π is often selected (from all $n!$ possibilities) uniformly at random; this mechanism is called Random Serial Dictatorship (RSD). SD may be a reasonable choice of mechanism when all institutions have the same priority order over applicants, and RSD may be a reasonable choice of mechanism when applicants do not have meaningful priorities at the institutions.

the corresponding i-outcome $[a]_i$. We also let $\succeq_i^{t_i}$ denote the natural partial order over i-outcomes induced by the preference relation over outcomes. For example, in cases such as matching where types (of the players) are most naturally described as linear orders (i.e., preference lists) over the institutions, we simply let the type of player i be denoted by $\succ_i$, and write $a \succ_i b$ if i prefers institution a to institution b.

[8] A matching μ is Pareto-optimal for the applicants if there is no other matching μ' such that $\mu'(a) \succeq^{t_i} \mu(a)$ for all applicants a, and $\mu'(a) \succ^{t_i} \mu(a)$ for some applicant a.

Top Trading Cycles (TTC, Definition 2.3) is also a strategyproof mechanism that always produces a Pareto-optimal matching. The intuitive rationale behind the TTC mechanism is as follows: If an applicant a has top priority at some institution h, then that applicant "has a right" to attend h, and a will never match below h in her preference list. However, the applicant may trade away her right to match to h. For example, if another applicant b has top priority at institution h', and a's favorite institution is h', but b's favorite institution is h, then a and b may trade their priorities at h and h'. This trade exactly occurs as a two-applicant (and two-institution) cycle in TTC. Longer cycles are more elaborate priority trading cycles. Moreover, after the applicants in such a cycle are matched, further trades can be made in the sub-market resulting from removing these applicants and institutions, and this process continues until all applicants are matched.

This rationale can be formalized into a number of remarkable properties of TTC. First, since preferences are strict in our model, the TTC outcome is independent of the order in which cycles are chosen to match along:

Lemma B.4 (Follows from Shapley and Scarf, 1974; Roth and Postlewaite, 1977)**.** *The TTC algorithm defines a unique social choice function for each set of (strict) priority orders* $\{\succ_p\}_p$.

Second, TTC produces a Pareto-optimal outcome in which each applicant is matched to an institution at least as good as any institution in which she has top priority.

The most important mechanism we consider is Deferred Acceptance (DA, Definition 2.4). Note that we use DA to refer to applicant-proposing DA. One can also consider *institution* proposing DA (simply interchanging the two sides, and hence the rolls of preferences and priorities), to which we refer as IPDA. (And when we need to clarify, we use APDA to refer to the applicant-proposing version.) When we need to refer specifically to the *algorithm* traditionally used to compute DA (the so-called "Gale–Shapley" algorithm), we call it the "DA algorithm" or similar.

The central objective of DA is *stability* of the resulting matching. A matching μ is unstable if there exists an (unmatched) pair a, h of an applicant and an institution such that $h \succ_a \mu(a)$ and $a \succ_h \mu(h)$. A matching is stable if it is not unstable. Intuitively, the DA process starts from matching each applicant to her favorite institution, then performs the minimal amount of adjustments needed to ensure stability. For example, if every applicant has a distinct favorite institution, then in the first round of DA they will all propose to their favorite institutions, and exactly this matching will be output. However, if two applicants a_1 and a_2 propose to the same institution h, then in order to preserve stability, we can only tentatively assign to h whichever applicant has higher priority at h. This logic is repeated until every applicant finds some tentative match, which is then made final.

The matching DA produces is always stable. Moreover, it produces the applicant-optimal (for all applicants simultaneously) stable matching. (Thus, DA produces a unique outcome that is independent of the order in which applicants are chosen to propose.) Finally, DA is the unique stable matching mechanism that is strategyproof for the applicants. (Moreover, no stable mechanism can be strategyproof for both applicants and institutions.)

Note that, at a technical level, we use SD, TTC, and DA to refer to the corresponding *social choice function* (or to the corresponding direct-revelation mechanism), not any particular description or algorithm used to calculate it. When we need to refer to a specific algorithmic representation, we use words such as "the traditional DA algorithm".

B.3 Auctions

Our secondary domain of theoretical interest is *(combinatorial) auction* environments. These are cardinal-preference environments in which some set of items is to be offered for sale to some set of bidders.

Definition B.5. An *auction environment* for n players (also referred to as *bidders*) and a set of *items* M is one in which an outcome is an allocation $(A_1, \ldots, A_n)$ over M (i.e., $A_i \subseteq M$ for every i, and $A_i \cap A_j = \emptyset$ for every $i \neq j$), along with prices $(p_1, \ldots, p_n)$ to be paid by the respective bidders. The set of *types* $\mathcal{T}_1, \ldots, \mathcal{T}_n$ are (arbitrary) sets of valuation functions $v_i : 2^M \to \mathbb{R}_{\geq 0}$, which map subsets of M to a valuation for that set of items. The utility of player i under an outcome with allocation $(A_1, \ldots, A_n)$ and prices $(p_1, \ldots, p_n)$ is $v_i(A_i) - p_i$.

Note that the i-outcomes are completely determined by the items allocated to i, along with the price i pays. That is, all outcomes in which i receives the same set of items and pays the same price are indistinguishable from the point of view of bidder i.

In all of our different auction environments, we consider the allocation function that maximizes *welfare*, i.e., maximizes the sum $\sum_i v_i(A_i)$ over all possible allocations $(A_1, \ldots, A_n)$. Furthermore, we consider the price function that charges the canonical VCG prices, i.e., charges bidder i the externality she exerts on the welfare of the other bidders (i.e., the loss in welfare that the other bidders incur due to the presence of bidder i). That is, when the welfare-maximizing allocation is $(A_1, \dots, A_n)$, player i is charged

$$p_i(v_1, \dots, v_n) = \left(\max_{A'_{-i}} \sum_{j \neq i} v_j(A'_j) \right) - \sum_{j \neq i} v_j(A_j)$$

(where the max is taken over all allocations A'_{-i} to bidders other than i). The social choice function defined by these allocation and price functions is strategyproof.

For concreteness, we restrict attention to valuation functions (and thus prices) that are integers in $\{0, 1, \dots, K\}$, for some K that may be a function of m and/or n. (Note though that the scale of these utility values is arbitrary, so 1 unit can represent a single cent or even a smaller unit of currency.)

In each of the two auction environments on which we focus, each type set $\mathcal{T}_i$ is defined as the set of all valuation functions with some well-motivated properties. All valuations we discuss will be monotone (i.e., for any $S \subseteq T$, we have $v_i(S) \leq v_i(T)$) and normalized (so that $v_i(\emptyset) = 0$).

Definition B.6. An *additive valuation* $v_i : 2^M \to \mathbb{R}_{\geq 0}$ is a valuation such that[9] $v_i(S) = \sum_{j \in S} v_i(j)$ for each $S \subseteq M$. We denote the social choice function consisting of the welfare-maximizing allocation, along with the VCG prices, with n additive bidders and m items as $AD = AD^{m,n}$.

Definition B.7. A *unit demand valuation* $v_i : 2^M \to \mathbb{R}_{\geq 0}$ is a valuation such that $v_i(S) = \max_{j \in S} v_i(j)$ for each $S \subseteq M$. We denote the social choice function consisting of the welfare-maximizing allocation, along with the VCG prices, with n unit demand bidders and m items as $UD = UD^{m,n}$.

B.4 Notation

Following standard computer science notation, we write $f(n) = \widetilde{\mathcal{O}}\big(g(n)\big)$ (resp. $\widetilde{\Omega}\big(g(n)\big)$) when there exists an integer k such that $f(n) = \mathcal{O}\big(g(n) \log^k g(n)\big)$ (resp. $\Omega\big(g(n) \log^k g(n)\big)$). That is, $\widetilde{\mathcal{O}}$ and $\widetilde{\Omega}$ ignore logarithmic factors. In matching environments, we often use variables like d_i (mnemonic: doctor) to refer to applicants and h_i (mnemonic: hospital) to refer to institutions.

[9] We slightly abuse notation by writing $v_i(j)$ for $v_i(\{j\})$.

C Omitted Proofs

For the reader's convenience, throughout the appendix, we restate each result before giving the proof. We start by proving Theorem 3.1 without assuming the strategyproofness of DA, thus providing an alternative didactic approach for proving its strategyproofness. For completeness, known results used in the proof are stated with full proofs in Appendix E.

Theorem 3.1. *Description 1 is a menu description of DA. In particular, if every applicant is assigned to an institution according to this description, then the result is the applicant-optimal stable matching (i.e., the matching output by applicant-proposing DA).*

Proof. Fix an applicant d_*. Let P be a preference profile, and let $h_* = APDA_{d_*}(P)$ denote the match of d_* according to applicant-proposing DA. We wish to show that h_* is the P_i-favorite institution in the set containing (1) the "outside option" of going unmatched, and (2) all institutions h such that h prefers d_* to $IPDA_h(P_{-d_*})$ (the match of h according to institution-proposing DA in the market without d_*).

Let $P|_{d_*:\emptyset}$ denote the preference profile obtained by altering P so that d_* reports an empty preference list (i.e., marking all institutions as unacceptable). Note that $IPDA(P_{-d_*})$ and $IPDA(P|_{d_*:\emptyset})$ produce the same matching (ignoring d_*), and furthermore, the institutions h that prefer d_* to $IPDA_h(P_{-d_*})$ are exactly those that propose to d_* during (the calculation of) $IPDA(P|_{d_*:\emptyset})$. We therefore wish to prove:

1. If $h_* \neq \emptyset$, then then h_* proposes to d_* during $IPDA(P|_{d_*:\emptyset})$.
2. d_* gets no proposal in $IPDA(P|_{d_*:\emptyset})$ that is P_i-preferred to h_*.

We start with the first claim. Assume that $h_* \neq \emptyset$. Let $P|_{d_*:\{h_*\}}$ denote the preference profile obtained by altering P so that d_* reports a preference list consisting only of h_* (i.e., marking all other institutions as unacceptable). Observe that $APDA(P)$, the applicant-proposing DA outcome for preferences P, is stable under preferences $P|_{d_*:\{h_*\}}$. Thus, by the Lone Wolf / Rural Hospitals Theorem (Roth, 1986, see Theorem E.6), since d_* is matched in $APDA(P)$, she must be matched in $IPDA(P|_{d_*:\{h_*\}})$ as well. Thus, $IPDA(P|_{d_*:\{h_*\}}) = h_*$. Since regardless of the order in which we choose to make proposals in DA, the same proposals are made and the same outcome is reached (Dubins and Freedman, 1981, see Corollary E.3), the following is a valid run of $IPDA(P|_{d_*:\emptyset})$: first run $IPDA(P|_{d_*:\{h_*\}})$, then have d_* reject h_*, then continue running (according to $P|_{d_*:\emptyset}$) until IPDA concludes. Thus, h_* proposes to

d_* during $IPDA(P|_{d_*:\emptyset})$, proving the first claim.

We move on to the second claim. Let T denote d_*'s preference list, truncated just above h^* (i.e., obtained by altering d_*'s list in P by removing any institution she does not strictly prefer to h_*). Let $P|_{d_*:T}$ denote the preference profile replacing d_*'s preference list in P by the truncated list T. To prove the second claim, it suffices to prove that d_* is not matched in $IPDA(P|_{d_*:T})$. To see why this suffices, note that if this is the case, then d_* rejects all proposals made to it during $IPDA(P|_{d_*:T})$, and hence this run also constitutes also a valid run of $IPDA(P|_{d_*:\emptyset})$, and since d_* gets no proposals P_i-preferred to h_* in the former, neither does it receive such proposals in the latter, proving the second claim. It therefore remains to prove that d_* is not matched in $IPDA(P|_{d_*:T})$.

Suppose for contradiction that d_* is matched in $\mu' = IPDA(P|_{d_*:T})$. Since DA always results in a stable matching under the reported preferences (Gale and Shapley, 1962, see Lemma E.1), μ' is stable for $P|_{d_*:T}$. But by the fact that APDA results in the applicant-optimal stable matching (Gale and Shapley, 1962, see Corollary E.3), and since d_* prefers her match in μ' to her match in $APDA(P)$, μ' is not stable for P. Therefore, there is a blocking pair for μ' under P. Since such a pair must not block under $P|_{d_*:T}$ (since μ' is stable under these preferences), it must involve applicant d_*, as her preference order is the only one that differs between P and $P|_{d_*:T}$. Let (d_*, h) be this blocking pair. Therefore, $h \succ^P_{d_*} \mu'(d)$. But $\mu'(d)$ is still on d_*'s truncated list T (used in $P|_{d_*:T}$), and thus h is on this list as well. Thus, this pair blocks for μ' under $P|_{d_*:T}$ as well, and so μ' is unstable for $P|_{d_*:T}$, a contradiction. □

Remark C.1. As noted in Section 3, Theorem 3.1 extends to many-to-one markets with substitutable priorities. To quickly see why this extension holds in the special case in which institutions have responsive preferences (i.e., the special case in which each institution has a master preference order and a capacity), fix a many-to-one market, and following a standard approach, consider a one-to-one market where each institution from the original market is split into "independent copies." That is, the number of copies of each institution equals the capacity of the institution, each "copied" institution has the same preference list as the original institution, and each applicant ranks all the copies of the institution (in any order) in the same way she ranked the original institution. Ignoring the artificial difference between copies of the same institution, the run of applicant-proposing DA is equivalent under these two markets. Thus, an applicant's menu is equivalent under both markets, and so by Theorem 3.1, a menu description for the many-to-one market can be given through institution-proposing DA under the corresponding one-to-one market, which in turn is equivalent to institution-proposing DA under the original market (where at each step, each institution proposes to a number of applicants up to its capacity). The only change in Description 1 in this case would be replacing the condition $d \succ_h \mu_{-d}(h)$ with $\exists d' \in \mu_{-d}(h) : d \succ_h d'$.

Remark C.2. As additionally noted in Section 3, Theorem 3.1 also extends to many-to-one markets with contracts in which the institutions have substitutable preferences that satisfy the law of aggregate demand (the conditions under which Hatfield and Milgrom (2005) prove that the strategyproofness of applicant-proposing DA and the rural hospitals theorem hold), as shown in Description A.1. This new description generalizes Description 1 as follows: (1) Description A.1 uses the generalized Gale–Shapley algorithm of Hatfield and Milgrom (2005) starting from $(\emptyset, X)$ (where X is the set of all possible contracts) to calculate the institution-optimal stable outcome without d_* to get a matching μ_{-d_*}. (2) A given contract $c = (d_*, h, c)$ (i.e., an $(\text{applicant}, \text{institution}, \text{term})$ tuple) is on d_*'s menu if and only if h would choose (d_*, h, c) if given a choice from the set containing (d_*, c) and its matches in μ_{-d_*} (in the notation of Hatfield and Milgrom (2005), $c \in C_h(\mu_{-d_*}(h) \cup \{c\})$). Under this modification, each step of the proof of Theorem 3.1 in Section 3 holds by a completely analogous argument for this market.

Description A.1 A menu description of the applicant-optimal stable matching in a many-to-one market with contracts

(1) Calculate the institution-optimal stable matching with applicant d removed from the market using the generalized Gale–Shapley algorithm of Hatfield and Milgrom (2005). Call the resulting matching μ_{-d}. Let M be the set of contracts $c = (d, h, t)$ involving applicant d such that $c \in C_h(\mu_{-d}(h) \cup \{c\})$.

(2) Match d to d's highest-ranked contract in M.

Remark C.3. In this remark, we show how Theorem 3.1, which characterizes the menu in DA in terms of Description 1, can be used to prove results from Ashlagi et al. (2017) via arguments similar to Cai and Thomas (2022). Consider a randomized market with $n+1$ applicants and n institutions, where such that each applicant/institution draws a full-length preference list uniformly at random, and let μ be the result of (applicant-optimal) DA with these preferences. We prove that the expected rank each applicant receives on their preference list (formally, the expectation of $|\{h : h \succeq_d \mu(d)\}|$ for any d) is at least $(1 - \epsilon)n/\log(n)$ for any $\epsilon > 0$ and large enough n.

Fix an applicant d_*, and consider calculating d_*'s menu using Description 1 in this market. This is equivalent to considering IPDA in a market where d_* rejects all proposals, and setting d_*'s menu to consist of all proposals she receives. By the principle of deferred decisions, this run of IPDA can be constructed by letting each institution h proposes to a uniformly random applicant (among those h has not yet proposed to) each time she proposes. Observe that this run of IPDA will terminate as soon as each of the n applicants other than d_* receives a proposal. Thus (much like the standard case of n applicants and n institutions in APDA Wilson (1972)), the total number of proposals made in this run of IPDA is stochastically dominated by a coupon collector random variable. Thus, intuitively, the total number of proposals will be $n \log(n)$, and $\log(n)$ of these will go to d_* in expectation, and d_*'s top choice out of these $\log(n)$ proposals will be their $n/\log(n)$th ranked choice overall.

Formally, let Y denote the number of proposals d_* receives, and let $\overline{Y}$ denote the same quantity in a market where each institution makes each proposal completely uniformly at random (without regard to prior proposals); it follows that Y is stochastically dominated by $\overline{Y}$. Let $\overline{Z_i}$ denote the total number of proposals between the $(i-1)$th and ith distinct applicant in $\mathcal{D} \setminus \{d_*\}$ receiving a proposal (in the market with repeated proposals). The expected value of Z_i is exactly $(n+1)/(n+1-i)$, and each of these Z_i proposals (except for the final one) has a $1/i$ probability of going to d_*. Thus, we have

$$\mathbb{E}[Y] \le \mathbb{E}\left[\overline{Y}\right] = \sum_{i=1}^{n} \frac{1}{i}\left(\frac{n+1}{n+1-i} - 1\right) = \sum_{i=1}^{n} \frac{1}{i}\left(\frac{i}{n+1-i}\right) = H_n \le \log(n) + 1.$$

Now, let $R = |\{h : h \succeq_d h_*\}|$, where h_* is d_*'s top-ranked proposal received (i.e., d_*'s match in APDA). One can show that, conditioned on $Y = y$, we have the expected value of R exactly equal to $(n+1)/(y+1)$ (see for example (Cai and Thomas, 2022, Claim A.1)). Thus, by Jensen's inequality, we have

$$\mathbb{E}[R] = \mathop{\mathbb{E}}_{y \sim Y}\left[\frac{n+1}{y+1}\right] \ge \frac{n+1}{\mathbb{E}[Y]+1} \ge \frac{n+1}{\log(n)+2} \ge \big(1-\epsilon\big)\frac{n}{\log(n)}$$

for any $\epsilon > 0$ and large enough n, as desired.

Theorem 7.2. *If there are at least three applicants and three institutions, then for every applicant i there exist priorities of the institutions such that any applicant-proposing menu description of DA for applicant i is non-local.*

Proof. Assume for contradiction that D is a local applicant-proposing menu description for some applicant d_* in a market with applicants d_1, d_2, d_* and institutions h_1, h_2, h_3. We first define priorities of three institutions as follows:

$$\begin{aligned} &h_1 : d_2 \succ d_* \succ d_1 \\ &h_2 : d_1 \succ d_* \succ d_2 \\ &h_3 : (\text{any list ranking } d_1, d_2, \text{ and } d_*) \end{aligned}$$

Next, we consider two possible preference lists for each of d_1, d_2:

$$\begin{aligned} &\succ_1 : h_1 \succ h_2 \succ h_3 \qquad &&\succ_1' : h_1 \succ h_3 \succ h_2 \\ &\succ_2 : h_2 \succ h_1 \succ h_3 \qquad &&\succ_2' : h_2 \succ h_3 \succ h_1 \end{aligned}$$

Consider executing D when preferences are $P = (\succ_1, \succ_2)$ (i.e. where d_1 has preference $\succ_1$ and d_2 has preference $\succ_2$). Consider the final time that the description learns the difference between $\succ_i$ and $\succ_i'$ for some $i \in \{1, 2\}$, that is, the latest node v along the execution path of P where the execution diverges from that of some $P' \in \{(\succ_1', \succ_2), (\succ_1, \succ_2')\}$. By the symmetry in the defined set of preferences, it is without loss of generality to assume that this node queries applicant d_1, and thus v has one successor node consistent with preferences $P = (\succ_1, \succ_2)$, and a different successor node consistent with preferences $P' = (\succ_1', \succ_2)$. When preferences are P, applicant d_*'s menu is $\{h_3\}$. But when preferences are P', applicant d_*'s menu is $\{h_1, h_3\}$. Now, h_1 has already been queried from both d_1 and d_2's lists in predecessor nodes of v (for d_1, this is because both $\succ_1$ and $\succ_1'$ rank h_1 first; for d_2, this is because $\succ_2$ ranks h_1 before h_3, but $\succ_2'$ does not, and at v we already know d_2's preference is not $\succ_2'$). Thus, because D is local, the label L_{h_1} (which determines whether h_1 is or is not on the menu) must be equal in v and in all successor nodes of v. This is a contradiction, because there are some successor nodes of v where h_1 is on the menu and some where h_1 is not on the menu. Thus, no local applicant-proposing menu description of DA exists. □

Theorem 7.3. *If there are at least three applicants and two institutions, then there exist preferences of the applicants such that any institution-proposing outcome description of DA is non-local.*

Proof. Assume for contradiction that D is a local institution-proposing outcome description in a market with institutions h_1, h_2 and applicants d_1, d_2, d_3. Recall that for such a description, we consider the preferences of the applicants to be fixed and consider descriptions which query the priorities of the institutions. We first define preferences of three applicants as follows:

$$\begin{aligned} &d_1 : h_2 \succ h_1 \\ &d_2 : h_1 \succ h_2 \\ &d_3 : \text{(any complete preference list)} \end{aligned}$$

Next, we consider two possible preference lists for each of h_1, h_2:

$$\begin{aligned} &\succ_1 : d_1 \succ d_2 \succ d_3 \qquad && \succ_1' : d_1 \succ d_3 \succ d_2 \\ &\succ_2 : d_2 \succ d_1 \succ d_3 && \succ_2' : d_2 \succ d_3 \succ d_1 \end{aligned}$$

Analogously to the previous proof, consider the last vertex v along the execution path with priorities $Q = (\succ_1, \succ_2)$ where the execution diverges from that of some priority profile in $\{(\succ_1', \succ_2), (\succ_1, \succ_2')\}$, and without loss of generality suppose that this vertex v has one successor consistent with Q and another consistent with $Q' = (\succ_1', \succ_2)$. In predecessors of v, the description has queried d_1 from each institution's priority list, so the label L_{d_1} (which determines the match of d_1) cannot be updated in any successor nodes of v. Under Q, the matching is $\{(h_1, d_2), (h_2, d_1)\}$, and under Q', the matching is $\{(h_1, d_1), (h_2, d_2)\}$. But by locality, D must assign d_1 to the same match in all successor nodes of v, a contradiction. Thus, no local institution-proposing outcome description of DA exists.[10] □

Theorem 8.3. *No item-linear description of an auction with unit-demand bidders exists. In fact, any item-read-once description for unit-demand bidders requires memory $\Omega(m^2)$. This holds both for outcome descriptions and for menu descriptions.*

Proof. Consider the special case where $m = n$ and every bidder has value either 0 or 1 for each item. In this case, the welfare optimal matching is simply given by the maximum size matching in the bipartite graph where edges are drawn between a bidder and an item if and only if the bidder values that item at 1. Computing this matching requires at least as much memory as the problem of checking whether a *perfect matching* (one of size n) exists. Computing the menu of an additional $(n + 1)$th bidder is also at least as hard as this problem: bidder $n + 1$ will face price 1 on every item if and only if a perfect matching exists. We show that checking whether a perfect matching exists requires memory $\Omega(n^2)$ with a item-read-once algorithm.

Let $n = 2k$, and consider any item-read-once algorithm. Consider a set of k^2

[10] This construction can also be modified to hold in a market with three applicants and three institutions by adding an institution which all applicants (including d_3) rank last.

bits $x_{i,j} \in \{0,1\}$ for $i,j \in [k]$. We adversarially build a collection of inputs, one for each bitstring $\{x_{i,j}\}_{i,j}$, as follows: Start by dividing the n bidders into two classes, $v_1, \ldots, v_k$ and $w_1, \ldots, w_k$. For each $i \in [k]$, let z_i denote the item the which is queried ith by the algorithm. Without loss of generality, the algorithm learns all bidder's values for the item z_i at once. We give these values as follows: v_i demand z_i, but no other v_j wants z_i for $j \neq i$, and for each j such that $x_{i,j} = 1$, we let w_i demand z_j.

Now, consider a pair of indices $(p,q) \in [k]$. For $i \in \{n/2+1, \ldots, n\}$, let z_i denote the ith item queried. For the $k-1$ such items with $i < n$, let each z_i be demanded by exactly one bidder in $\{w_1, \ldots, w_k\} \setminus \{w_q\}$. For $i = n$, let z_n be demanded only by the bidder v_p.

Lemma C.4. *There exists a perfect matching (of bidders to items which they demand) if and only if $x_{p,q} = 1$.*

Proof. All of the items z_i with $i \in \{n/2+1, \ldots, n\}$ are demanded by exactly one bidder, and thus must match there if there is a perfect matching. In particular, v_p must be taken by z_n and each w_j for $j \neq q$ must be taken by $z_{n/2+j}$. Each bidder z_i for $i \neq p$ has the option to match to v_i. If $x_{p,q} = 1$, then z_p can match to to w_q to complete the matching. On the other hand, if some perfect matching exists, then every bidder v_i for $i \neq p$ must be take some item, but this item must of course be b_i. Thus, z_p must receive w_q, and we must have $x_{p,q} = 1$. □

Lemma C.5. *After the first $n/2$ items are queried in the above process, the description must be in a distinct state for each distinct bitstring $\{x_{i,j}\}_{i,j}$.*

Proof. Suppose for contradiction that the mechanism was in the same state after reading inputs corresponding to $\{x_{i,j}\}_{i,j}$ and $\{x'_{i,j}\}_{i,j}$, where (without loss of generality) $x_{p,q} = 1$ and $x'_{p,q} = 0$. Consider the inputs to the second half of the bidders corresponding to (p,q). Under the inputs corresponding to x, there is a perfect matching, but under x', there is not. However, the program was in the same state after reading x and x', thus it cannot be correct for both inputs, a contradiction. □

Thus, the space required by the program is at least the space needed to store the full bitstring $\{x_{i,j}\}_{i,j \in [k]}$. This is $\Omega(k^2) = \Omega(n^2)$ bits.[11] □

[11] This theorem reduces to proving the desired lower bound for any algorithm which computes the allocation. This is unlike most of our results (which hold only for individualized dictatorships descriptions, or otherwise explore novel simplicity conditions). For this reason, it is similar to results already known in the context of streaming algorithms (for example, Assadi (2020) gives a lower bound proof for streaming algorithms which is very technically similar to this proof).

D Delicate Descriptions of Deferred Acceptance

In this section, we present additional descriptions of DA. While technically interesting, we believe these descriptions are vastly more complicated than traditional descriptions of DA, and quite impractical. For notational convenience, in this appendix, we refer to the priorities of institutions as "preferences."

D.1 Institution-proposing outcome description of DA

First, we construct an *institution*-linear outcome description of DA (i.e., a description of the *applicant-optimal* stable matching, traditionally described using *applicant*-proposing DA). Interestingly, essentially this same algorithm was used as a lemma by Ashlagi et al. (2017) (henceforth, AKL).[12]

Theorem D.1 (Adapted from Ashlagi et al., 2017)**.** *Description A.2 computes the applicant-optimal stable outcome. Moreover, Description A.2 is an institution-linear description (i.e., it is institution-proposing and $\widetilde{O}(n)$-memory).*

Proof. AKL refer to the sides of the market as "men" and "women", and define "Algorithm 2 (MOSM to WOSM)", a men-proposing algorithm for the women-optimal stable matching. Description A.2 follows the exact same order of proposals as this algorithm from AKL. The only difference apart from rewriting the algorithm in a more "pseudocode" fashion is that Description A.2 performs bookkeeping in a slightly different way—Algorithm 2 from AKL maintains *two* matchings, and their list V keeps track of only women along a rejection chain; our list V keeps track of both applicants and institutions along the rejection chain (and can thus keep track of the "difference between" the two matchings which AKL tracks).

Moreover, the algorithm is institution-proposing, by construction. Furthermore, as it runs it stores only a single matching μ, a set $\mathcal{D}_{\text{term}} \subseteq \mathcal{D}$, and the "rejection chain" V (which can contain each applicant $d \in \mathcal{D}$ *at most once*). Thus, it uses memory $\widetilde{O}(n)$. □

[12] For context, Ashlagi et al. (2017) needs such an algorithm to analyze (for a random matching market) the expected "gap" between the applicant and institution optimal stable matching. Their algorithm builds on the work of Immorlica and Mahdian (2005), and is also conceptually similar to algorithms for constructing the "rotation poset" in a stable matching instance Gusfield and Irving (1989) (see also Cai and Thomas (2019)).

Description A.2 An institution-proposing outcome description of (applicant-optimal) DA

Input: Preferences of all applicants $\mathcal{D}$ and institutions $\mathcal{H}$
Output: The result of applicant-proposing deferred acceptance

```
▷ We start from the institution-optimal outcome, and slowly "improve the match for the applicants" ◁
Let $\mu$ be the result of institution-proposing DA
Let $\mathcal{D}_{\text{term}}$ be all applicants unmatched in $\mu$ ▷ $\mathcal{D}_{\text{term}}$ is all applicants at their optimal stable partner
while $\mathcal{D}_{\text{term}} \neq \mathcal{D}$ do
    Pick any $\widehat{d} \in \mathcal{D} \setminus \mathcal{D}_{\text{term}}$, and set $d = \widehat{d}$
    Let $h = \mu(d)$ and set $V = [(d, h)]$
    while $V \neq []$ do
        Let $d \leftarrow$ NextAcceptingApplicant($\mu, h$)
        if $d = \emptyset$ or $d \in \mathcal{D}_{\text{term}}$ then
            ▷ In this case, all the applicants in $V$ have reached their optimal stable partner. ◁
            Add every applicant which currently appears in $V$ to $\mathcal{D}_{\text{term}}$
            Set $V = []$
        else if $d \neq \emptyset$ and $d$ does not already appear in $V$ then ▷ Record this in the rejection chain
            Add $(d, \mu(d))$ to the end of $V$
            Set $h \leftarrow \mu(d)$            ▷ The next proposing institution will be the "old match" of $d$.
        else if $d \neq \emptyset$ and $d$ appears in $V$ then
            ▷ A new "rejection rotation" should be written to $\mu$ ◁
            WriteRotation($\mu$, $V$, $d$, $h$)            ▷ Updates the value of $\mu$, $V$, and (possibly) $h$
Return $\mu$

function NextAcceptingApplicant($\mu$, $h$)
    repeat
        Query $h$'s preference list to get their next choice $d$
    until $d = \emptyset$ or $h \succ_d \mu(d)$
    Return d

procedure WriteRotation($\mu$, $V$, $d$, $h$)
    Let $T = (d_1, h_1), \ldots, (d_k, h_k)$ be the suffix of $V$ starting with the first occurrence of $d = d_1$
    Update $\mu$ such that $\mu(h_i) = d_{i+1}$ (for each $i = 1, \ldots, k$, with indices taken mod $k$)
    ▷ Now we fix $V$ and $h$ to reflect the new $\mu$ ◁
    Update $V$ by removing $T$ from the end of $V$
    if $V \neq \emptyset$ then
        Let $(d_0, h_0)$ denote the final entry remaining in $V$
        ▷ The next proposing institution will either $h_k$ or $h_0$, depending on which $d_1$ prefers ◁
        if $h_k \succ_{d_1} h_0$ then
            Set $h \leftarrow h_0$
        else if $h_0 \succ_{d_1} h_k$ then
            Add $(d_1, h_k)$ to the end of $V$
            Set $h \leftarrow h_1$
```

D.2 Applicant-proposing menu description of DA

In this section, we construct an applicant-linear menu description of (applicant-optimal) DA. On an intuitive level, the algorithm works as per Example 2.8, but avoiding the need to "restart many times" by using the various properties of DA and by careful bookkeeping (to intuitively "simulate all of the separate runs of the brute-force description on top of each other"). On a formal level, we describe the algorithm as a variant of Description A.2. The proof constructing this algorithm uses a bijection between one applicant's menu in DA under some preferences, and some data concerning the *institution*-optimal stable matching under a related set of preferences. Our applicant-linear algorithm is then phrased as a variation of Description A.2, which (reversing the roles of applicants and institutions from the presentation in Description A.2) is able to compute the institution-optimal matching using an applicant-proposing algorithm.

Fix an applicant d_* and set P that contains (1) the preferences of all applicants $\mathcal{D} \setminus \{d_*\}$ *other than* d_* over $\mathcal{H}$ and (2) the preferences of all institutions $\mathcal{H}$ over all applicants $\mathcal{D}$ (including d_*). We now define the "related set of preferences" mentioned above. Define the *augmented preference list* P' as follows: For each $h_i \in \mathcal{H}$, we create two additional applicants $d_i^{\text{try}}, d_i^{\text{fail}}$ and two additional institutions $h_i^{\text{try}}, h_i^{\text{fail}}$. The entire preference lists of these additional agents in P' are as follows: for each $h_i \in \mathcal{H}$:

$$\begin{aligned} d_i^{\text{try}} &: h_i^{\text{try}} \succ h_i \succ h_i^{\text{fail}} & \qquad d_i^{\text{fail}} &: h_i^{\text{fail}} \succ h_i^{\text{try}} \\ h_i^{\text{try}} &: d_i^{\text{fail}} \succ d_i^{\text{try}} & \qquad h_i^{\text{fail}} &: d_i^{\text{try}} \succ d_i^{\text{fail}} \end{aligned}$$

We need to modify the preference lists of the pre-existing institutions as well. But this modification is simple: for each $h_i \in \mathcal{H}$, replace d_* with d_i^{try}. The institution-optimal matching for this augmented set of preferences P' will encode the menu, as we need.[13]

Proposition D.2. *An institution $h_i \in \mathcal{H}$ is on d_*'s menu in APDA with preferences P if and only if in the institution-optimal stable matching with the augmented*

[13] For the reader familiar with the rotation poset of stable matchings, the intuition for this construction is the following: having h_i^{try} reject applicant d_i^{try} corresponds to d_* "trying" to get $h_i \in \mathcal{H}$, i.e., "trying to see if h_i is on their menu." If d_* would be rejected by h_i after proposing, either immediately or after some "rejection rotation," then so will d_i^{try} (because they serve the same role as d_* at h_i). So if a rotation swapping h_i^{try} and h_i^{fail} exists (e.g., in the institution optimal matching) then h_i is *not* on d_*'s menu. On the other hand, if d_* could actually permanently match to h_i, then d_i^{try} proposing to h_i will result in a rejection chain that ends at some other applicant (either exhausting their preference list or proposing to an institution in $\mathcal{H}_{\text{term}}$), which does not result in finding a rotation (or writing a new set of matches as we "work towards the institution-optimal match"). Thus, if h_i^{try} and h_i^{fail} do not swap their matches in the institution-optimal stable outcome, then h_i *is* on d_*'s menu.

preferences P', we have h_i^{try} matched to d_i^{try}.

Proof. For both directions of this proof, we use the following lemma, which is a special case of the main technical lemma in Cai and Thomas (2022):

Lemma D.3. *In P', each h_i^{try} has a unique stable partner if and only if, when h_i^{try} rejects d_i^{try} (i.e. if h_i^{try} submitted a list containing only d_i^{fail}, and all other preferences remained the same), h_i^{try} goes unmatched (say, in the applicant-optimal matching).*

Note that each h_i^{try} is matched to d_i^{try} in the applicant-optimal matching with preferences P' (and the matching among all original applicants and institutions is the same as μ_{app}).

($\Leftarrow$) By the lemma, if h_i^{try} is matched to d_i^{try} in the institution-optimal matching under P', then h_i^{try} must go unmatched when h_i^{try} rejects d_i^{try}. But, after h_i^{try}, we know d_i^{try} will propose to h_i, and some rejection chain may be started. Because d_i^{try}'s very next choice is h_i^{fail} (and proposing there would lead directly to h_i^{try} receiving a proposal from d_i^{fail}), the *only* way for h_i^{try} to remain unmatched is if d_i^{try} remains matched to h_i. But because (relative to all the original applicants) d_i^{try} is in the same place as d_* on h_i's preference list, the resulting set of rejections in P' will be precisely the same as those resulting from d_* submitting a preference list in P which contains only h_i. In particular, d_* would remain matched at h_i in P if they submitted such a list. Thus, h_i is on d_*'s menu.

($\Rightarrow$) Suppose h_i^{try} is matched to d_i^{fail} in the institution optimal matching under P'. Again, h_i^{try} must receive a proposal from d_i^{fail} when h_i^{try} rejects d_i^{try}. But this can only happen if d_i^{try} is rejected by h_i (then proposes to h_i^{fail}). But because the preferences of the original applicants in P' exactly corresponds to those in P, we know that d_* would get rejected by h_i if they proposed to them in μ_{app} under P. But then h_i cannot be on d_*'s menu. □

With this lemma in hand, we can now show that there is an applicant-linear menu description of (applicant-optimal) DA. This description is given in Description A.3.

Description A.3 An applicant-proposing menu description of DA

Input: An applicant d_* and preferences of all applicants $\mathcal{D} \setminus \{d_*\}$ and institutions $\mathcal{H}$
Output: The menu of d_* in applicant-optimal DA given these preferences

```
Simulate the flipped-side version of Description A.2 (such that applicants
  propose) on preferences $P'$ to get a matching $\mu$
**Return** the set of all institutions $h_i$ such that $h_i^{\text{try}}$ is matched to $d_i^{\text{try}}$ in $\mu$
```

Theorem D.4. *There is an applicant-proposing, $\widetilde{O}(n)$ memory menu description of (applicant-optimal) DA.*

Proof. The algorithm proceeds by simulating a run of Description A.2 on preferences P' (interchanging the role of applicants and institutions, so that applicants are proposing). This is easy to do while still maintaining the applicant-proposing and $\widetilde{O}(n)$ memory. In particular, P' adds only $O(n)$ applicants and institutions, with each d_i^{try} and d_i^{fail} making a predictable set of proposals. Moreover, the modification made to the preferences lists of the institutions $h \in \mathcal{H}$ is immaterial—when such institutions receive a proposal from d_i^{try}, the algorithm can just query their lists for d_*. □

D.3 Institution-proposing individualized dictatorship description of DA

In this section, we construct an institution-linear individualized dictatorship description of (applicant-optimal) DA.[14] Throughout this section, let $P|_{d_i:L}$ denote altering preferences P by having d_i submit list L.

Unlike our applicant-proposing menu description of DA from Section D.2, our institution-proposing individualized-dictatorship description cannot be "reduced to" another algorithm such as Description A.2. However, the algorithm is indeed a modified version of Description A.2 that "embeds" our simple institution-proposing menu algorithm Description 1 (i.e., IPDA where an applicant d_* submits an empty preference list) as the "first phase." The key difficulty the algorithm must overcome is being able to "undo one of the rejections" made in the embedded run of Description 1. Namely, the algorithm must match d_* to her top choice from her menu, and "undo" all the rejections caused by d_* rejecting her choice.[15] To facilitate this, the description

[14] For some technical intuition on why such a description might exist, consider the construction used in Theorem 6.4, and consider an individualized dictatorship for applicant i executed on these preferences. To find the menu in this construction with an applicant-proposing algorithm, all of the "top tier rotations" must be "rotated", but to find the correct final matching after learning t_i, some arbitrary subset of the rotations must be "unrolled" (leaving only the subset of rotations which t_i actually proposes to). Theorem 6.4 shows that all of this information must thus be remembered in full. Now consider a run of Description A.2 on these preferences (or on a modified form of these preferences where institutions' preference lists determine which top tier rotations propose to bottom tier rotations). Some subset of top-tier institutions will propose to applicant i. To continue on with a run of Description A.2, it suffices to undo *exactly one* of these proposals. So, if two or more top-tier rotations trigger a bottom-tier rotation, then we can be certain that the bottom-tier rotation will be rotated, and we only have to remember which bottom-tier rotations are triggered by exactly one top-tier rotation (which takes $\widetilde{O}(n)$ bits).

[15] Description A.2 is independent of the order in which proposals are made. Moreover, one can even show that d_* receives proposals from all h on her menu in Description A.2. However, this does

has d_* reject institutions that propose to d_*"as slowly as possible," and maintains a delicate $\widetilde{O}(n)$-bit data structure that allows it to undo one of d_*'s rejections.[16] The way this data structure works is involved, but one simple feature that illustrates how and why it works is the following: *exactly one* rejection from d_* will be undone, so if some event is caused by *more than one* (independent) rejection from d_*, then this event will be caused regardless of what d_* picks from the menu.

We present our algorithm in Description A.4. For notational convenience, we define a related set of preferences P_{hold} as follows: For each $h_i \in \mathcal{H}$, add a "copy of d_*" called d_i^{hold} to P_{hold}. The only acceptable institution for d_i^{hold} is h_i, and if d_* is on h_i's list, replace d_* with d_i^{hold} on h_i's list. Given what we know from Section 3, the proof that this algorithm calculates the menu is actually fairly simple:

Lemma D.5. *The set $\mathcal{H}_{\text{menu}}$ output by Description A.4 is the menu of d_* in (applicant-proposing) DA.*

Proof. Ignoring all bookkeeping, Phase 1 of this algorithm corresponds to a run of $IPDA(P|_{d_*:\emptyset})$. The only thing changed is the order in which d_* performs rejections, but DA is invariant under the order in which rejections are performed. Moreover, $\mathcal{H}_{\text{menu}}$ consists of exactly all institutions who propose to d during this process, i.e. d_*'s menu (according to Section 3). □

The correctness of the matching, on the other hand, requires an involved proof. The main difficulty surrounds the "unroll DAG" Δ, which must be able to "undo some of the rejections" caused by d_* rejecting different h. We start by giving some invariants of the state maintained by the algorithm (namely, the values of Δ, μ, P, and h):

Lemma D.6. *At any point outside of the execution of* AdjustUnrollDAG*:*

(1) P contains all nodes in Δ of the form (d, h) (where h is the "currently proposing" $h \in \mathcal{H}$).

not suffice to construct our individualized dictatorship simply by changing the order of Description A.2. The main reason is this: in Description A.2, the preferences of d_* are already known, so d_* can reject low-ranked proposals without remembering the effect that accepting their proposal might have on the matching. While the "unrolling" approach of Description A.4 is inspired by the way Description A.2 effectively "unrolls rejection chains" (by storing rejections in a list V and only writing these rejections to μ when it is sure they will not be "unrolled"), the bookkeeping of Description A.4 is far more complicated (in particular, the description maintains a DAG Δ instead of a list V).

[16] Interestingly, this "rolled back state" is *not* the result of institution-proposing DA on preferences $(P, d_i : \{h_j\})$, where h_j is d_i's favorite institution on her menu. Instead, it is a "partial state" of Description A.2 (when run on these preferences), which (informally) may perform additional "applicant-improving rotations" on top of the result, and thus we can continue running Description A.2 until we find the applicant-optimal outcome.

Description A.4 An institution-proposing individualized dictatorship description of DA

Phase 1 input: An applicant d_* and preferences of applicants $\mathcal{D} \setminus \{d_*\}$ and institutions $\mathcal{H}$
Phase 1 output: The menu $\mathcal{H}_{\text{menu}}$ presented to d_* in (applicant-proposing) DA
Phase 2 input: The preference list of applicant d_*
Phase 2 output: The result of (applicant-proposing) DA

```
▷ Phase 1: ◁
Simulate a run of $IPDA(P_{\text{hold}})$ and call the result $\mu'$
Let $\mathcal{H}_*$ be all those institutions $h_i \in \mathcal{H}$ matched to $d_i^{\text{hold}}$ in $\mu'$ ▷ These institutions "currently sit at $d_*$"
Let $\mu$ be $\mu'$, ignoring all matches of the form $(d_i^{\text{hold}}, h)$
Let $\mathcal{H}_{\text{menu}}$ be a copy of $\mathcal{H}_*$ ▷ We will grow $\mathcal{H}_{\text{menu}}$
Let $\Delta$ be an empty graph ▷ The "unroll DAG". After Phase 1, we'll "unroll a chain of rejections"
while $\mathcal{H}_* \neq \emptyset$ do
    Pick some $h \in \mathcal{H}_*$ and remove $h$ from $\mathcal{H}_*$
    Add $(d_*, h)$ to $\Delta$ as a source node
    Set $P = \{(d_*, h)\}$ ▷ This set stores the "predecessors of the next rejection"
    while $h \neq \emptyset$ do
        Let $d \leftarrow$ NextInterestedApplicant($\mu, \Delta, h$)
        AdjustUnrollDag($\mu$, $\Delta$, $P$, $d$, $h$) ▷ Updates each of these values
Return $\mathcal{H}_{\text{menu}}$
▷ Phase 2: We now additionally have access to $d_*$'s preferences ◁
Permanently match $d_*$ to their top pick $h_{\text{pick}}$ from $\mathcal{H}_{\text{menu}}$
$(\mu, \mathcal{D}_{\text{term}}) \leftarrow$ UnrollOneChain($\mu, \Delta, h_{\text{pick}}$)
Continue running the Description A.2 until its end, using this $\mu$ and $\mathcal{D}_{\text{term}}$, starting from Line 4
Return the matching resulting from Description A.2

function NextInterestedApplicant($\mu$, $\Delta$, $h$)
    repeat
        Query $h$'s preference list to get their next choice $d$
    until $d \in \{\emptyset, d_*\}$ OR ($d$ is in $\Delta$, paired with $h'$ in $\Delta$, and $h \succ_d h'$) OR ($d$ is not in $\Delta$ and $h \succ_d \mu(d)$)
    Return d

procedure UnrollOneChain($\mu$, $\Delta$, $h_{\text{pick}}$)
    Let $(d_0, h_0), (d_1, h_1), \ldots, (d_k, h_k)$ be the (unique) longest chain in $\Delta$ starting from $(d_0, h_0) = (d^*, h_{\text{pick}})$
    Set $\mu(d_i) = h_i$ for $i = 0, \ldots, k$
    Set $\mathcal{D}_{\text{term}} = \{d_*, d_1, \ldots, d_k\}$
    return $(\mu, \mathcal{D}_{\text{term}})$
```

```
procedure AdjustUnrollDag($\mu$, $\Delta$, P, $d$, $h$)
    if $d = \emptyset$ then
        Set $h = \emptyset$                                    ▷ Continue and pick a new $h$
    else if  $d = d^*$  then          ▷ $h$ proposes to $d_*$, so we've found a new $h$ in the menu
        Add $h$ to $\mathcal{H}_{\text{menu}}$
        Add $(d_*, h)$ to $\Delta$
        Add $(d_*, h)$ to the set $P$ ▷ $h$ still proposes; the next rejection will have multiple
        predecessors
    else if $d$ does not already appear in $\Delta$ then                 ▷ Here $h \succ_d \mu(d)$
        Add $(d, \mu(d))$ to $\Delta$                        ▷ Record this in the rejection DAG
        Add an edge from each $p \in P$ to $(d, \mu(d))$ in $\Delta$, and set $P = \{(d, \mu(d))\}$
        Set $h' \leftarrow \mu(d)$, then $\mu(d) \leftarrow h$, then $h \leftarrow h'$
        ▷ The next proposing institution will be the "old match" of d.                  ◁
    else if $d$ appears in $\Delta$ then
        AdjustUnrollDagCollision($\mu$, $\Delta$, $P$, $d$, $h$)      ▷ Updates each of these values

procedure AdjustUnrollDagCollision($\mu$, $\Delta$, P, $d$, $h$)
    Let $p_1 = (d_1, h_1)$ be the pair where $d = d_1$ appears in $\Delta$        ▷ We know $h \succ_{d_1} h_1$
    Let $P_1$ be the set of all predecessors of $p_1$ in $\Delta$

    ▷ First, we drop all rejections from $\Delta$ which we are now sure we won't have to unroll   ◁
    Let $(d_1, h_1), \ldots, (d_k, h_k)$ be the (unique) longest possible chain in $\Delta$ starting from $(d_1, h_1)$
            such that each node $(d_j, h_j)$ for $j > 1$ has exactly one predecessor
    Remove each $(d_i, h_i)$ from $\Delta$, for $i = 1, \ldots, k$, and remove all edges pointing to these nodes

    ▷ Now, we adjust the nodes to correctly handle $d_1$ (which might have to "unroll to $h_{min}$") ◁
    Let $h_{\min}$ be the institution among $\{\mu(d_1), h\}$ which $d_1$ prefers least
    Let $p_{\text{new}} = (d_1, h_{\min})$; add $p_{\text{new}}$ to $\Delta$
    if  $h_{\min} = h$  then                                   ▷ We replace $p_1$ with $p_{new}$
        Add an edge from each $p \in P_1$ to $p_{\text{new}}$
        Add $p_{\text{new}}$ to $P$                              ▷ $h$ is still going to propose next
    else                   ▷  Here $h_{min} = \mu(d_1)$; we add $p_{new}$ below the predecessors $P$
        Add an edge from every $p \in P$ to $p_{\text{new}}$
        Set $P = P_1 \cup \{p_{\text{new}}\}$
        Set $h' \leftarrow \mu(d_1)$, then $\mu(d) \leftarrow h$, then $h \leftarrow h'$      ▷ $d_1$'s old match will propose next
```

(2) All of the nodes in P have out-degree 0.

(3) The out-degree of every node in Δ *is at most* 1.

(4) Every source node in Δ *is of the form* (d_*, h_i) *for some* $h_i \in \mathcal{H}_{\text{menu}}$.

(5) For every edge (d_0, h_0) *to* (d_1, h_1) *in* Δ*, we have* $\mu(d_1) = h_0$.

(6) For each $d \in \mathcal{D} \setminus \{d_*\}$*, there is at most one node in* Δ *of the form* (d, h_i) *for some* h_i.

Each of these properties holds trivially at the beginning of the algorithm, and it is straightforward to verify that each structural property is maintained each time AdjustUnrollDag runs.

We now begin to model the properties that Δ needs to maintain as the algorithm runs.

Definition D.7. At some point during the run of any institution-proposing algorithm with preferences Q, define the *truncated revealed preferences* $\overline{Q}$ as exactly those institution preferences which have been queried so far, and assuming that all further queries to all institutions will return $\emptyset$ (that is, assume that all institution preference lists end right after those preferences learned so far).

For some set of preferences Q we say the revealed truncated preferences $\overline{Q}$ and the pair $(\mu', \mathcal{D}'_{\text{term}})$ is a *partial AKL state* for preferences Q if there exists some execution order of Description A.2 and a point along that execution path such that the truncated revealed preferences are $\overline{Q}$, and μ and $\mathcal{D}_{\text{term}}$ in Description A.2 take the values μ' and $\mathcal{D}'_{\text{term}}$

Let Q be a set of preferences which does not include preference of d_*, and let $\overline{Q}$ a truncated revealed preferences of Q. Call a pair (μ, Δ) *unroll-correct for* Q *at* $\overline{Q}$ if 1) μ is the result of $IPDA(\overline{Q})$, and moreover, for every $h \in \mathcal{H}_{\text{menu}}$, the revealed preferences $\overline{Q}$ and pair UnrollOneChain(μ, Δ, h) is a valid partial AKL state of preferences $(\overline{Q}, d_* : \{h\})$.

The following is the main technical lemma we need, which inducts on the total number of proposals made in the algorithm, and shows that (μ, Δ) remain correct every time the algorithm changes their value:

Lemma D.8. *Consider any moment where we query some institution's preferences list withing* NextInterestedApplicant *in Description A.4. Let* h *be the just-queried institution, let* d *be the returned applicant, and suppose that the truncated revealed preferences before that query are* $\overline{Q}$*, and fix the current values of* μ *and* Δ.

Suppose that (μ, Δ) are unroll-correct for Q at $\overline{Q}$.

Now let $\overline{Q}'$ be the revealed preferences after adding d to h's list, and let μ' and Δ' be the updated version of these values after Description A.4 processes this proposal (formally, if NextInterestedApplicant *returns d, fix μ' and Δ' to the values of μ and Δ after the algorithm finishes running* AdjustUnrollDag*; if* NextInterestedApplicant *does not return d, set $\mu' = \mu$ and $\Delta' = \Delta$). Then (μ', Δ') are unroll-correct for Q at $\overline{Q}'$.*

Proof. First, observe that if h's next choice is $\emptyset$, then the claim is trivially true, because $\overline{Q} = Q$ (and AdjustUnrollDAG does not change μ or Δ). Now suppose h's next choice is $d \neq \emptyset$, but is not returned by NextInterestedApplicant. This means that: 1) $d \neq d_*$, 2) $\mu(d) \succ_d h$, and 3) either d does not appear in Δ, or d does appear in Δ, in which case d matched to some h' such that $h' \succ_d h$. Because (μ, Δ) are unroll-correct for Q at $\overline{Q}$, and because Lemma D.6 says that d can appear at most once in Δ, the only possible match which d could be unrolled to at truncated revealed preferences $\overline{Q}$ is h' (formally, if the true complete preferences were $\overline{Q}$, then for all $h_* \in \mathcal{H}_{\text{menu}}$, the partial AKL state under preferences $(\overline{Q}, d : \{h_*\})$ to which we we would unroll would match d to either $\mu(d)$ or h'). But d would not reject $\mu(d)$ in favor of h, nor would she reject h' in favor of h. Thus, (for all choices of $h_* \in \mathcal{H}_{\text{menu}}$) we know h will always be rejected by d, and (μ, Δ) are already unroll-correct for Q at $\overline{Q}'$.

Now, consider a case where h's next proposal $d \neq \emptyset$ is returned by NextInterestedApplicant. There are a number of ways in which AdjustUnrollDAG may change Δ. We go through these cases.

First, suppose $d = d_*$. In this case, the menu of d_* in $\overline{Q}'$ contains exactly one more institution than the menu in $\overline{Q}$, namely, institution h. Moreover, for any $h_* \in \mathcal{H}_{\text{menu}} \setminus \{h\}$, the same partial AKL state is valid under both preferences $(\overline{Q}, d : \{h_*\})$ and $(\overline{Q}', d : \{h_*\})$ (the only difference in $(\overline{Q}', d : \{h_*\})$ is a single additional proposal from h to d_*, which is rejected; the correct value of $\mathcal{D}_{\text{term}}$ is unchanged). For $h_* = h$, the current matching μ, modified to match h to d_*, is a valid partial AKL state for $(\overline{Q}', d : \{h\})$, and this is exactly the result of UnrollOneChain (with $\mathcal{D}_{\text{term}} = \{d_*\}$, which is correct for preferences $(\overline{Q}', d : \{h\})$). Thus, (using also the fact from Lemma D.6 that P contains all nodes in Δ involving h), each possible result of UnrollOneChain is a correct partial AKL state for each $(\overline{Q}', d : \{h_*\})$, so (μ', Δ') is unroll-correct for Q at $\overline{Q}'$.

Now suppose $d \notin \{\emptyset, d_*\}$ is returned from AdjustUnrollDAG, and d does not already appear in Δ. In this case, $h \succ_d \mu(d)$, and for every $h_* \in \mathcal{H}_{\text{menu}}$, the unrolled state when preferences $(\overline{Q}, d : \{h_*\})$ will pair d to $\mu(d)$. Under preferences $(\overline{Q}', d : \emptyset)$, a single additional proposal will be made on top of the proposals of $(\overline{Q}, d : \emptyset)$, namely, h will propose to d and d will reject $\mu(d)$. However, *if h_* is such that h is "unrolled"* (formally, if h_* is such that UnrollOneChain(μ, Δ, h_*) changes the partner of h) then h cannot propose to d in $(\overline{Q}, d : \emptyset)$ (because all pairs in Δ can only "unroll" h to partners before $\mu(h)$ on h's list), nor in $(\overline{Q}', d : \emptyset)$ (because $\overline{Q}'$ only adds a partner to h's list after $\mu(h)$). Thus, for all h_* such that h is unrolled, the pair $(d, \mu(d))$ should be unrolled as well. On the other hand, for all h_* such that h is not unrolled, h will propose to d (matched to d'), so d will match to h in the unrolled-to state. This is exactly how μ' and Δ' specify unrolling should go, as needed.

(Hardest case: AdjustUnrollDagCollision.**)** We now proceed to the hardest case, where $d \notin \{\emptyset, d_*\}$ is returned from AdjustUnrollDag, and d already appears in Δ. In this case, AdjustUnrollDagCollision modifies Δ. Define p_1, P_1, and $h_{\min}$, following the notation of AdjustUnrollDagCollision. Now consider any $h_* \in \mathcal{H}_{\text{menu}}$ under preferences $\overline{Q}$. There are several cases of how h_* may interact with the nodes changed AdjustUnrollDagCollision, so we look at these cases and prove correctness. There are two important considerations which we must prove correct: first, we consider the way that AdjustUnrollDagCollision removes nodes from Δ (starting on Line 19), and second, we consider the way that it creates a new node to handle d (starting on Line 22).

(First part of AdjustUnrollDagCollision.**)** We first consider the way AdjustUnrollDagCollision removes nodes from Δ. There are several subcases based on h_*. First, suppose UnrollOneChain(μ, Δ, h_*) does not contain p_1. Then, because AdjustUnrollDagCollision only drops p_1 and nodes only descended through p_1, the chain unrolled by UnrollOneChain(μ', Δ', h_*) is unchanged until h. (We will prove below that the behavior when this chain reaches h is correct.) Thus, the initial part of this unrolled chain remains correct for Q at $\overline{Q}'$.

On the other hand, suppose that UnrollOneChain(μ, Δ, h_*) contains p_1. There are two sub-cases based on Δ. First, suppose that there exists a pair $p \in P$ in Δ such that p is a descendent of p_1 (i.e. there exists a $p = (d_x, h) \in P$ and a path from p_1 to p in Δ). In this case, under preferences $\overline{Q}$, UnrollOneChain(μ, Δ, h_*) would unroll to each pair in the path starting at h_*, which includes p_1 and all nodes on the path from p_1 to p. Under Δ', however, *none of the nodes from p_1 to p* will be unrolled in this case. The reason is this: in Description A.2, the path from p_1 to p, including the proposal of h to d_1, form an "improvement rotation" when the true preferences are $\overline{Q}'$. Formally, under preferences $(\overline{Q}', d_* : \{h_*\})$, if d_1 rejected h_1, the rejections would follow exactly as in the path in Δ between p_1 and p, and finally h would propose to

d_1. Description A.2 would then call WriteRotation, and the value of μ would be updated for each d on this path. So deleting these nodes is correct in this subcase.[17]

For the second subcase, suppose that there is *no* path between p_1 and any $p \in P$ in Δ. In this case, there must be some source $(d_*, \overline{h})$ in Δ which is an ancestor of some $p \in P$, and such that the path from $(d_*, \overline{h})$ to p does not contain any descendent of p_1. (This follows because each $p \in P$ must have at least one source as an ancestor, and no ancestor of any $p \in P$ can be descendent of p_1.) To complete the proof in this subcase, it suffices to show that at preferences $(\overline{Q}', d_* : \{h_*\})$, we "do not need to unroll" the path in Δ starting at h_* after p_1 (formally, we want to show that if you unroll from μ' the path in Δ from h_* to just before p_1 (including the new node added by the lines starting on Line 22), then this is a partial AKL state of Q at $\overline{Q}'$). The key observation is this: in contrast to preferences $(\overline{Q}, d_* : \{h_*\})$, where pair p_1 is "unrolled", under preferences $(\overline{Q}', d_* : \{h_*\})$, we know h *will propose to* d_1 *anyway*, because d_* will certainly reject $\overline{h}$ (and trigger a rejection chain leading from $(d_*, \overline{h})$ to h proposing to d_1).

(Second part of AdjustUnrollDagCollision.) We now consider the second major task of AdjustUnrollDagCollision, namely, creating a new node to handle d. The analysis will follow in the same way regardless of how the first part of AdjustUnrollDagCollision executed (i.e., regardless of whether there exists a path between p_1 and P). The analysis has several cases. First, suppose (d_*, h_*) is not an ancestor of any node in $P_1 \cup P$ in Δ. This will hold in Δ' as well, so neither UnrollOneChain(μ, Δ, h_*) nor will UnrollOneChain(μ', Δ', h_*) will not change the match of d. Instead, the match of d under UnrollOneChain(μ', Δ', h_*) will be $\mu'(d)$, which is a correct partial AKL state under $(\overline{Q}', d_* : \{h_*\})$, as desired.

Second, suppose h_* is such that (d_*, h_*) is an ancestor of some node in P_1 in Δ. There are two subcases. If $h_{\min} = h$, then we have $\mu(d_1) = \mu'(d_1)$, but when UnrollOneChain(μ', Δ', h_*) is run, we unroll d_1 to h. Correspondingly, in IPDA with preferences $(\overline{Q}', d_* : \{h_*\})$, we know d_1 will not receive a proposal from $\mu(d_1)$ (as this match is unrolled in $\overline{Q}$) but d_1 will receive a proposal from h (as this additional proposal happens in $\overline{Q}'$ but not in $\overline{Q}$, regardless of whether this happens due to a "rejection rotation" of AKL, or simply due to two rejection chains causing this proposal, as discussed above), which d_1 prefers to the unrolled-to match under preferences $\overline{Q}$. Thus, under preferences $(\overline{Q}', d_* : \{h_*\})$, we know d_1 will match to $h_{\min} = h$ in a valid partial AKL-state. So (μ', Δ') is correct for $\overline{Q}'$ in this subcase. If, on the other hand, $h_{\min} = \mu(d_1)$, then in Δ', UnrollOneChain(μ', Δ', h_*) will not contain the new node

[17] This is the core reason why Description A.4 cannot "unroll" to $IPDA(Q, d : \{h_i\})$—instead, it unrolls to a "partial state of AKL".

p_{new}. However, $\mu'(d_1) = h$, and we know d would receive a proposal from h $(\overline{Q}', d_* : \{h_*\})$, and would accept this proposal. So (μ', Δ') is correct for $\overline{Q}'$ in this subcase.

Third and finally, suppose h_* is such that (d_*, h_*) is an ancestor of some node in P in Δ. The logic is similar to the previous paragraph, simply reversed. Specifically, there are two subcases. If $h_{\min} = h$, then when preferences are $(\overline{Q}', d_* : \{h_*\})$, then d_1 will no longer receive a proposal from h, but will still receive a proposal from $\mu(d_1)$. So d_1 should remain matched to $\mu(d_1)$ during $\textsc{UnrollOneChain}(\mu', \Delta', h_*)$, and (μ', Δ') is correct for $\overline{Q}'$ in this subcase. If $h_{\min} = \mu(d_1)$, then $\mu'(d_1) = h$, and in Δ', $\textsc{UnrollOneChain}(\mu', \Delta', h_*)$ will contain the new node p_{new}, which unrolls d_1 to their old match $\mu(d_1)$. This is correct, because in $\overline{Q}$, according to Δ, we know h will be unrolled to some previous match, and correspondingly, in preferences $(\overline{Q}', d_* : \{h_*\})$, we know d_1 will never receive a proposal from h. So (μ', Δ') is correct for $\overline{Q}'$ in this subcase.

Thus, for all cases, (μ', Δ') are unroll-correct for Q at $\overline{Q}'$, as required. □

To begin to wrap up, we bound the computational resources of the algorithm:

Lemma D.9. *Description A.4 is institution-proposing and uses memory $\widetilde{O}(n)$.*

Proof. The institution-proposing property holds by construction. To bound the memory, the only thing that we need to consider on top of AKL is the "unroll DAG" Δ. This memory requirement is small, because there are at most $O(n)$ nodes of the form (d_*, h) for different $h \in \mathcal{H}$, and by Lemma D.6, a given applicant $d \in \mathcal{D} \setminus \{d_*\}$ can appear *at most once* in Δ. So the memory requirement is $\widetilde{O}(n)$. □

We can now prove our main result:

Theorem D.10. *Description A.4 is an institution-proposing, $\widetilde{O}(n)$ memory individualized dictatorship for (applicant-proposing) DA.*

Proof. We know Description A.4 correctly computes the menu, and that it is institution-proposing and $\widetilde{O}(n)$ memory. So we just need to show that it correctly computes the final matching. To do this, it suffices to show that at the end of Phase 1 of Description A.4, (μ, Δ) is unroll-correct for Q at the truncated revealed preferences $\overline{Q}$ (for then, by definition, running Description A.2 after $\textsc{UnrollOneChain}$ will correctly compute the final matching).

To see this, first note that an empty graph is unroll-correct for the truncated revealed preference after running $IPDA(P_{\text{hold}})$, as no further proposals beyond d_* can be made in these truncated preferences. Second, each time we pick an $h \in \mathcal{H}_*$ on

Line 8, a single (d_*, h) added to Δ (with no edges) is unroll-correct for Q at $\overline{Q}'$, by construction. Finally, by Lemma D.8, every other query to any institution's preference list keeps (μ, Δ) unroll-correct after the new query. So by induction, (μ, Δ) is unroll-correct at the end of Phase 1, as desired. □

E Proofs of Known Results

This section is primarily dedicated to reproducing complete proofs from scratch all lemmas we need related to DA and stable matchings. While this adds completeness to the paper, the primary purpose of including these proofs is to facilitate a comparison between two approaches to proving the strategyproofness of DA: first, the proof of Theorem 3.1 given in Appendix C, which shows the correctness of Description 1 without relying on the strategyproofness of DA, and second, a classical, direct proof presented in Section E.2.

E.1 Lemmas for Proving Theorem 3.1

Here, we supply all lemmas needed for the direct proof of Theorem 3.1 given in Appendix C.

Lemma E.1 (Gale and Shapley, 1962)**.** *The output of DA is a stable matching.*

Proof. Consider running the traditional description of DA on some set of preferences, and let the output matching be μ. Consider a pair $d \in D$, $h \in H$ which is unmatched in μ. Suppose for contradiction $h \succ_d \mu(d)$ and $d \succ_h \mu(h)$. In the DA algorithm, d would propose to h before $\mu(d)$. However, it's easy to observe from the traditional description of DA that once an institution is proposed to, they remain matched and can only increase their preference for their match. This contradicts the fact that h was eventually matched to $\mu(h)$. □

Note that Lemma E.1 gives a very interesting existence result: it was not at all clear that stable matching existed before it was proven.

Lemma E.2 (Gale and Shapley, 1962)**.** *If an applicant $d \in D$ is ever rejected by an institution $h \in H$ during some run of APDA (that is, d proposes to h and h does not accept) then no stable matching can pair d to h.*

Proof. Let μ be any matching, not necessarily stable. We will show that if h rejects $\mu(h)$ at any step of DA, then μ is not stable.

Consider the first time during in the run of DA where such a rejection occurred. In particular, let h reject $d \overset{\text{def}}{=} \mu(h)$ in favor of $\widetilde{d} \neq d$ (either because $\widetilde{d}$ proposed to h, or because $\widetilde{d}$ was already matched to h and d proposed). We have $\widetilde{d} \succ_h d$. We have $\mu(\widetilde{d}) \neq h$, simply because μ is a matching. Because this is the first time any applicant has been rejected by a match from μ, $\widetilde{d}$ has not yet proposed to $\mu(\widetilde{d})$. This means $h \succ_{\widetilde{d}} \mu(\widetilde{d})$. However, this means μ is not stable.

Thus, no institution can ever reject a stable partner in applicant-proposing DA. □

Lemma E.2 immediately implies that the result of applicant-proposing DA is the optimal stable outcome for each applicant, and that the result is independent of the order in applicant proposals are made.

Corollary E.3 (Gale and Shapley, 1962)**.** *In the matching returned by APDA, every applicant is matched to her favorite stable partner.*

Corollary E.4 (Dubins and Freedman, 1981)**.** *The matching output by the DA algorithm is independent of the order in which applicants are selected to propose.*

A dual phenomenon occurs for the institutions:

Lemma E.5 (McVitie and Wilson, 1971)**.** *In the match returned by applicant-proposing DA, every $h \in H$ is paired to their worst stable match in D.*

Proof. Let $d \in D$ and $h \in H$ be paired by applicant-proposing deferred acceptance. Let μ be any stable matching which does not pair d and h. We must have $h \succ_d \mu(d)$, because h is the d's favorite stable partner. If $d \succ_h \mu(h)$, then μ is not stable. Thus, h cannot be stably matched to any applicant they prefer less than d. □

Finally, the applicant and institution optimality conditions can be combined to prove that the set of matched agents must be the same in each stable matching as the corresponding set in APDA (and thus, the same in every stable matching).

Theorem E.6 (Lone Wolf / Rural Hospitals Theorem, Roth, 1986)**.** *The set of unmatched agents is the same in every stable matching.*

Proof. Consider any stable matching μ in which applicants D^μ and institutions H^μ are matched, and let D^0 and H^0 be matched in APDA. Each applicant in $D \setminus D^0$ proposes to all of his acceptable institutions. Thus, by Lemma E.2, no stable matching can possibly pair any applicant in $D \setminus D^0$ with any institution. Thus, we have $D \setminus D^0 \subseteq D \setminus D^\mu$, and in turn $D^0 \supseteq D^\mu$. On the other hand, Lemma E.5 implies that each agent in H^0 is matched in every stable outcome, so $H^0 \subseteq H^\mu$. But then we have $|D^0| = |H^0|$ as well as $|D^0| \geq |D^\mu| = |H^\mu| \geq |H^0|$, so the same number of agents (on each side) are matched in in μ as in APDA. Thus, $D^0 = D^\mu$ and $H^0 = H^\mu$. □

Remark E.7. One can carefully check that all of our earlier proofs work when we consider the match of an agent to be $\emptyset$—this was intentional in order to make the proof of Theorem E.6 go through. Thus, these proofs seamlessly handle partial lists and market imbalance.

E.2 Direct Proof of the Strategyproofness of DA from its Traditional Description

To contrast between Section E.1 and Theorem 3.1, we also include a direct proof of the strategyproofness of DA, adapted from Gale and Sotomayor (1985). Note, however, that the following proof also shows that DA is *weakly group* strategyproof, whereas Theorem 3.1 does not.

Lemma E.8 (Attributed to J.S. Hwang by Gale and Sotomayor, 1985)**.** *Let $\mu = APDA(P)$ and μ' be any other matching. Let T denote the set of all applicants who* strictly *prefer their match in μ' to their match in μ, and suppose $T \neq \emptyset$. Then there exists a blocking pair (d, h) in μ'* with *$d \notin T$.*

Proof. We consider two cases. Let $\mu(T)$ denote the set of matches of agents in T under μ (and similarly define $\mu'(T)$).

Case 1: $\mu(T) \neq \mu'(T)$. Every applicant in T is matched in μ', so $|\mu'(T)| \geq |\mu(T)|$. Thus, there exists some $h \in \mu'(T)$ but $h \notin \mu(T)$, that is, $h = \mu'(d')$ with $d' \in T$ but $h = \mu(d)$ with $d \notin T$. By the definition of T, we have $h = \mu(d) \succ_d \mu'(d)$. Because $h \succ_{d'} \mu(d')$, we know d' would propose to h in $APDA(P)$. So $d = \mu(h) \succ_h d'$. Thus, (d, h) is a blocking pair in μ' (with $d \notin T$).

Case 2: $\mu(T) = \mu'(T)$. This case is a bit harder. Consider the run of $APDA(P)$. First, for any $d \in T$, note that d must have proposed to $\mu'(d)$ before proposing to $\mu(D)$, so each institution in $\mu(T)$ receives at least two proposals from applicants in T.

Now, consider the *final* time in a run of $APDA(P)$ when an applicant $d_f \in T$ proposes to an institution h in $\mu(T)$. As h receives at least two proposals from applicants in T, we know h must be tentatively matched, say to d, and h must reject d for d_f. However, d cannot herself be in T, as then d would need to make another proposal to $\mu(d) \in \mu(T)$ (and we assumed this is the final proposal from an applicant in T to an institution in $\mu(T)$).

We claim that (d,h) is a blocking pair in μ'. Proof: As $d \notin T$ and d proposes to h during $APDA(P)$, we have $h \succ_d \mu(d) \succeq_d \mu'(d)$. Now, again consider when h rejects d in $APDA(P)$. At this point in time, h has already rejected every agent in T, other than $d_f = \mu(h)$, who proposes to h during $APDA(P)$. In particular, h has already rejected $\mu'(h) \in T$ (who also proposes to h in $APDA(P)$, as noted above), so $d \succ_h \mu'(h)$.

Thus, in either case there exists a blocking pair (d,h) in μ' with $d \notin T$. □

Theorem E.9 (Roth, 1982; Dubins and Freedman, 1981)**.** *APDA is (weakly group-)strategyproof for the applicants.*

Proof. Suppose a set L of applicants change their preferences, and each of them improve their match. In particular, if $\mu' = APDA(P')$, where P' is the altered list of preferences, then $L \subseteq T$ as in Lemma E.8. Thus, there exists a blocking pair (d,h) for μ' under preferences P, where we additionally have $d \notin T$. In particular, $d \notin L$. Thus, d and h each keep their preferences the same in P' as in P. So, (d,h) is also a blocking pair under preferences P', so μ' cannot possibly be stable under P'. This is a contradiction. □

References for Appendices

I. Ashlagi, Y. Kanoria, and J. D. Leshno. Unbalanced random matching markets: The stark effect of competition. *Journal of Political Economy*, 125(1):69 – 98, 2017. Abstract in Proceedings of the 14th ACM Conference on Electronic Commerce (EC 2013).

S. Assadi. Lecture 2: The maximum bipartite matching problem, 2020. URL https://people.cs.rutgers.edu/~sa1497/courses/cs671-f20/lec2.pdf. Lecture notes for Rutgers course CS 671: Graph Streaming Algorithms and Lower Bounds.

Y. Breitmoser and S. Schweighofer-Kodritsch. Obviousness around the clock. *Experimental Economics*, 25:483–513, 2022.

L. Cai and C. Thomas. Representing all stable matchings by walking a maximal chain. Mimeo, 2019. URL https://arxiv.org/abs/1910.04401.

L. Cai and C. Thomas. The short-side advantage in random matching markets. In *Proceedings of the 5th SIAM Symposium on Simplicity in Algorithms (SOSA)*, pages 257–267, 2022.

E. L. Dubins and A. D. Freedman. Machiavelli and the Gale-Shapley algorithm. *American Mathematical Monthly*, 88:485–494, 1981.

D. Gale and L. S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–14, 1962.

D. Gale and M. Sotomayor. Some remarks on the stable matching problem. *Discrete Applied Mathematics*, 11(3):223–232, 1985.

D. Gusfield and R. Irving. *The stable marriage problem: Structure and algorithms*. MIT Press, 1989.

J. W. Hatfield and P. R. Milgrom. Matching with contracts. *American Economic Review*, 95(4):913–935, 2005.

N. Immorlica and M. Mahdian. Marriage, honesty, and stability. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 53–62, 2005.

S. Li. Obviously strategy-proof mechanisms. *American Economic Review*, 107(11):3257–87, 2017.

D. G. McVitie and L. B. Wilson. The stable marriage problem. *Communications of the ACM*, 14(7), 1971.

A. E. Roth. The economics of matching: stability and incentives. *Mathematics of Operations Research*, 7(4):617–628, 1982.

A. E. Roth. On the allocation of residents to rural hospitals: A general property of two-sided matching markets. *Econometrica*, 54(2):425–427, 1986.

A. E. Roth and A. Postlewaite. Weak versus strong domination in a market with indivisible goods. *Journal of Mathematical Economics*, 4(2):131–137, 1977.

L. Shapley and H. Scarf. On cores and indivisibility. *Journal of Mathematical Economics*, 1(1):23–37, 1974.

L. B. Wilson. An analysis of the stable marriage assignment algorithm. *BIT Numerical Mathematics*, 12(4):569–575, Dec 1972. ISSN 1572-9125. doi: 10.1007/BF01932966. URL https://doi.org/10.1007/BF01932966.