Strategyproofness-Exposing Mechanism Descriptions*

Yannai Gonczarowski† Ori Heffetz‡ Clayton Thomas§

August 2, 2022

Preliminary draft, please do not circulate

Abstract

A *menu description* defines a mechanism to player i in two steps: Step (i) uses the reports of other players to describe the set of outcomes i might receive; Step (ii) uses i's report to select i's favorite outcome from this set. Can menu descriptions better convey strategyproofness, without sacrificing simplicity? We propose a new, simple menu description of Deferred Acceptance. We prove that—in contrast to other common matching mechanisms—this menu description must differ greatly from traditional descriptions. We demonstrate, with lab experiments, the promise and challenges of menu descriptions.

*The authors thank Robert Day, Benjamin Enke, Xavier Gabaix, Nicole Immorlica, David Laibson, Shengwu Li, Paul Milgrom, Assaf Romm, Al Roth, Ilya Segal, Ran Shorrer, Matt Weinberg, Leeat Yariv, Huacheng Yu, and participants in HUJI's JESC, Cornell's BERG, the 2020/2021 DIMACS Matching Reading Group, Harvard's CS Theory Seminar, Stanford GSB's Economics Seminar, WALE 2022, Noam Nisan's 60th Birthday Conference, and the 4th ACM/INFORMS Workshop on Market Design at EC 2022, 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, and Yonatan Rahimi provided excellent research assistance. Parts of the work of Gonczarowski were 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 and a Wallace Memorial Fellowship in Engineering.

†Department of Economics and Department of Computer Science, Harvard University — *E-mail*: yannai@gonch.name.

‡Johnson Graduate School of Management, Cornell University, Bogen Department of Economics and Federmann Center for Rationality, The Hebrew University of Jerusalem, and NBER — *E-mail*: oh33@cornell.edu.

§Department of Computer Science, Princeton University — *E-mail*: claytont@princeton.edu.

1 Introduction

Mechanism design strives to achieve desirable outcomes in the presence of strategic behavior. A foundational idea of the field is strategyproof social-choice rules, which eliminate any need for "perfectly rational" players to strategize. Unfortunately, real people do not conform to the economic assumptions often placed upon them; significant and growing evidence shows that real people frequently do not play their theoretically dominant strategies.[1]

Recent work provides different theories for why people play seemingly dominated strategies, and proposes potential solutions. These new explanations of non-straightforward[2] behavior include contingent-reasoning failures (Li, 2017), limited foresight (Pycia and Troyan, 2021), and expectations-based reference-dependent (EBRD) preferences (Dreyfuss et al., 2022). The suggested solutions share a common overarching approach: given a fixed strategyproof social-choice rule, implement it via different *interactive* mechanisms. While already making a seminal contribution, this line of work has limitations. It may be theoretically restrictive (i.e., provably impossible for many desirable mechanisms[3]), impractical (when a dynamic implementation poses challenges), or empirically incomplete (i.e., leaving some non-straightforward behavior unexplained).

In this paper, we take a step further: given a *fixed implementation* of a fixed strategyproof social-choice rule, we investigate different ways to *describe* it to participants. We propose, and experimentally motivate, a general framework for describing a mechanism to one player at a time, in a way that can make strategyproofness easier to see. Crafting descriptions to emphasize specific information to participants is, of course, the bread and butter of experimentalists and practitioners. To our knowledge, ours is the first *theoretical* study of different ways to describe mechanisms. By proving theorems about what can or cannot be achieved, we contribute to the science behind what is often more of an art.

Our main application is to deferred acceptance (DA) (Gale and Shapley, 1962). This matching rule is widely used (Roth, 2002), from school choice (Abdulkadiroğlu and Sönmez, 2003) to medical residency matching. While DA has many advantages, its strategyproofness is delicate to prove from its traditional description. We present a novel, simple description of DA that makes its strategyproofness easy to see. Furthermore, we prove a precise sense

[1] This evidence comes both from the field (Hassidim et al., 2017; Shorrer and Sóvágó, 2017; Rees-Jones, 2018) and the lab (Kagel and Levin, 1993; Li, 2017; Chen and Sönmez, 2006; Hakimov and Kübler, 2019). It persists (somewhat ameliorated) even when the participants are explicitly informed of the strategyproofness of the mechanism (Guillen and Hakimov, 2018; Masuda et al., 2019).

[2] Throughout this paper, we use the term "straightforward" to describe the strategy an agent would play under classic economic assumptions (sometimes called the "truthtelling" strategy).

[3] Of all the mechanisms we will discuss in this introduction, only two have simple mechanisms in the theoretical framework of Li (2017): the second price auction, and the serial dictatorship matching rule.

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

(a) **Median Voting:** Two descriptions of the median voting rule with three voters with single-peaked preferences (candidates are represented by numbers). For each voter, voting for her favorite candidate is a dominant strategy. Each voter's menu consists of all candidates between the votes of the other two voters.

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.

(b) **Second-Price Auction:** Two descriptions of a single-item, sealed bid, second-price auction. For each bidder, submitting a bid equal to her value for the item is a dominant strategy. 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.

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.

in which our new description must differ substantially from the traditional one, and we investigate theoretical tradeoffs between alternative descriptions.

To make things 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 as direct a way as possible, the latter attempts to first and foremost 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 that i might receive, given the reports of all other players. We define a two-step form for a *menu description* for player i, 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 highest-ranked outcome (as ranked in her report) from her menu.

The main idea behind this paper is that while strategyproofness might be hard to infer from traditional descriptions of some mechanisms, it can be easily shown for menu descriptions. Indeed, a player has to notice only that in Step (1) her menu cannot be affected by her report, and that in Step (2) straightforward reporting ("truthtelling") guarantees her favorite

outcome from the menu.

To further motivate our theoretical investigation, we experimentally explore how real participants respond to menu descriptions in simple settings. We conduct a preregistered (between-subjects) lab experiment using essentially the two pairs of descriptions 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 simple mechanisms, strategyproofness may be equally apparent from traditional and menu descriptions.

Every strategyproof mechanism has a menu description (Hammond, 1979). In particular, every such mechanism has a "brute force" menu description, in which Step (1) calculates the menu by going over every hypothetical report player i could have submitted, and collecting all possible outcomes. However, a brute-force description is often impractical to convey to participants.[4] Our theoretical framework therefore focuses on the strengths and limitations of *simple* menu descriptions—those that are practical to convey.

After providing preliminaries in Section 2, in Section 3 and onward we turn our attention to matching rules, a common setting in which complicated mechanisms are used in practice to match *applicants* to *institutions*.[5] For one popular matching mechanism, *Serial Dictatorship* (henceforth SD), the traditional description *already is* a menu description. For another popular mechanism, *Top Trading Cycles* (henceforth TTC), we show that, perhaps surprisingly, a simple menu description can be constructed for one player at a time via only a slight modification of the traditional description. Finally, we consider our main application, DA.

DA[6] is traditionally described via an "applicant-proposing deferred acceptance" procedure, where applicants iteratively propose to institutions in order of their reported prefer-

[4]For example, even for the relatively simple second price auction, this menu description will seem almost obfuscated: Step (1) would describe the constructed menu as containing possibly many prices for winning the item, and Step (2) would pick the utility-maximizing (allocation, price) pair out of this menu. Compare this description with the considerably simpler menu description in Table 1(b).

[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*.

[6]Since the name Deferred Acceptance refers to properties of this traditional description, it may have in fact been more accurate to call this direct mechanism the Applicant-Optimal Stable Matching mechanism, which describes the outcome. We nonetheless follow popular usage and refer to the mechanism as DA, and only use "applicant-optimal stable matching" when confusion might arise.

Table 2: **Deferred Acceptance:** Two descriptions of the *applicant-optimal* stable matching. (Both use the traditional deferred acceptance procedure as a building block; the menu version describes only the match of the applicant reading the description.)

Traditional Desc.:	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.

ences, and each institution tentatively accepts the best proposal (according to its priorities) it received so far. The resulting matching is the applicant-optimal stable outcome (Gale and Shapley, 1962). From the point of view of applicants, who are often laypeople with no experience with economic mechanisms, arguably the most important facet of DA is its strategyproofness. However, clearinghouses are rarely able to elucidate *why* DA is strategyproof, as proving this requires an involved mathematical argument.[7]

Finding an appealing menu description of DA is harder than finding an appealing menu description of TTC (a theme we investigate formally below). Nevertheless, our main positive result is a simple, novel menu description of DA, presented in Table 2. Quite surprisingly, this menu description is virtually as streamlined as the traditional description is. Step (1) simply consists of a "flipped-side-proposing" deferred acceptance procedure among all participants except some player i (to whom it is described), and player i's menu consists of those institutions that prefer i to the applicant matched to them by this flipped-side-proposing procedure. In sharp contrast with the traditional description of DA, strategyproofness can be readily seen from this menu description in the same way as the simple examples in Table 1. We emphasize that it was previously unknown that the menu in DA can be described as in Table 2, or, indeed, in any other way that does not require *multiple* runs of DA while going over different hypothetical reports by player i.

While our new description provides a simple characterization of the menu in DA, one may wonder whether it must differ so significantly from the traditional description. Indeed, our menu descriptions of SD and TTC are obtained by observing that a traditional description

[7]One approach taken by clearinghouses is to instead rely on appeals to authority. As reported by Dreyfuss et al. (2022), an informative video published by the 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.

(or its tweak) "contains" a menu description. Namely, for each player i, there is a traditional description of each of SD and TTC with the following *three*-step outline, whose first two steps 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 such a description an *individualized dictatorship*.

In Section 4, we define a general model for reasoning about mechanism descriptions, including individualized dictatorships. In Section 5, we prove our main impossibility theorem: in contrast to SD and TTC, no individualized dictatorship for DA is *remotely* similar to its traditional description. Since DA is commonly described essentially as an algorithm, it is only natural to capture similarity to the traditional description through an algorithmic lens. We therefore delineate two fundamental properties of the traditional description of DA.[8] First, it is applicant-proposing (querying applicants' preferences in favorite-to-least-favorite order). Second, it works by iteratively modifying a single tentative matching (without performing more complicated bookkeeping or rote memorization; formally, using $\widetilde{O}(n)$ bits of memory). We prove that any applicant-proposing individualized dictatorship description of DA must keep track of vastly more information than merely 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 tentative matchings; formally, requiring $\Omega(n^2)$ bits of memory). Thus, in contrast to (SD and) TTC, no small tweak of the traditional description of DA contains a menu description. This provides a formal sense in which it is difficult to recognize that DA "has a menu"—and hence is strategyproof—from its traditional description.

While our menu description of DA may expose its strategyproofness, it makes it hard to see that the matching collectively described by the different players' menu descriptions is feasible (i.e, that no two applicants infeasibly get the same assignment). In contrast, the traditional description may make feasibility easier to see, but strategyproofness hard to see. Could some other description make both properties easy to see? In Section 6, we provide a full classification of one-side-proposing descriptions of DA. We search for menu descriptions (which emphasize strategyproofness), descriptions of the full matching (which emphasize feasibility), and individualized dictatorships (which combine both to emphasize both). Most significantly, we prove an additional impossibility theorem: any conceivable description in

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

our classification—other than traditional DA and our new menu description—cannot be as simple to follow as these two descriptions, and must in some precise sense be convoluted. Roughly speaking, such a description must update bookkeeping concerning some participants when making queries that are unrelated to them.[9] For simple, practical descriptions of DA, 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.

In Section 7 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 8, review related work in Section 9, and conclude in Section 10, where we also discuss limitations of our framework.

2 Preliminaries

This paper studies strategyproof social choice functions in different economic environments. We define all these terms fully in Appendix B. Briefly, an *environment* has some set of possible outcomes A, and n agents (also known as players) with their own desires regarding the outcomes. These are modeled by "types" $t_i \in \mathcal{T}_i$ of the agents, where $\mathcal{T}_i$ denotes the set of possible types player i might have in the given environment. We write $\succ_i^{t_i}$ for the preferences of type t_i over the outcomes.

Definition 2.1. *A* social choice function *in some environment 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 $t_{-i} \in \mathcal{T}_{-i}$, we have*

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

When we study matching environments, each player is an *applicant*, and outcomes match each player to an *institution*. The types of applicants are simply their ordinal preferences over the institutions. We denote the type of player i as $\succ_i$, and write $x \succ_i y$ if player i prefers institution x to institution y. 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*. We study the canonical mechanisms of Serial Dictatorship (SD), Top Trading Cycles (TTC),

[9] In fact, in Section 6 we construct descriptions of every type not ruled out by our main impossibility theorem. Unfortunately, each is exceedingly convoluted and impractical, which is what inspired this additional impossibility result.

and Deferred Acceptance (DA), all fully defined in Appendix B.

In many interesting environments (especially those without externalities), players do not "care about" the entire outcome, but only 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 under the indifference of bidder i. (For a formal definition, see Definition B.2.) In matching environments, agents are assumed to only care about their own match, and thus the i-outcomes are defined only by the match of player i. For a general environment, we denote the i-outcome corresponding to a by $[a]_i$, and the set of i-outcomes by A_i.

2.1 Menus

The central notions of our paper are the definition of the *menu* presented by a mechanisms, and the alternative definition of strategyproofness that this definition provides. Briefly, the menu[10] presented by $t_{-i} \in \mathcal{T}_{-i}$ is the set of outcomes that player i might receive when other players have types t_{-i}.

Definition 2.2. *For any social choice function f, the menu $\mathtt{M}_{t_{-i}}$ of player i given types $t_{-i} \in \mathcal{T}_{-i}$ is the subset of all i-outcomes $a \in A_i$ for which there exists some $t_i \in \mathcal{T}_i$ such that $f(t_i, t_{-i}) \in a$. That is,*

$$\mathtt{M}_{t_{-i}} = \left\{ \; [f(t_i, t_{-i})]_i \;\middle|\; t_i \in \mathcal{T}_i \right\} \subseteq A_i.$$

Theorem 2.3 (Hammond, 1979)**.** *A social choice function 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 $x \in \mathtt{M}_{t_{-i}}$.*

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 their favorite i-outcome from her menu, then player i always prefers submitting type 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}, and then assigns $[f(t_i, t_{-i})]_i$ to i using t_i. This paper seeks menu descriptions that are as natural and easy-to-understand as traditional descriptions. First, for our main domain of interest,

[10] This definition has gone by a few different names: Hammond (1979) says that the menus "decentralize the mechanism," and Guesnerie (1981, 1998) refers to Theorem 2.3 as the "taxation principle." Leshno and Lo (2021) call them "proper budget sets." We follow the EconCS literature (Hart and Nisan (2017), Dobzinski (2016), and follow-ups) in calling these sets "menus." This definition is distinct from different definitions of menus sometimes used in market design papers such as Mackenzie and Zhou (2020); Bó and Hakimov (2020).

we give a *non-simple* menu description as a baseline:[11]

Example 2.4 (A universal *non*-simple menu description for matching mechanisms)**.** *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. For any description D of f, consider the following procedure:*

(i) Given types $\succ_{-i}$, execute D on each type profile of the form $(\{h\}, \succ_{-i})$ for each institution h separately. Let M be the set of all institutions h such that i is matched to h at the end of the corresponding execution of D.

(ii) Match applicant i to her highest-ranked institution in M, according to $\succ_i$.

By strategyproofness, h will be included in M in Step (i) if and only if h is on the menu. Thus, Example 2.4 is a menu description of f. This menu description is sharply less intuitive than standard descriptions of mechanisms such as DA and TTC, and any information about agent i's menu is acquired "one institution at a time" by checking a separate execution of D. Thus, we look for more natural menu descriptions.

3 A Simple Menu Description of Deferred Acceptance

In this section, we present our simple menu description of (applicant-optimal) Deferred Acceptance. This is given in Description 1 (and also Table 2 from the introduction). In terms of pure algorithmic simplicity, this description is quite comparable to traditional descriptions of DA; in fact, it simply adds a very straightforward (and perhaps even intuitive) "menu calculation and matching" step on top of (a traditional description of) DA. Note, though, that *institution*-proposing DA is used here to calculate an applicant's outcome in the *applicant*-optimal DA outcome, which is traditionally described via *applicant*-proposing DA.

Description 1 A menu description for DA for applicant $d \in \mathcal{D}$

(i) 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 $d \succ_h \mu_{-d}(h)$.

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

From the point of view of an applicant, this description is qualitatively different from

[11] A menu description essentially equivalent to this one was independently described by Katuščák and Kittsteiner (2020).

traditional descriptions of the same mechanism, i.e., from the applicant-proposing DA algorithm. Traditional descriptions of DA cleanly explain that every applicant will be matched to a different institution. However, strategyproofness remains completely opaque (requiring complex mathematical proof such as in Section F.2). In Description 1, the situation reverses: strategyproofness is easy to observe, but the reasoning behind why this procedure produces a 1-to-1 matching (let alone a stable matching) becomes quite complex (and in some ways turns out to be equivalent to proofs that DA is strategyproof). In adopting this description, a clearinghouse could shift the "burden of complexity" from deducing strategyproofness (a primary concern of the applicants) to deducing that the result is a feasible and/or stable matching (a primary concern of the clearinghouse or policymakers, rather than the applicants themselves).

Formally, our theorem giving the correctness of this description is the following:

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).*

We prove Theorem 3.1 in Appendix C. All told, Description 1 provides a complete, simple, and accurate description of the match of each applicant, and in stark contrast to traditional descriptions, one can see for themselves why this process is strategyproof. We thus contend that this novel description could provide a viable and useful approach to describing real-world stable matching algorithms.[12]

3.1 On Deploying Our Menu Description

In this section, we make a few remark for real-life centralized clearinghouses that use deferred acceptance and might consider switching from describing it traditionally to describing it using our novel menu description, to make strategyproofness far easier to grasp. While our description differs in crucial ways from traditional DA descriptions, we contend that it is plausibly understandable to many laypeople, and far better conveys the role that one's rank-order-list plays in deciding their match. In clarifying the role one's list plays, the strategyproofness of deferred acceptance becomes clear.

Of course, real-world matching markets already advertise strategyproofness prominently. However, they are just unable to explain precisely *why* strategyproofness holds. For interested applicants, they may mention some of the economic theory that underpins the

[12]We mention here that Theorem 3.1 extends to many-to-one matching markets. Specifically, to adapt Figure 1, one can run the many-to-one institution-proposing DA (without applicant d). This algorithm creates an "independent copy" of each institution for every "seat" it has available, and can intuitively be understood in terms of the "available seats" proposing to the applicants. We show in Remark C.2 that this constitutes a menu description of DA in this many-to-one setting.

mechanism. Description 1 makes this strategyproofness easier, but deflects the problem of explaining why every applicant won't end up matched to the same institution.[13] Similarly to the current situation with stratgyproofness, if a clearinghouse described the mechanism with Description 1, they could communicate (for the interested applicants) that the procedure is equivalent to the same DA algorithm that has always been used. While an applicant who is curious about the global matching and about stability still (in principle) has to understand mathematical proofs, an applicant who only cares about her match (and about how best to rank institutions) can observe these directly from Description 1.

In addition to just demonstrating strategyproofness, Description 1 affords 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" is both unclear (in what is meant by best) and entirely unverifiable. Instead of clearinghouses instructing agents that "they should not strategize" (an often borderline moral command[14]). Description 1 gives a concrete reason why submitting your true preference order *is* strategizing—this list will always award an applicant their favorite attainable institution.

There may be additional psychological framing benefits to our description. It is much harder for an applicant to confuse an institution-proposing, menu-picking process such as Description 1 with a mechanism such as Boston, where seats can fill up and the mechanism is far from strategyproof. In this way, Description 1 may help reduce confusion or avoid applicants developing misguided intuitions as to how they should behave in DA, and clarify that in contrast with real-world "job hunt" types of processes, there are no opportunity costs in DA to trying to get a "dream match." Finally, instead of framing DA in terms of applicants getting repeatedly rejected (or even in terms of the applicant whose menu is explained rejecting institutions that she does not want), Description 1 frames DA in terms of "choosing the best institution" out of a set, a far more positive framing that may have considerable advantages of its own.

[13] For this reason, *public trust* that the clearinghouse will implement the rule as described must be high. But the target of the trust is shifted: from the "advice" on how applicants should rank their lists, to the concrete factual description of the process, which may be harder for applicants to "doubt," and which is already required under traditional descriptions as well. Still, the full implications of this "shift of trust" may be complex, and should be carefully considered by policymakers.

[14] 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...".

4 Formal Models of Mechanism Descriptions

While we find Description 1 from Section 3 appealing, many questions remain regarding what is possible with simple and natural menu descriptions. In this section, we introduce a general formal model for reasoning about descriptions of mechanisms. As a guide towards natural descriptions, and as a lens into the hardness of "revealing strategyproofness" of a description, we also formally define a model for descriptions that compute the menu "within" traditional descriptions.

4.1 Extensive-Form Descriptions of Mechanisms

Mechanisms can be described to agents in a variety of ways. The present paper alone contains a range of options, including textual descriptions, illustrative examples, and algorithmic pseudocode. To abstract over all these possibilities, yet formally reason about what can or cannot be achieved by a description, we introduce the definition of an *extensive-form description*.

At a technical level, an extensive-form description is 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).[15] Note, however, that the semantic interpretation is very different than that of an extensive-form mechanism: Rather than modeling an interactive process where the agents may act multiple times, an extensive-form description spells out the steps of calculating the social choice function using the directly reported types of the players.

Definition 4.1. *An* extensive-form description *in some social choice environment is defined by a directed graph on some set of vertices V. There is a single source vertex $s \in V$, and the vertices of V are organized into "layers" $i = 1, \ldots, L$ such that each edge goes between layer i and $i+1$ for some i. The sink vertices in V are labeled with outcomes from the environment. For a vertex v, let $S(v)$ denote the edges outgoing from v. Each vertex v is labeled with some player i and some function $\ell_v : \mathcal{T}_i \to S(v)$ from types of player i to edges outgoing from v. (We also allow vertices with out-degree* 1*, which are not labeled with any player.)*

An extensive-form description D defines a unique social choice function f_D as follows: to compute $f_D(t_1, \ldots, t_n)$, follow the execution path *on inputs $(t_1, \ldots, t_n)$, which we define as follows. Start in the source vertex s. At every vertex v labeled by player i and ℓ_v, the next edge is $\ell_v(t_i)$. Eventually a sink vertex w labeled by some outcome a is reached. The out-*

[15] Another interpretation of extensive-form descriptions, borrowed from the computer science literature, is as a certain type of "branching program" computing the social choice function. Another viewpoint is that an extensive-form description is a finite automaton where state transitions are given by querying the types of players.

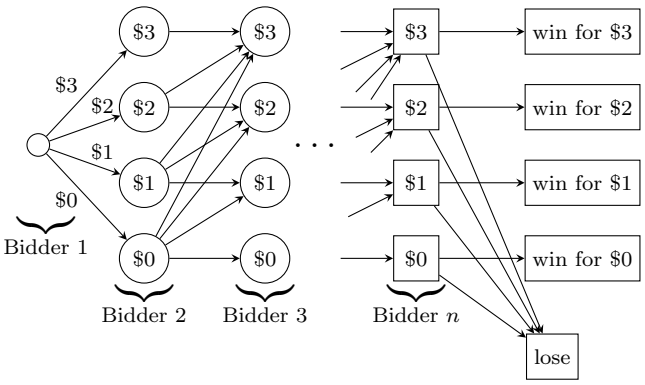


Figure 1: An extensive-form menu description for player n in a second price auction (i.e. an extensive-form formalization of the menu description in Table 1(b)). The mechanism visits the bidders one by one and keeps track of only the highest bid seen so far, requiring fairly low memory. The second to last layer is labeled with bidder n's menu, which is defined by the "price they must pay to win the item".

come of f_D is defined as $f_D(t_1, \ldots, t_n) \triangleq a$. For a social choice function f, a extensive-form description for f *is any extensive-form description D such that $f_D = f$.*

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 V. This is precisely the number of bits required to store that vertex number of the current vertex within a layer.*

We sometimes abuse notation and refer to an extensive-form description simply as a "description". See Figure 1 for an illustration of an extensive-form description.

Any precise and complete "algorithmic" description of a mechanism induces a extensive-form description in a natural way: the nodes in layer i are the possible states of the algorithm (or automaton) after querying the types of different players i times. Note, however, than a extensive-form description can also compute social choice function in a "non-algorithmic" manner, e.g., by "fully remembering" the type of each player: having a separate sink vertex for each type profile, and simply labeling each sink vertex with the right outcome.[16] Because k bits of information can represent 2^k possible states, the memory requirement of an extensive-form description gives a natural quantification for the amount of bookkeeping the description requires – intuitively, it is (up to constant factors) the amount of "scratch paper" that the extensive-form description would need to keep track of its current state throughout an execution path.

4.2 Individualized Dictatorship Descriptions

Recall how the Serial Dictatorship (SD) matching mechanism looks, from the point of view of applicant i:

[16] Moreover, algorithms typically must work for any number of players, where one can use a separate and "custom made" extensive-form description for each number of players n (in computer science terminology, this means that a extensive-form description can be *nonuniform*, and this is an "information-theoretic" model of computation).

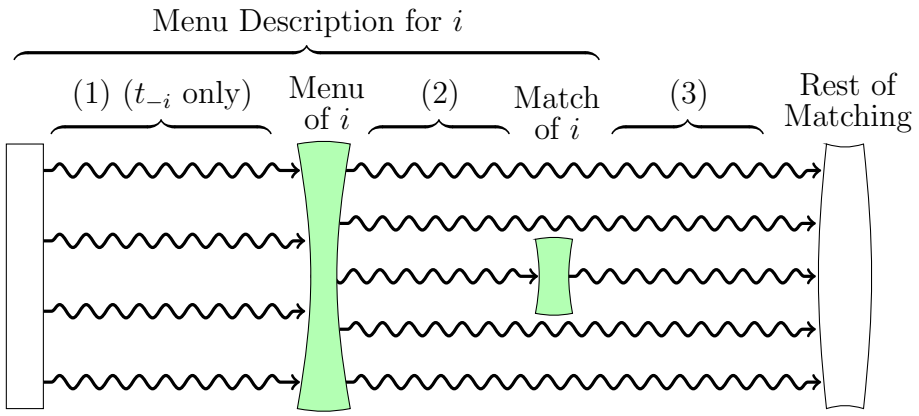


Figure 2: An individualized-dictatorship description for player i.

(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 institutions.

While Steps (1) and (2) form a menu description in SD, what makes SD quite simple is that the menu description lies *within* traditional descriptions. We ask if this is "approximately" the case for other matching mechanisms as well. In other words, we observe that the strategyproofness of this description for player i would still be easy to see for *any* possible computations in Steps (1) and (3), so long as Step (1) does not use player i's type. We ask whether a generalized form of this outline can explain the strategyproofness of other mechanisms. To this end, we formally define *individualized dictatorship descriptions*; see Figure 2 for an illustration.

Definition 4.2. *For any strategyproof social choice function f, an* individualized dictatorship description *of f for player i is a procedure for computing f with* three *stages:*

(1) Using $t_{-i} \in \mathcal{T}_{-i}$ as input, calculate the menu $\mathtt{M}_{t_{-i}}$ of player i given t_{-i}.

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

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

An individualized dictatorship *extensive-form description of f for player i is an extensive-form description with some layer $k \in \{1, \ldots, L\}$ such that: (a) every query in every vertex preceding layer k is made to some player other than i, and (b) every vertex v in layer k is labeled with some set $\mathtt{M}(v) \subseteq A_i$, such that if v is on the execution path on inputs $t_{-i} \in \mathcal{T}_{-i}$, then $\mathtt{M}(v) = \mathtt{M}_{t_{-i}}$ is the menu of player i given t_{-i} in the social choice function defined by this extensive-form description.*

5 Matching Mechanisms: a Spectrum between Strategic Simplicity and Strategic Complexity

5.1 Simple-to-Grasp Descriptions of Matching Mechanisms

We now instantiate our general framework from Section 4 to capture the general patterns in how matching mechanisms, our main domain of theoretical study, are commonly described. As discussed in the introduction, we identify two crucial properties that the traditional descriptions of all of the matching mechanisms that we study (and many popular others) share. First, they only consider the preferences of applicants once,[17] in a fixed, natural order (from favorite to least favorite). Second, they require very little "bookkeeping" as they run (little more than just the bookkeeping required to remember a single matching).

Definition 5.1.

1. *In a matching environment, an extensive-form description D is* applicant-proposing[18] *if it satisfies the following: For every agent i and every possible execution path through D, let $v_1, v_2, \ldots, v_k$ denote the vertices along the execution path that query agent i. Then, for $j = 1, \ldots, k$, the function $\ell_{v_j} : \mathcal{T}_i \to S(v_j)$ (which decides the next vertex in the execution path) depends only on the jth institution on agent i's preference list (possibly an "empty institution" if their list has fewer than j institutions on it).*

2. *In a matching environment with n applicants and n institutions, an extensive-form description is* applicant-linear *if it is applicant-proposing, and additionally requires memory at most*[19] $\widetilde{\mathcal{O}}(n)$.

Note that the definition of applicant-proposing has no bite if used alone: if no bounds on the memory requirement are enforced, the extensive-form description can simply remember every type fully and output the correct answer. 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

[17] In addition to facilitating easy explanations of the description, the fact that preferences are read at most once makes it far easier to mentally keep track of what "next steps" could be consistent with the calculation so far, as there is no issue of checking whether these "next steps" are consistent with previous queries to the preferences. This is makes the description simple not necessarily from the usual computer-science perspective of simplicity of running the mechanism "ex-post" on given inputs, but rather from a perspective of simplicity of "ex ante" grasping the relation between inputs and outputs of the mechanism before any specific inputs are specified.

[18] 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.

[19] 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.

matching. (For instance, because there are $n! = 2^{O(n \log n)}$ distinct matchings involving n applicants and n institutions.) The name "linear" thus refers to two distinct things: the linear order in which the description reads preferences, and the (nearly) linear amount of bookkeeping needed (i.e., nearly constant bookkeeping per agent).

Each of SD, TTC, and DA are traditionally defined using applicant-linear descriptions:

Observation 5.2. *SD, TTC, and DA each have applicant-linear extensive-form descriptions.*

In this section we will investigate which of these mechanisms also have applicant-linear individualized dictatroship descriptions (and, hence, menu descriptions within a more traditional description).

5.2 Strategic Simplicity of SD and TTC

We start with perhaps the simplest mechanism we consider in this paper, SD. Unsurprisingly, SD is already a "dictatorship description" (without even needing to be "individualized").

Fact 5.3. *For each participant i, SD has an applicant-linear individualized dictatorship description for player i. Moreover, there is a single applicant-linear extensive-form description that constitutes an individualized dictatorship description for each player simultaneously.*

Moving on to Top Trading Cycles (TTC), we find, perhaps surprisingly, that this mechanism has a fairly simple individualized dictatorship description. Unsuprisingly, unlike SD, this description can only work for a single player at a time. We relate this description to the traditional description using the fact that TTC is *independent of the order in which trading cycles are executed*. In particular, it turns out that by delaying the matching of applicant i as long as possible, we can calculate the menu of applicant i. This is illustrated in Figure 3. The resulting individualized dictatorship describes the entire matching, while making strategyproofness clear to applicant i.

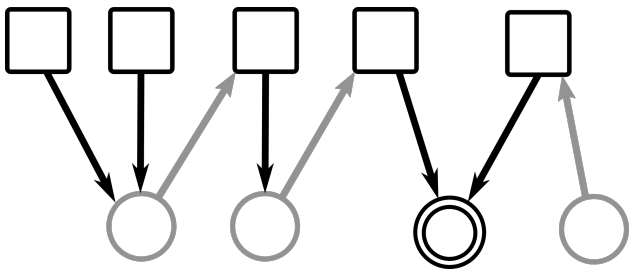


Figure 3: Illustration of the menu calculation in Theorem 5.4. Circles are applicants; squares are institutions. When all possible trading cycles not involving applicant i (the double-circled node) have been performed, applicant i can complete a cycle by pointing to any remaining institution. Thus, every remaining institution is on i's menu.

Theorem 5.4. *For each player i, TTC has an applicant-linear individualized dictatorship description.*

Proof. We will use the fact that by Lemma B.6, TTC (as formally described in Definition B.5) is independent of the order in which we choose to "execute trading cycles"

Start running TTC, and as long as this is possible, choose to only execute cycles that do not include i. This stage of TTC does not depend on i's preferences,[20] and any institution matched during this stage is not on i's menu.

After this, each remaining institution is on i's menu. Indeed, i now points to her favorite remaining institution, and since the next executed cycle *must* involve i, she would get this institution.

Afterwards, continue running TTC as usual (executing 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. □

5.3 Strategic Complexity of DA

Our main impossibility theorem proves that nothing remotely like Theorem 5.4 is possible for DA. Formally:

Theorem 5.5. *For any player i, there exist priorities of the institutions such that no applicant-linear individualized dictatorship description of DA exists. In fact, any individualized dictatorship extensive-form description for DA that is applicant-proposing requires $\Omega(n^2)$ memory.*[21]

One may wonder how this theorem could hold given that, like TTC, the traditional description of DA is independent of the execution order that the mechanism chooses (i.e., the order in which proposals are chosen). To begin to see why this theorem could hold, note that in contrast to TTC, allowing all other players to propose before player i proposes does *not* suffice to calculate the menu. This is because player i proposing to an institution might result in a "rejection cycle" that leads to i being rejected from that institution.

Now, we give some intuition into the proof. The proof constructs (for a fixed "worst case" set of institution priorities) a very large set of applicant preferences such that: (a) to learn the menu of player i, a large fraction of *each other applicant's* preference list must be read, and (b) to correctly compute the final matching, all of the information from these

[20] This can also be thought of as running TTC, but during this first stage not letting participant i point to any institution.

[21] That is, for large enough n, memory is lower-bounded by cn^2 for some constant c that does not depend on n.

applicant's lists must be remembered in full. The $\Omega(n^2)$ lower bound comes from this "large amount of information" that must be remembered.

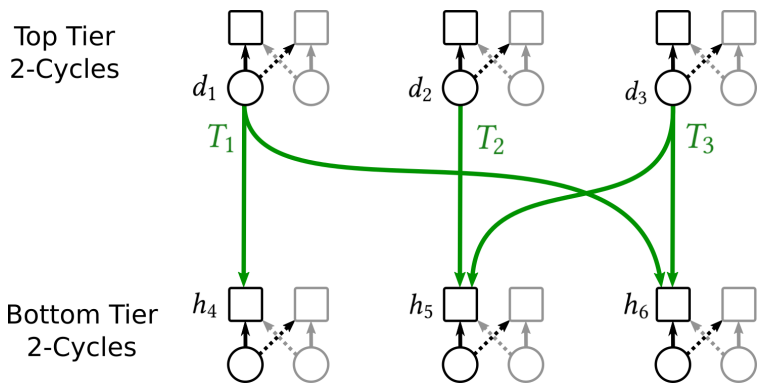


Figure 4: Illustration of the set of preferences witnessing that DA has no applicant-linear individualized dictatorship description. Dark nodes represent d_i or h_i for some i, and grey nodes represent d'_i or h'_i. The green arrows directed outwards from a top-tier d_i represent the sets T_i.

Proof. Fix an applicant d_* and consider any applicant-proposing individualized dictatorship extensive-form description of DA.

Our proof proceeds by constructing a set $\mathcal{S}$ of preferences and priorities (of all applicants and institutions other than d_*) with the following properties:

1. In order to determine the menu of d_*, the entire initial preference lists of some set of applicants $d_1, \ldots, d_{n/4}$ must be read. More precisely, for each $i \in \{1, \ldots, n/4\}$, the list of d_i must be read up to the position of a certain institution h'_i.

2. For each $i = 1, \ldots, n/4$, each of the institutions $h_{n/4+1}, \ldots, h_{n/2}$ may or may not (each, independently) appear on d_i's list (before h'_i appears). Let $T_i \subseteq \{h_{n/4+1}, \ldots, h_{n/2}\}$ denote those institutions which actually appear.

3. Each set T_i (for each $i \in \{1, \ldots, n/4\}$) must be remembered in full in order to correctly compute the final matching (for all possible preference lists of applicants d_*).

Property (1) ensures that for each $i \in \{1, \cdots, n/4\}$ we must read the entire set T_i during the "menu computation" phase of the individualized dictatorship description. Property (3) ensure that the entire set T_i must be remembered by the mechanism. Property (2) ensures that that there are "many" possible distinct ways to set $\{T_i\}_{i=1}^{n/4}$ (indeed, so many that remembering these sets requires $\Omega(n^2)$ bits).

We now describe $\mathcal{S}$ (as 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. For top cycles with $i \in \{1, \ldots, n/4\}$, we have

$$\begin{aligned} h_i: &\quad d'_i \succ d_* \succ d_i \\ h'_i: &\quad d_i \succ d'_i \end{aligned}$$

For bottom cycles with $i \in \{n/4+1, \ldots, n/2\}$, we have

$$\begin{aligned} h_i: &\quad d'_i \succ d_1 \succ d_2 \succ \ldots \succ d_{n/4} \succ d_i \\ h'_i: &\quad d_i \succ d'_i \end{aligned}$$

For the top cycle applicants d_i with $i \in \{1, \ldots, n/4\}$, the preferences vary (we will define *how* they can vary momentarily). Other applicant preference (i.e. those of all applicants d'_i, and those applicants d_i in bottom tier cycles (where $i \in \{n/4+1, \ldots, n/2\}$)) are fixed. Specifically, we have

$$\begin{aligned} d_i: &\quad h_i \succ h'_i \quad \forall i \in \{n/4+1, \ldots, n/2\} \\ d'_i: &\quad h'_i \succ h_i \quad \forall i \in \{1, \ldots, n/2\} \end{aligned}$$

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

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

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

We additionally define a set of inputs $\mathcal{S}' \supseteq \mathcal{S}$. Specifically, let $\mathcal{S}'$ denote $\mathcal{S}$, where additionally any top cycle applicant d_i may truncate the final institution h'_i off her list[22]. We now proceed with two lemmas formally establishing properties (1) and (3) of $\mathcal{S}$ (property (2) holds by the definition of the possible preferences of applicants d_i with $i \in \{1, \cdots, n/4\}$).

Lemma 5.6. *In order to correctly compute the menu of d_* on $\mathcal{S}'$, any applicant-proposing mechanism must query the entire preference list of each top cycle applicant d_i (up to the position of h'_i).*

Proof. Consider the rejection chain that occurs when d_* submits a list containing only of

[22] This collection of preferences can also be constructed with full preference list by adding some unmatched "dummy institution" to represent truncating d_i's list.

h_i. First, d_i is rejected, then she proposes to every institution $h_j \in T_i$. This "rotates" the bottom-tier 2-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 such h_j.

Finally, after getting rejected from each institution in T_i, d_i may or may not propose to h'_i. If they do not, then h_i will be on d_*'s menu. If they do, then h'_i will reject d'_i, who will propose to h_i, who will reject d_*, so h_i will not be on d_*'s menu.

Thus, any applicant-proposing mechanism must read all of d_i's list before being able to correctly output the menu of d_*. □

This proves that, at the moment when the extensive-form description presents the menu to d_*, it has already queried the entire set T_i from each applicant d_i. Thus, the description cannot access this information (unless it stored it in memory) again during the "remainder" phase of computation.

Lemma 5.7. *For each distinct set of preference profiles t_{-*} in $\mathcal{S}$, the induced function $DA(\cdot, t_{-*}) : \mathcal{T}_* \to A$ from types of d_* to matchings is distinct.*

Proof. Consider two instances of preference profiles in $\mathcal{S}$, one profile t_{-*} defined by $\{T_i\}_{i\in\{1,\dots,n/4\}}$, and the other set 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 "eliminated", 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 function from $\mathcal{T}_*$ to outcomes. □

Thus, at the moment any applicant-proposing individualized dictatorship extensive-form description 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 many states, and thus uses space $\Omega(n^2)$. □

Theorem 5.5 is the main impossibility theorem of our paper: we believe it gives a strong and precise sense in which it is hard to infer from traditional descriptions that DA has a menu (equivalent to strategyproofness). The theorem says that the following three requirements are not compatible for any description of the matching of DA: applicant-proposing, "reasonable" memory requirements, and individualized dictatorship. The theorem is tight in the sense that none of these three assumptions can be dropped. First, the traditional description

is applicant-proposing and requires low memory, but does not compute the menu. Second, *any* function of the types can be computed with an applicant-proposing algorithm description with $\widetilde{O}(n^2)$ memory, by simply memorizing the entire input (thus, our $\Omega(n^2)$ lower bound is as high as possible, up to logarithmic factors). Finally, we can show that an $\widetilde{O}(n)$ memory individualized dictatorship description of DA exists that makes *just two* passes through the preference lists of all players (one pass before, and one pass after the menu is computed).[23]

All told, there is a stark contrast between SD and TTC on the one hand, and DA on the other. SD, as normally described, "already" witnesses its strategyproofness, simultaneously for all players. For TTC, we had to work some and tweak the procedure to demonstrate strategyproofness to the players individually, but once this is done, strategyproofness is easy to see. For DA, on the other hand, nothing remotely close to its traditional description of DA suffices for witnessing the strategyproofness of DA through menus.

5.4 The "Source" of Simplicity in TTC

Our contrasting results for TTC and DA may seem mysterious at first. After all, both mechanisms are fairly complicated—where does the strategic simplicity of TTC "come from"? To probe further into this question, we consider a variant of the notion of an individualized dictatorship description, in which t_{-i} cannot be accessed during Step (3) (the step that calculates the matches of participants other than i). This restricted definition may actually be a more natural first attempt at clarifying the relationship between one player's menu and the overall outcome, as it does not contain two phases making use of t_{-i}.

Can a preference-linear description meeting this restricted definition still calculate both the menu and the matching in TTC? We prove that it cannot. In fact, we show that any such description of TTC would require memory $\Omega(n^2)$. In particular, this shows that the "pointing graph" in the proof of Theorem 5.4 just before participant i is matched does not contain *nearly enough* information to calculate the rest of the matching. We formulate the precise meaning of this result (and investigate the analagous questions for SD and DA) in Appendix D. Very briefly, this result holds because when participant i in TTC selects an item from their menu (in Theorem 5.4), this choice can dramatically affect the rest of the run of TTC (in particular, affecting the *order* in which participants will be matched). Thus, our individualized dictatorship calculating TTC cannot be further simplified (by avoiding accessing t_{-i} in Step (3)), and thus very precisely captures what is strategically simple about TTC.

[23] Towards this goal, we show in Section E.2 that an applicant-linear *menu-only* description of DA exists. This description can thus be used to compute the menu of one player, and then the process can "restart" and use the traditional description of DA to compute the rest of the matching.

Table 3: Our full classification of $\widetilde{O}(n)$-memory applicant-proposing and institution-proposing algorithms for (applicant-optimal) DA, which compute the menu, compute the matching, or compute both the menu and the matching in an individualized dictatorship.

	Menu (exposes strategyroofness)	Matching (exposes feasibility)	Individualized Dictatorship (exposes both)
Applicant proposing	In Section E.2. Necessarily **non-local** by Theorem 6.2.	Traditional DA algorithm	**Completely impossible** by Theorem 5.5.
Institution proposing	Description 1 (Section 3)	In Section E.1 / Ashlagi et al. (2017). Necessarily **non-local** by Theorem 6.3.	In Section E.3. Necessarily **non-local** by Theorem 6.3.

6 The Landscape of Menu Descriptions of DA

We proved that a menu description for DA cannot be found "within" traditional, applicant-proposing descriptions (which calculate the full matching). One may then ask whether any nontrivial applicant-proposing description can calculate the menu in DA *whatsoever* (without regard to calculating the final matching). We found such a description. Unfortunately, it is *very* difficult to understand. Informally, it works by ignoring applicant i's list and having applicant i "try proposing to every institution" (conceptually, as in Example 2.4), while performing careful bookkeeping to avoid looping through preference lists multiple times. While the resulting algorithm has nearly the same memory requirements as traditional description of DA, this hard-to-understand and convoluted description does not provide a practical alternative to traditional explanations of DA.

Alternatively, one may consider extending the institution-linear[24] Description 1 to calculate the rest of the matching. In other words, one might ask whether an institution-proposing individualized dictatorship description exists for DA. In perhaps one of the most mathematically surprising results of this paper, we construct such a description. This description very carefully merges Description 1 as a "preceding step" before running an institution-proposing algorithm for (applicant-optimal) DA from Ashlagi et al. (2017), conducting very careful bookkeeping to not violate applicant-linearity. While the result gives some evidence that

[24]While we have not formally defined institution-proposing and institution-linear descriptions, whenever we use any of these terms we mean the analogous definition to Definition 5.1, in which sides are flipped.

the algorithm of Ashlagi et al. (2017) can be extended so it is somewhat easier to recognize as strategyproof (compared to traditional applicant-proposing DA, which by Theorem 5.5 cannot be extended in such a way), the resulting algorithm is *very* convoluted and hard to construct.

We construct both of these convoluted descriptions—the applicant-proposing menu description and the institution-proposing individualized dictatorship that we found—in Appendix E. The fact that these two descriptions are convoluted, though, still does not immediately imply that there are no other, more attractive, such descriptions. Our final set of impossibility results provide a negative answer to this hope, showing that any applicant-proposing menu description of DA and any institution-proposing description of the full DA outcome must be convoluted in a precise sense that we term *non-locality*: it must update bookkeeping concerning some participants when making queries that are unrelated to them. Since applicants proposing implies that any menu description will be convoluted, and institutions proposing implies that any outcome description will be convoluted, we conclude that any individualized dictatorship, whether applicant- or institution-proposing (regardless even what its memory requirements are), must be convoluted as it contains both a menu description and an outcome description. In Table 3, we overview all our descriptions and impossibility results for DA.

More technically, we call an applicant-proposing description of a menu of an applicant or of a matching *local* if (a) in additional to any "global" bookkeeping, it also maintains "local" bookkeeping for each institution that is only updated following that institution being read from any applicant's list, and (b) if an institution's information in the outcome (i.e., whether this institution is on the menu if the description is of a menu, or the match of this institution if the description is of a matching) only depends on the final state of the local bookkeeping for this institution when the execution concludes. The definition of locality for institution-proposing descriptions is similar.

Definition 6.1. *In a matching environment, an applicant-proposing extensive-form description D, of either the menu of some player $\tilde{\imath}$ or of the matching, is* local *if every vertex v in D is labeled by some label $\big(L_1(v), \ldots, L_n(v)\big)$ such that both of the following hold for every vertex v in D:*

- *If v is an internal vertex, then let i be the participant queried in v. Since D is applicant-proposing, recall that the function $\ell_v : \mathcal{T}_i \to S(v)$ depends only on the jth institution (possibly an "empty institution") on i's preference list, for some j. Abusing notation, we therefore consider ℓ_{v_j} to be a function from institutions to $S(v_j)$. Then, for every institution k, the labels of v and of $\ell_v(k)$ may only differ in their kth coordinate, L_k.*

- *If v is a sink vertex, then for every institution k, the determination for the outcome of v of whether k is on $\tilde{\imath}$'s menu (if the description is of $\tilde{\imath}$'s menu) or of with whom k is matched (if the description is of a matching) depends only on $L_k(v)$.*

The definition of locality of an institution-proposing extensive-form description D is completely analogous, flipping sides.

Both the traditional applicant-proposing description of the DA matching, and Description 1 are local under an appropriate definition.

We can now state the two parts of our impossibility result.

Theorem 6.2. *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 description of applicant i's DA menu is non-local.*

Theorem 6.3. *If there are at least four applicants and three institutions, then there exist preferences of the applicants such that any institution-proposing description of the DA outcome matching is non-local.*

Corollary 6.4. *If there are at least four applicant and three institutions, any one-side-proposing individualized dictatorship description of DA is non-local.*

Among all of the possible description types in Table 3, only two exist and are non-local: the traditional description of DA (which conveys feasibility but not strategyproofness) and our menu description—Description 1 (which conveys strategyproofness but not feasibility). For simple, practical descriptions of DA, 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. Description 1 is at the other corner solution.

7 Menu Descriptions of Auctions

As a second broad application of our theoretical framework, we explore the possibility of simple menu descriptions for multi-item auction. For particularly simple multi-item auctions, menu descriptions can be given via a separate, simple menu description regarding each item separately. We first formalize what this means.

Definition 7.1.

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 execution*

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 function ℓ_v (which decides the next vertex in the execution path) only depends on some bidder's value for item j, and (b) for each vertex w outside of this interval, the function ℓ_w does not *depend on any bidder's value for j.*

2. *An* item-linear *description is an item-read-once description with memory requirement at most $\widetilde{O}(m)$.*

Exactly as discussed in Section 5.1 for matching, the bidder-read-once definition only has bite if additionally complemented with restrictions on the memory used by the algorithm. As for item-linear auctions, the memory requirement of $\widetilde{O}(m)$ is (up to logarithmic factors) the memory required to write down a "tentative allocation" of the items, or a single price for each item.

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

Theorem 7.2. *The VCG auction with additive bidders has an item-linear description. For any bidder i, it also has an item-linear menu description for bidder i.*

Proof. The item-linear description of the auction simply goes through the items one by one, and queries each bidder for their value, while keeping track of the highest bidder (and their bid) and of the second-highest bid. The menu description is even simpler: it keeps track of only the highest bid on each item. □

Thus, the situation for additive auctions is similar to that of TTC—there is a menu description which is a fairly simple modification to the traditional description, specializing the order of steps to compute the menu of player i.[25] Moreover, this shows how the mechanism's strategyproofness can be understood in terms of the (separate) strategyproofness of m separate single item auctions. This demonstrates a general property of our framework: allowing for the "composition" of simple menu descriptions when players are additive over

[25]One could more formally consider the prospect of an individualized dictatorship for additive auctions. This is possible, with a modification of Definition 7.1 that would allow queries to bidder i (the player whose menu is described) to come after all other queries. In detail, consider a description that goes item-by-item and bidder-by-bidder, and remembers for each item the highest bid b_1, the bidder who placed this bid, and the second highest bid b_2. Then, if player i's bid is b, there are three cases to determine the outcome on that item: if $b > b_1$, bidder i wins and pays b_1; if $b_1 \geq b \geq b_2$, then the highest bidder wins and pays b; if $b_2 \geq b$, then the highest bidder wins and pays b_1. Note, however, a fine-grained distinction between this description and our description of TTC: this description does not need to access t_{-i} during "Step (3)," which determines the allocation to bidders other than i, a facet that we proved unavoidable for TTC (as discussed in Section 5.4).

the "sub-mechanisms."[26]

When bidders have unit demand over items, concrete descriptions become harder—even more than for DA in a sense. This holds both for menu descriptions and for descriptions of the outcome, which we prove in Appendix C.

Theorem 7.3. *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 description (computing the allocation), and for menu descriptions (computing one player's menu).*

8 Experiment

Menu descriptions offer a format for describing strategyproof mechanisms via the same, simple proof outline. But do real people recognize strategyproofness better under menu descriptions? Our experiment tests this question in two simple mechanism design settings.

8.1 Experiment Flow

Our experiment consists of two parts corresponding to two different mechanisms. The mechanisms are, in order: median voting in an election with three single-peaked voters (henceforth, Median) and bidding in a single-item second-price auction with five bidders (henceforth, Auction). In each part, respondents are independently randomized into either Traditional (T) or Menu (M) treatments, resulting in four treatment combinations: TT, TM, MT, and MM. Treatments T and M differ from each other only in the way the mechanism is described. That is, the difference between the treatments is only in the language describing the mapping from reports to outcomes (and in the examples[27] and comprehension questions that accompany the description).

After an informed-consent and an 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 supplemental material for screenshots. The instructions screen describes the mechanism and how participants earn money, and the practice rounds illustrate how the participants report to the mechanism. The final setup screen reinforces the description and confirms understanding. Respondents

[26] 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 additive buyers Bade and Gonczarowski (2017).

[27] In all the examples, the player reports are identical across the two treatments, but the language is adjusted to the treatment. In Median, the examples are also illustrated visually.

who fail to correctly answer a comprehension question within three attempts are given the answer. They must enter it to proceed.[28]

The setup screens work together to convey (and confirm understanding of) the working of the mechanism, that is, the way that the reports of the participant (and the other randomized reports) determine the outcome and the earnings for the participant. Because our goal is to test what strategic properties the participants infer from the descriptions of the mechanism alone, the materials are careful to not give any form of strategic advice.

After completing the setup screens, participants participate in ten rounds of voting (in Median) or bidding (in Auction). In each round, participants see their private information (their single-peaked value table in Median and a private value in Auction)[29] and have 30 seconds to submit their report. They then see the result of that round, including their earnings. On each screen, 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).

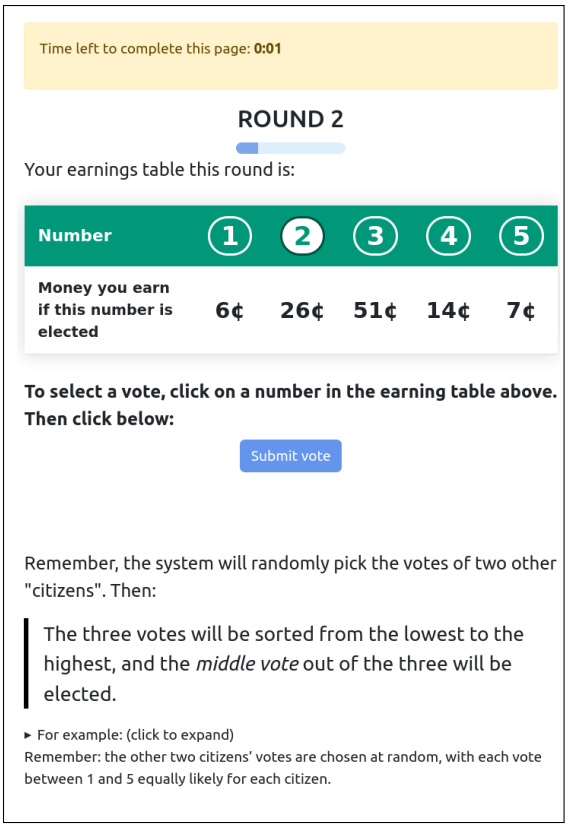
After completing Median and Auction, respondents fill out an exit questionnaire that includes demographic questions, an informal numeracy test, and (for each mechanism) open-ended questions on their understanding, strategies, and thoughts. They are paid the sum of their earnings in all rounds. The experiment is programmed on Otree (Chen et al., 2016).

8.2 Traditional (T) and Menu (M) Treatments

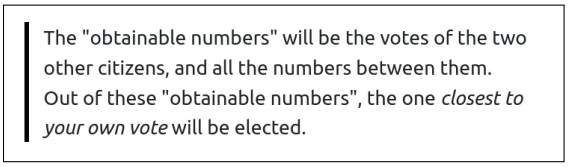
Figure 5(a) reproduces the main screen for a Median round in T. Participants vote for one of five candidate numbers (labeled 1–5) by clicking on their chosen number. The only difference between T and M is the larger-font text to the right of the thick black vertical line (and the way the outcome is described in the examples—which the participant can see again if they click an expandable link—and comprehension questions). Its M version is shown below in Figure 5(b). The main round screen for Auction in T (and its changed text in M) is presented in Figure 6. In Auction, participants pick a bid between \$0.00 and \$5.00.

[28] In Median, there are six total comprehension questions concerning three examples. In T and M, respectively, 88 and 68 percent of participants answer them all correctly on first attempt; 98 and 93 percent make at most six mistakes (one per question, on average). In Auction, there are twelve questions, of which six are Yes/No. In T and M, 63 and 59 percent answer them all with no mistakes, and 97 and 89 with at most six mistakes. For additional analysis of these comprehension questions, see Appendix A.

[29] For both mechanisms, the distribution which the private information is drawn from is tailored to provide a consistent and decently high expected monetary reward for straightforward behavior—see the supplemental material for a description of these distributions.

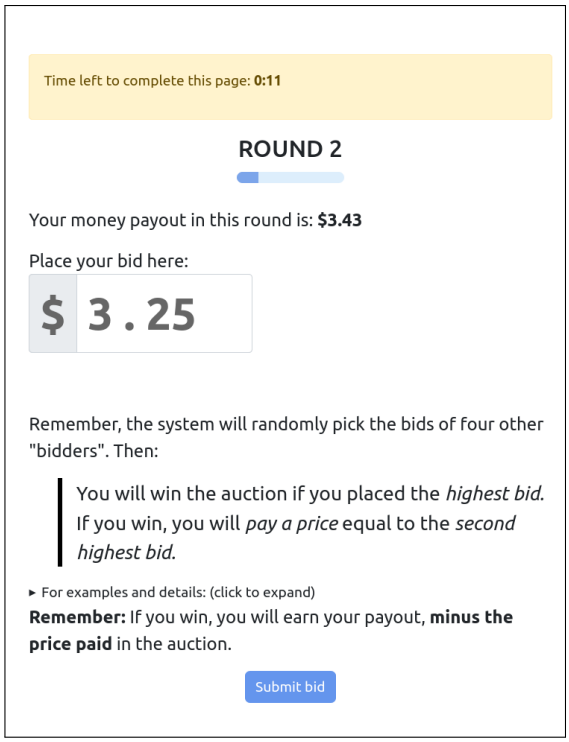


(a) T treatment: entire screen

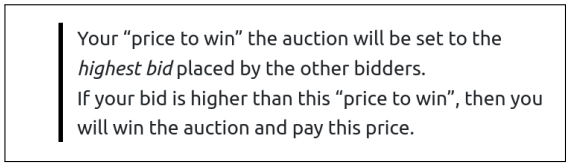


(b) M treatment: changed text

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



(a) T treatment: entire screen



(b) M treatment: changed text

Figure 6: Auction round screen. The participant has entered a bid of $3.25, but has not yet submitted it.

8.3 Data

We conducted our experiment on February 4–5, 2022. We recruited US 18+ participants on the Prolific platform (https://www.prolific.co). 226 participants started the experiment and progressed beyond the informed-consent screen; 200 completed it.[30] 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.

	Median Voting			**Second Price Auction**		
	Trad. (T) ($N = 100$)	Menu (M) ($N = 100$)	p-value	Trad. (T) ($N = 100$)	Menu (M) ($N = 100$)	p-value
S.F. (fraction):	0.70 (0.03)	0.80 (0.03)	0.01	0.37 (0.04)	0.34 (0.03)	0.55
S.F. (all rounds):	0.26 (0.04)	0.53 (0.05)	0.0003	0.13 (0.03)	0.10 (0.03)	0.66
Earning ($):	2.83 (0.07)	3.00 (0.07)	0.10	3.40 (0.26)	3.40 (0.24)	0.98

Table 4: Experiment results by treatment. "S.F. (fraction)" is the average fraction of rounds (across all participants) with straightforward behavior. In Median, S.F. means voting for the ideal number. In Auction, S.F. means bidding within $0.10 of the private value (out of a possible bidding range from $0.00-$5.00). "S.F. (all rounds)" is the proportion of participants who play straightforwardly (as per these definitions) in all ten rounds. "Earnings" are participants' average dollar amount earned across all rounds. Standard errors are in parentheses. p-values for "S.F. (fraction)" and "Earnings": two-sample, two-sided equality-of-means t-test (Welch's t-test); p-values for "S.F. (all rounds)": two-sample, two-sided equality of proportions test.

8.4 Results

Table 4 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. This difference is arguably large; a two-sided t-test for equality of means yields a p-value of 0.01. The difference is larger still when comparing the fraction of participants who play straightforwardly *in every round*: 26 percent in T versus 53 percent in M, $p = 0.0003$; see Figure 7 for the full distributions. Participants also earn 6 percent more in M ($3.00) than in T ($2.83),

[30] Of the incompletes, 6 dropped on the introductory screen before any treatment began, 16 dropped during Median setup screens (5 in T, 11 in M), and 4 during Auction (3 in T, 1 in M). Our (pre-registered) intention was actually 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; the Median t-test p-value reported in Table 4 changes from 0.01 to 0.02. See Appendix A for additional discussion of the dropouts.

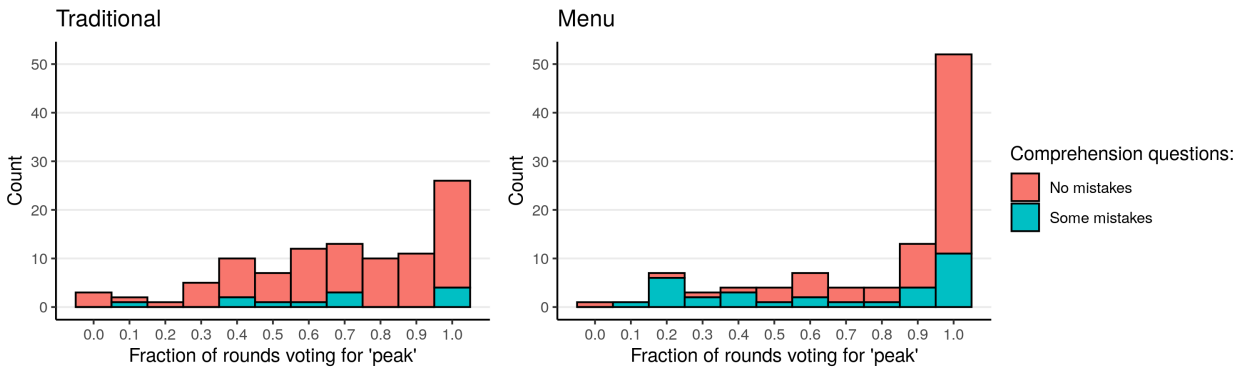


Figure 7: Median: Histograms of the fraction of rounds where participants in each treatment played straightforwardly. Each bar of the histogram is sub-divided to show which participants in that bar had mistakes in the comprehension questions.

Treatment	All correct first try	At least one mistake	p-value	# answered wrong: 1	2	3
Traditional	0.70 (0.03) $N = 88$	0.67 (0.08) $N = 12$	0.79	0.66 (0.11) $N = 7$	0.5 (0.09) $N = 3$	1.0 (N/A) $N = 2$
Menu	0.87 (0.03) $N = 68$	0.65 (0.06) $N = 32$	0.002	0.69 (0.11) $N = 10$	0.67 (0.08) $N = 17$	0.50 (0.14) $N = 5$

Table 5: The average rate of straightforward play in Median, conditioned on treatment and on how the participant fared on the comprehension questions. The p-value (two sided, Welch's test) is reported for the difference (within each treatment) between those participants answering all questions on their first try and those who do not. 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 with two questions) out of three where they made a mistake.

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.[31] In both mechanisms, comparing first to last five rounds to explore learning, fatigue, or boredom, we find essentially no difference.

To further investigate our strong results in Median, Figure 7 and Table 5 report on the relationship between straightforward play and participants' understanding of the description, as measured by the comprehension questions. Under M, answering all comprehension questions correctly on first attempt is associated with a significant increase in straightforward play (87 percent in the "All correct on first try" column versus 65 percent in the "At least one mistake" column, $p = 0.002$). In contrast, under T, we find no such difference (70 versus 67 percent, $p = 0.79$). At the same time, the comprehension questions seem harder to correctly answer on first attempt in M than in T ($N = 68$ versus 88 participants, respectively, in the "All correct on first try" column), but straightforward play is rather similar among participants with at least one mistake in M ($N = 32$, 65 percent) and participants in T—either those with ($N = 12$, 67 percent) or without mistakes ($N = 88$, 70 percent).

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. Specifically, we noticed that several participants mistakenly write their own bid (which is given to them in the question) as the price one 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 our theory does not shed light on. For additional discussion (and samples of the comprehension pages), see Appendix A.

Overall, the results in this section suggest that while our first attempt at designing new menu descriptions still needs work, it shows promise. On the one hand, in both Median and Auction, our menu descriptions are conspicuously longer, and appear more difficult to grasp, than traditional descriptions. On the other hand, our menu descriptions do not meaningfully reduce average straightforward play in Auction, and significantly increase it in Median.[32] Furthermore, in Median, our menu description does not on average seem to

[31] The figure displays the results for Auction with a stringent definition of straightforward play, resulting in low S.F. rates. See Appendix A for more lenient definitions, which result in S.F. rates closer in magnitude to those in Median.

[32] One hypothesis, yet untested, as to why we find a difference across M and T in Median and but not in Auction is that in Auction, M and T are simply too similar. Indeed, they essentially just switch the order of

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.

9 Related work

A major cornerstone of the contemporary "strategic simplicity" program in mechanism design is Li (2017), which introduces OSP mechanisms to formally explain why certain mechanisms implementing a social choice rule are easier to play than others. In an empirical follow-up work, Breitmoser and Schweighofer-Kodritsch (2019) show that even *describing* or *framing* a social choice rule as an OSP mechanism—even if it is implemented as a direct mechanism—can improve truth-telling rates. Alas, many desirable social choice rules do not have OSP mechanisms. This is the case for TTC (Li, 2017) and for DA (Ashlagi and Gonczarowski, 2018) (except in rare special cases of institutions' priorities, Troyan, 2019; Thomas, 2021), and even for Median and for two-item auctions for bidders with additive valuations (Bade and Gonczarowski, 2016), which greatly limits the applicability of this approach.

Our paper can be seen as contributing to this line of work by noting that once a social choice rule is described as being run via a certain extensive-form mechanism but is actually run as a direct mechanism, there is no reason not to describe it as a *different* extensive-form mechanism to each participant. Because a description naturally only needs to explain strategyproofness to one agent (the agent reading the description), it is enough to describe it as a mechanism that is OSP only for that agent. This ability to target only one agent fortunately brings the entire class of strategyproof social choice rules (not just those that are OSP implementable) back under our scope, and menu descriptions in fact describe any strategyproof mechanism to one agent at a time as a mechanism that is *strongly* obviously strategyproof (a concept defined by Pycia and Troyan, 2021) for that agent. Correspondingly, we achieve "strategically simple" descriptions of Median, TTC, and even DA. Note, however, an important difference between our framework and earlier ones: While any OSP mechanism provides a "simple to play" game, not every menu description provides a simple, strategyproofness-exposing description. Instead, "non-simple" (e.g., brute force) menu descriptions trivially exist for every strategyproof mechanism, and finding different menu descriptions and quantifying their simplicity can be a difficult (and domain-specific) theoretical task.

The menu of a player in a mechanism is different from a player's "budget set" (Segal, 2007), a notion that was applied to the study of matching markets (Azevedo and Leshno,

two sentences and fix the grammar accordingly. In contrast, in Median there is a genuine algorithmic change.

2016; Leshno and Lo, 2021; Immorlica et al., 2020).[33] However, there is some high-level conceptual relation: where a menu demonstrate, ex-ante, the strategyproofness of a rule, the budget sets help to verify, ex-post, that the choice rule worked as promised. Budget sets are used in these papers to calculate welfare in continuum markets; Leshno and Lo (2021) also mentions the fact that in these *continuum* markets, the budget set is independent of the player's report, which implies strategyproofness. Still, they do not provide an ex-ante way of seeing strategyproofness from any concrete, *finite* market, mechanism descriptions (in which a player's budget set may depend on her report). For DA, the menu is also a distinct notion from the set of stable partners of an applicant. And, because an applicant's report can affect their set of stable partners, this set cannot help to explain strategyproofness in the the applicant's menu can.

Our theoretical analysis, to the best of our knowledge, is the first to analyze different ways to describe multi-player mechanisms in terms of menus. There is some empirical overlap with the recent working paper Katuščák and Kittsteiner (2020), which suggests describing a matching mechanism (specifically, TTC) via menus. They use a single menu description that is very close to that of Example 2.4, which essentially calculates the menu by iterating over possible reports and running the standard TTC description for each report to see which outcomes are possible.[34] While their work suggests menu descriptions as a promising direction for research, we believe that menu descriptions will not be practical unless they are (nearly) as natural and easy-to-understand as traditional descriptions. Thus, our main focus is in finding menu descriptions that do not sacrifice understandability, and in formalizing the trade-offs between exposing different properties of mechanisms. We are not aware of any additional papers that empirically explore different, equivalent ways of explicitly describing the way a (direct) mechanism works, without changing the feedback or information presented to the participants. Further from us are empirical papers that explore *advice* that explicitly informs participants of the strategyproofness of the mechanisms, such as Masuda et al. (2020) (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

[33] For the specific mechanism of DA, the budget set of applicant i (which depends on t_i and t_{-i}) is 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. In finite markets, the menu in DA is not given in a simple way by the 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.

[34] Katuščák and Kittsteiner (2020) prove three theoretical results: one equivalent to the result of Hammond (1979) (i.e., Theorem 2.3), one essentially equivalent to Example 2.4, and one motivating the use of menu descriptions in terms of "partition obvious dominance" (Zhang and Levin, 2017).

exact workings of the mechanism.

There is a large literature within computer science looking at the menu in single-agent 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, 2013; Daskalakis et al., 2017; Babaioff et al., 2017; Saxena et al., 2018; Gonczarowski, 2018). Most relatedly within computer science is Dobzinski (2016) and follow-up works, which consider the properties of menus in multi-bidder auctions in some detail, and use menus as a tool for analyzing mechanisms and bounding communication complexity. Because our viewpoint lies on defining (instead of analyzing) mechanisms in terms of menus, many technical distinctions arise. For instance, in Appendix D we show that even a substantial generalization of the "taxation complexity" framework of this literature cannot separate TTC and DA, in contrast with our separation result from Section 5. We are not aware of any prior algorithmic work on finding the menu in matching mechanisms such as DA, for which we give multiple optimal (in terms of complexity) algorithms, as well as impossibility results.

10 Conclusion

Strategyproofness is often proposed as a way to make mechanisms fair by theoretically leveling the playing field for players who do not strategize well. However, if disparities remain in participants' *understanding* of strategyproofness, then the participants remain on uneven footing. These disparities may be hard to avoid. After all, strategyproof mechanisms have hard-won properties that most people are not familiar with. A second price auction is *not* a real-life haggling process where the bidder can influence their price, and deferred acceptance is *not* a real-life job hunt where one can miss out on reasonable choices because of time wasted applying to reach choices. Menu descriptions give a consistent way of conveying the crucial strategyproofness property, and may have the potential to further even out many playing fields through an improved understanding of real-world mechanisms.

In this paper, we propose an outline for defining mechanisms that enables strategyproofness to be immediately proven. We also experimentally investigate the degree to which real people see strategyproofness for themselves. Still, our theory provides no actual behavioral models of how participants make decisions. Thus, quantitative relationships between the "mental difficulty" of finding one's dominant strategy and the rates of playing this dominant strategy remain an intriguing open direction. Additionally, our theory provides no guidance on what to do if, e.g., due to issues of trust, the description of the mechanism cannot be

specialized to the individual reading the description.[35] A more general theory on how easy it is to derive strategyproofness from a description thus remains another open direction.

Even if a menu description is quite complex, it can be easy to observe its strategyproofness so long as it is clear that player i's report won't be used to calculate her menu. Our simplicity conditions are designed to guide us along a different axis—not the ease of seeing strategyproofness, but the ease of understanding the workings of the description itself. Future work may examine how these axes interact. One natural idea is for the description to simply state that "some menu will be calculated using only other players' types," without specifying any details. Note, however, that beyond strategyproofness, this approach conveys no information *at all* about what the mechanism actually is. Still, it remains an interesting question to investigate how the menu might be used to simply define strategyproofness to real-world participants, or to give them "strategic advice." We are currently developing such experiments, along with an experiment on our new menu description of DA.

As people's daily lives become increasingly dominated by algorithms, different mechanism properties—from fairness to feasibility—may become important to emphasize and explain. We study a way to expose strategyproofness in mechanisms (and how it interacts with exposing feasibility). Future studies may discover simple descriptions, of mechanisms as well as other algorithms, that expose additional important properties.

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.

I. Ashlagi and Y. A. Gonczarowski. Stable matching mechanisms are not obviously strategyproof. *Journal of Economic Theory*, 177:405–425, 2018. Originally appeared in 2015.

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. URL https://EconPapers.repec.org/RePEc:ucp:jpolec:doi:10.1086/689869.

[35] Though it is not precisely clear under what situation there would be sufficient trust in that a single given description properly describes the mechanism being run (trust that is required in essentially any centralized matching clearinghouse), but not in that given individualized descriptions describe it.

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

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. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 869–877, 2017. doi: 10.1145/3055399.3055426. URL http://doi.acm.org/10.1145/3055399.3055426.

S. Bade and Y. A. Gonczarowski. Gibbard-satterthwaite success stories and obvious strategyproofness. In *EC*, 2016.

S. Bade and Y. A. Gonczarowski. Gibbard-satterthwaite success stories and obvious strategyproofness. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, EC '17, page 565, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450345279. doi: 10.1145/3033274.3085104. URL https://doi.org/10.1145/3033274.3085104.

I. Bó and R. Hakimov. Pick-an-object mechanisms. *Available at SSRN*, 2020.

Y. Breitmoser and S. Schweighofer-Kodritsch. Obviousness Around the Clock. Rationality and Competition Discussion Paper Series 151, CRC TRR 190 Rationality and Competition, Apr. 2019. URL https://ideas.repec.org/p/rco/dpaper/151.html.

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 *Symposium on Simplicity in Algorithms (SOSA)*, pages 257–267. SIAM, 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. doi: 10.1016/j.jbef.2015.12.00. URL https://ideas.repec.org/a/eee/beexfi/v9y2016icp88-97.html.

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. Technical report, 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 *FOCS*, 2016.

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*, 2022.

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

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 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 123–131, 2018. doi: 10.1145/3188745.3188786. URL http://doi.acm.org/10.1145/3188745.3188786.

R. Guesnerie. On taxation and incentives: further remarks on the limits to redistribution. Discussion Paper 89, University of Bonn, 1981.

R. Guesnerie. *A contribution to the pure theory of taxation*. Number 25 in Econometric Society Monographs. Cambridge University Press, 1998.

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.

D. Gusfield and R. Irving. *The stable marriage problem: Structure and algorithms*. The MIT Press, Cambridge, MA, 1989. ISBN 9780262071185.

R. Hakimov and D. Kübler. Experiments on matching markets: A survey. Technical report, WZB Discussion Paper, 2019.

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

S. Hart and N. Nisan. The menu-size complexity of auctions. In *the 14th ACM Conference on Electronic Commerce (EC)*, 2013.

S. Hart and N. Nisan. Approximate revenue maximization with multiple items. *J. Economic Theory*, 172:313–347, 2017. doi: 10.1016/j.jet.2017.09.001. URL https://doi.org/10.1016/j.jet.2017.09.001.

A. Hassidim, D. Marciano, A. Romm, and R. I. Shorrer. The mechanism is truthful, why aren't you? *American Economic Review*, 107(5):220–24, 2017.

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 Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 53–62, 2005. URL http://dl.acm.org/citation.cfm?id=1070432.1070441.

N. Immorlica, J. Leshno, I. Lo, and B. Lucier. Information acquisition in matching markets: The role of price discovery. *Available at SSRN 3705049*, 2020.

J. H. Kagel and D. Levin. Independent private value auctions: Bidder behaviour in first-, second-and third-price auctions with varying numbers of bidders. *The 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. *The 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. *Available at SSRN 3521257*, 2020.

T. Masuda, T. Sakai, S. Serizawa, and T. Wakayama. A strategy-proof mechanism should be announced to be strategy-proof: An experiment for the vickrey auction. Discussion Paper 1048, ISER, Osaka University, 2019.

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. Discussion Paper 1109, ISER, Osaka University, 2020.

M. Pycia and P. Troyan. A theory of simplicity in games and mechanism design. *CEPR Discussion Paper No. DP14043*, 2021. URL AvailableatSSRN:https://ssrn.com/abstract=2853563.

A. Rees-Jones. Suboptimal behavior in strategy-proof mechanisms: Evidence from the residency match. *Games and Economic Behavior*, 108(C):317–330, 2018. doi: 10.1016/j.geb.2017.04.011. URL https://ideas.repec.org/a/eee/gamebe/v108y2018icp317-330.html.

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

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.

R. R. Saxena, A. Schvartzman, and S. M. Weinberg. The menu complexity of "one-and-a-half-dimensional" mechanism design. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2026–2035, 2018. doi: 10.1137/1.9781611975031.132. URL https://doi.org/10.1137/1.9781611975031.132.

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

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

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

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

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

L. Zhang and D. Levin. Bounded rationality and robust mechanism design: An axiomatic approach. *American Economic Review*, 107(5):235–39, 2017.

A Additional Experimental Analysis

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

A.1 Median

What do participants vote for? First, one can ask: *when participants do not play straightforwardly, what strategies are they following?* Figure 8 gives a full table of how votes were placed, conditioned on the location of the peak in the earnings table. First, we observe that there is a clear trend towards voting for numbers *adjacent* (or "close to") the peak. In Traditional, 12% of all "mistakes" (aka voting for the non-peak) are cast for a number more than 1 distance from the peak; in Menu, the same statistic is 14%. If mistakes were uniformly random over all non-peak numbers, this statistic should be 60%. Some, but far from all, of the "mistakes" are 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 peak at 1 or 2 would maximize their utility by voting for 1 (and similarly, all those with peak at 4 or 5 would want to vote for 5). Thus, confusion with an averaging mechanism cannot explain the large fraction of votes which are placed "closer to the middle" 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.

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	–

Figure 8: The marginal tables of voting across all rounds of all participants. When the Ideal Number (the "peak" of the single-peaked earnings table) is not 1 or 5, the fraction of rounds where the participant voted for their *second* highest valued number is also shown. Note that a single participant in the Menu treatment failed to submit a vote within the 30 second time allotted in one round.

Comprehension questions. Now, we investigate *how comprehension of the mechanism may effect behavior.* Samples of the comprehension page are provided in Figure 9. The comprehension questions in the T treatment seemed noticeably easier than in the M treatment—12 of the 100 T participants failed to answer all the comprehension questions on first attempt[36]; the same number is 32 in M[37]. 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.

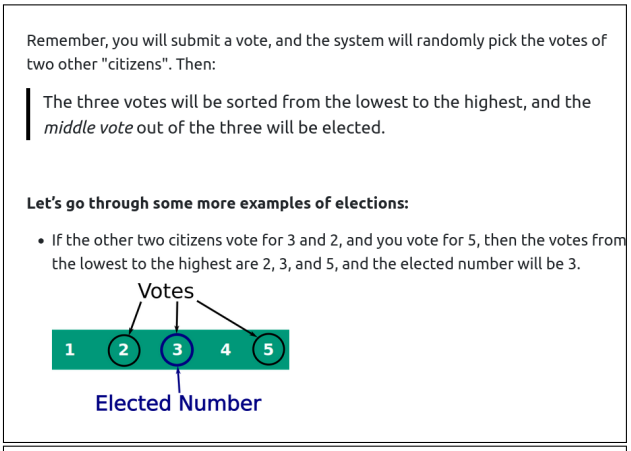
Overall, completing all comprehension questions on the first try is associated with a very sharp and statistically significant increase in the rate of straightforward behavior under the Menu treatment, but not under the Traditional treatment. Table 5 shows these results. On one hand, this may not be exceptionally surprising given that the comprehension questions are far more difficult under Menu, and thus may much 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 suggests that the more complicated menu descriptions may *only be worth using if participants can still understand them.*

Dropout rate. We would also like to address the dropout rate we see in the experiment. 20 participants dropped out after the informed consent and introductory screens; the majority (16) dropped out during the first mechanism, Median. The 4 dropouts from the Auction are not included in our data set, but including them could effect our conclusions (for both Auction and Median) only trivially. On the other hand, for Median there is some concern here that the rate of dropout could be conditional on the treatment—5 of the 16 dropouts have T treatment; 11 have M.[38] Our findings depend on assumptions about the would-be behavior of these dropouts. We can consider what happens if we add back in these 16 dropouts under different assumed proportions of straightforward play. If all dropouts in both T and

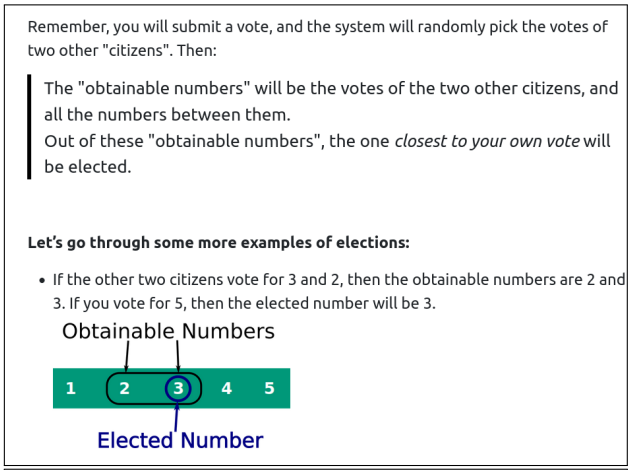
[36]Some (arbitrarily chosen) examples of mistakes in T include: not writing a repeated vote twice; sorting the numbers high-to-low; (seemingly) confusing which bubble corresponded to which elected number; writing the numbers by their listed order instead of their sorted order; or entering the elected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

[37]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.

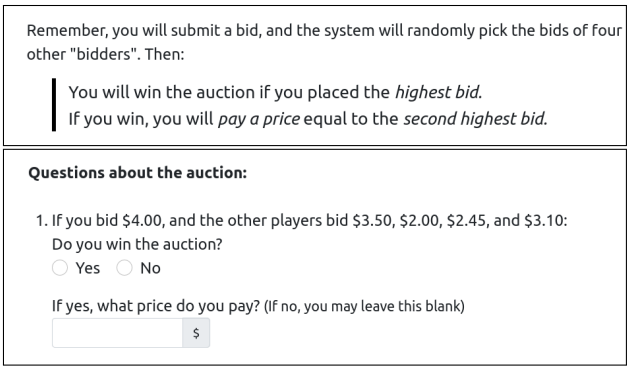
[38]Sub-dividing these 16 participants further, 5 dropped on the first screen describing the mechanism (1 in T, 4 in M), 1 dropped on the practice round (in T), and 10 dropped on the comprehension screen (3 in T, 7 in M).



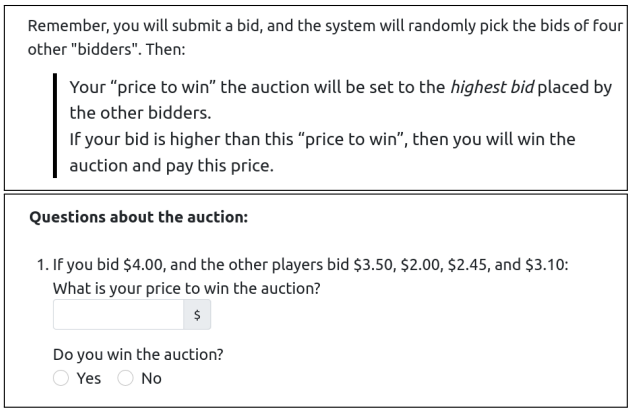
(a) Median mechanism, Traditional (T) treatment



(b) Median mechanism, Menu (M) treatment



(c) Auction mechanism, Traditional (T) treatment



(d) Auction mechanism, Menu (M) treatment

Figure 9: Samples of the comprehension pages of both mechanisms. Each page started with a reminder of the description of the mechanism, then contained several examples, then had comprehension questions. For the full contents of these pages, see the supplemental material.

M play straightforwardly in 2/10 rounds (matching random play), then the average under T is 0.67 and under M is 0.74, with a p-value of 0.11 (two-sided). Under the less-pessimistic assumption that the dropouts play similarly to those participants who *failed to answer the comprehension questions* on their first try (which may be plausible because a large fraction of the dropouts occur on the comprehension screen), the results are less affected. In particular, if all dropouts in T are assumed to play straightforward with proportion 0.67, and in M with proportion 0.65, then the average under T is 0.69 and under M is 0.79 (p = 0.01, two-sided).[39] However, even under worst-case behavior of the dropouts, there is good evidence that the two distributions of play are different. If all dropouts in T play straightforwardly in all rounds, and none do so in M, then the *fraction of participants who play straightforwardly in all rounds* is 0.30 in T and 0.47 in M; a two-sided equality-of-proportions test yields $p = 0.01$.

Straightforward play

d ($)	Trad. ($N = 100$)	Menu ($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

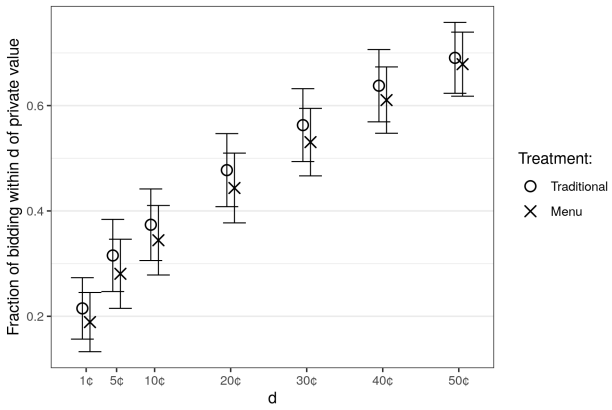


Figure 10: Rates of straightforward behavior, i.e. bidding within d dollars of the private value, under different values of d. (Only rounds where each bid within plus or minus d of private value are considered.) Standard errors are reported in parentheses. All p-values are two sided (Welch's test, weighted by the number of rounds such that all bids within d of the private value are possible). In the bar plots, the 95% confidence intervals are shown.

A.2 Auction

Overall, the Auction may seem "more difficult" for the participants than Median. Few bids followed the exact dominant strategy of submitting a bid equal to their private value (or, due to tie-breaking, bidding one cent more than their bid). In fact, bids within 1 cent of the private value are 21% of bids in T and 19% of bids in M. Moreover, only 7% of participants

[39] Under the extremely unrealistic assumption that all dropouts in T would have played straightforwardly *every* round, yet all in M would have played straightforwardly in *no* rounds, the averages in the two treatments are nearly the same: 0.71 in T and 0.72 in M.

Number of participants with specific errors

	Trad. $N = 100$	Menu $N = 100$	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	

Comprehension and S.F. play

		Num. mistakes		
d (\$)		$= 0$	≥ 1	p
0.05	Trad.	0.33	0.25	0.33
	Menu	0.31	0.23	0.23
	Overall	0.32	0.24	0.11
0.20	Trad.	0.50	0.39	0.18
	Menu	0.47	0.39	0.07
	Overall	0.49	0.39	0.05
0.50	Trad.	0.73	0.59	0.10
	Menu	0.70	0.63	0.06
	Overall	0.72	0.61	0.04

Figure 11: Different tests of the comprehension of the Auction mechanism, and the effect of overall comprehension (making a mistake in any question) on straightforward play.

The first table shows different types of errors, and the count of participants who make those errors. 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 you 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).

The second table shows rates of straightforward play for different values of d, with subjects broken down by performance in the comprehension questions overall. 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).

bid within 1 cent every round under T; the same number is 1% under M.

On the other hand, the number of possible strategies was far larger in Auction than in Median, as any bid (of a whole number of cents) between \$0.00 and \$5.00 was possible. Taking a "range of strategies" consisting of 1/5 of all possible bids (the same overall fraction as straightforward play in Median) gets 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 rates of straightforward play of 69% under T and 67% under M.

It is unclear which value of d corresponds to a "good understanding" of the (approximate) dominant strategy of the mechanism. However, Figure 10 indicates that many reasonable values of d produce similar non-results. Other (perhaps more standard) measures of how far participants deviate from "bidding their value" also yield non-results. Namely, one can consider the "mean absolute deviation" (i.e. the mean of the absolute difference between the private value and the bid), a statistic used by Li (2017) and prior work. The mean absolute deviation in T is \$0.51, and in M it is \$0.55; a two-sided t-test for equality of means yields a p-value of 0.58. In all of our results reporting straightforward behavior, we ignore all rounds where the private value was within d of the maximum possible or minimum possible bid (because in these rounds, not all possible bids within plus or minus d of the private value are possible). This also has almost no impact on our findings; including these rounds yields similar non-results for all tests.

Participants are much more likely to under-bid than over-bid. Specifically, 63% of bids are below the payout in T, and 66% are below the payout in M; 21% of bids are above the payout in T, and 22% are below in M. Neither of these comparisons are statistically different across treatments.

Despite these experimental negative results, we believe that menu descriptions make strategyproofness mathematically more apparent, even for a second price auction. In our case, this mathematical ease did not translate at all into an experimental result. While we cannot know precisely why this is, looking at the comprehension questions provides some insights as to how participants were simply *confused* by the menu description. Namely, one of the main misconceptions we hoped that menu descriptions could dispel is that *your price in the Auction does not equal your bid* (a special case of the defining property of menu descriptions, which is that you cannot influence the price you will pay). Our menu description did not achieve this goal whatsoever. Figure 11 shows how many participants under each treatment submitted answers to comprehension questions where they wrote their own bid as their price. 8 participants submitted such answers in T; 18 in M. This difference suggests (two-sided p-value of 0.06) that our menu description completely failed in this goal. It may be that the added complexity (in terms of word length and overall conceptual difficulty) of

the M treatment overshadowed any emphasis that the M treatment was intended to put on how the price was calculated.[40] It may be the case that menu descriptions of second price auctions will not be useful in practice without additional explanations or clarifications; it may be that future work may provide better and more useful menu descriptions than ours.

B Additional Preliminaries

B.1 Environments

First, we formally define an *environment*.

Definition B.1. *An* environment *with n agents consists of a set A of outcomes, and sets $\mathcal{T}_1, \ldots, \mathcal{T}_n$ of types of the n agents. For* ordinal environments*, each $t_i \in \mathcal{T}_i$ induces a weak order*[41] *$\succeq_i^{t_i}$ 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 $u_i(t_i, a) \geq u_i(t_i, b)$).*[42]

We remark that all of the mechanisms we consider are "direct", in that the agents are simultaneously asked to report their type 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 $\{\{b \in A : a \sim_i b\} | a \in A\}$. We let $[a]_i = \{b \in A : a \sim_i b\}$, and we denote the collection of i-outcomes by A_i.*[43]

[40] 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).

[41] 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 $b \succeq a$).

[42] 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 $b \succeq^{t_i} a$.

[43] When no confusion can arise, we sometimes do not distinguish between an outcome a and the corresponding i-outcome $[a]_i$. We also let $\succeq_i^{t_i}$ denote the natural partial order over i-outcomes corresponding to the preference relation over outcomes. For example, in cases such as matching where types (of the

B.2 Matching mechanisms

Our first domain of interest is in (ordinal) *matching mechanisms*, specifically the three canonical mechanisms of Serial Dictatorship, Top Trading Cycles, and Deferred Acceptance. Each of these mechanisms is used abundantly 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 which may leave some agents unmatched) of agents (known as "applicants") to "institutions". Types consist of all possible strict orderings over some subset of the institutions (the subset which the type views as "acceptable").*

Note that for each player i, the i-relevant outcome sets are completely determined by i's partner in the matching produced. That is, the i-outcomes in matching environments consist of all matchings μ such that $\mu(i) = p$ for some institution p, and all such matchings are indistinguishable from the point of view of i. When no confusion can arise, we identify i-outcomes in matching markets with the institution an applicant matches to in that i-outcome.

Note that we assume that the preferences of agents are strict, i.e. no agent is indifferent between two institutions. All of our results apply equally well in environments with and without "outside options", i.e. regardless of whether agents 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 agents). Note that we do not treat the institutions as strategic in any of our matching settings.

Definition B.4. *Serial Dictatorship (SD) is defined with priority order π (i.e. π is any bijection between $[n]$ and the applicants). The matching is produced as follows: in order $i = \pi(1), \pi(2), \ldots, \pi(n)$, applicant i selects and is permanently matched to their favorite institution which has not already been selected by some preceding (in the order) agent.*

Serial Dictatorship always produces a Pareto-optimal[44] matching, and is extremely simple to understand and play. In practice, the priority order π is often selected (from all $n!$ possibilities) uniformly at random; this mechanism is called Random Serial Dictatorship (RSD). RSD may be a reasonable choice of mechanism when applicants do not have meaningful priorities at the institutions. Similarly, SD may be a reasonable choice of mechanism when all institutions have the same priorities over applicants.

applicants) 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.

[44] 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.

Definition B.5. *Top Trading Cycles (TTC) is defined with respect to priority orders $\succ_p$, one for each institution p, over applicants. The matching is produced as follows. Repeat the following until everyone is matched (or has exhausted their preference lists): each remaining (i.e., not-yet-matched) 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 are only a finite number of vertices). For each applicant in this cycle, permanently match her to the institution she is pointing to. (And she, and the institution, will not participate in future iterations.)*

The intuitive rationale behind the TTC mechanism is as follows: If an agent a has top priority at some institution p, then that agent "has a right" to attend p, and a will never go below p in their preference list (assuming they report truthfully). However, the agent may "trade away their right" to match to p. For example, if another agent b has top priority at institution q, and a's favorite institution is q, but b's favorite institution is p, then a and b can "trade their priorities" at p and q. This trade exactly occurs as a two-agent (and two-institution) cycle in the TTC procedure. Longer cycles are more elaborate priority trading cycles. Moreover, after the agents in such a cycle are matched, these trades can be made in the sub-market resulting from removing these agents and institutions, and this process is continued until all agents 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.6 (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 agent is matched to an institution at least as good as any institution where they have top priority.*

Definition B.7. *Deferred Acceptance (DA) is defined with respect to priority orders $\succ_p$, one for each institution p, over applicants. The matching is produces as follows. Repeat the following until everyone is matched (or has exhausted their preference list): A currently unmatched applicant is chosen to "propose" to their favorite institution which has not yet "rejected" them. The institution "reject" every proposal except from the* top priority applicant *who has proposed to them thus far. Rejected applicants become unmatched (while the top priority applicant is "tentatively matched" to the institution). The tentative allocations become final in the last step.*

Typically, when we refer to DA we mean applicant-proposing DA as described above. One can also consider *institution* proposing DA (simply interchanging the roll of preferences and priorities). We denote this 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, p of an applicant and institution such that $p \succ_a \mu(a)$ and $a \succ_p \mu(p)$. A matching is stable if it is not unstable. Intuitively, the DA process starts from the preferences of the applicants, 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 p, then in order to preserve stability, we can only tentatively assign to p whichever applicant p prefers. This logic is repeated until every applicant finds some tentative match, which is then made final.

DA is an equally remarkable mechanism. The matching it produces is always stable. Moreover, it produces the "applicant-optimal" stable matching (and thus 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 (and 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*, not any particular mechanism used to implement it. When we need to refer to a specific algorithmic representation, we use words such as "the traditional DA algorithm".

B.3 Auctions

Our second domain of interest is *(combinatorial) auction* environments. These are cardinal-preference environments in which some set of items is sold to some set of bidders.

Definition B.8. *An* auction environment *for n players (known as bidders) and a set of items M is one in which the outcomes are allocations $(A_1, \ldots, A_n)$ over M (i.e. each $A_i \subseteq M$ and $A_i \cap A_j = \emptyset$ for $i \neq j$), along with prices $(p_1, \ldots, p_n)$ that the bidders must pay. The set of* types *$\mathcal{T}_1, \ldots, \mathcal{T}_n$ are (arbitrary) sets of valuations function $v_i : 2^M \to \mathbb{R}_{\geq 0}$, which map subsets of M to a cardinal utility 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-relevant outcome sets 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 typical auction domains of interest, each type set $\mathcal{T}_i$ is an some set $\mathcal{V}$ of valuation functions with some well-motivated properties. All valuations we discuss will be monotone (i.e. for any $S \subseteq T$, we have $v(S) \leq v(T)$) and normalized (so that $v(\emptyset) = 0$). Across these different settings, we consider the social choice function maximizing *welfare*, i.e. the maximizing the sum $\sum_i v_i(A_i)$ over all possible allocations $(A_1, \ldots, A_n)$.

We assume that the price functions charged by the mechanisms are the canonical VCG prices, which charge bidder i the externality they exert 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 $(A_1, \ldots, A_n)$, player i is charged

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

(where the max is taken over all allocations A' to bidders other than i). When VCG prices are charged in a welfare-maximizing setting, the result is always a strategyproof social choice function. For concreteness, we will assume that all valuation functions (and thus all prices) are integers in $\{0, 1, \ldots, K\}$, 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).

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

Definition B.10. *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 welfare maximizing social choice function with* n *unit demand bidders and* m *items as* $UD = UD^{m,n}$.

Definition B.11. *A* gross substitutes valuation $v_i : 2^M \to \mathbb{R}_{\geq 0}$ *is a valuation such that the following holds: for any price vector* $(p_j)_{j \in [m]}$ *and* $S \in \arg\max_T v_i(T) - \sum_{j \in T} p_j$, *if another price vector* $(q_j)_{j \in [m]}$ *is such that* $q_j \geq p_j$ *for all* j, *then there exists an* $S' \in \arg\max_T v_i(T) - \sum_{j \in T} q_j$ *such that* $S \cap S' \supseteq \{j \in [m] \mid p_j = q_j\}$. *We denote the welfare maximizing social choice function with* n *gross substitutes bidders and* m *items as* $GS = GS^{m,n}$.

B.4 Notation

We say that $f(n) = \widetilde{\mathcal{O}}(g(n))$ (resp. $\widetilde{\Omega}(g(n))$) when there exists an integer k such that $f(n) = \mathcal{O}(g(n)\log^k(g(n)))$ (resp. $\Omega(g(n)\log^k(g(n)))$). That is, $\widetilde{\mathcal{O}}$ and $\widetilde{\Omega}$ ignore logarithmic factors. For an integer n, we let $[n] = \{1, 2, \ldots, n\}$. In matching markets, we often use variables like d_i to refer to applicants and h_i to refer to institutions.

C Omitted Proofs

For the reader's convenience, throughout the appendix, we restate each result before giving the proof.

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 a applicant d_* and let $\mu_0 = APDA(P)$. Let $\overline{P} = P|_{d_*:\emptyset}$ denote altering P by having d_* submit an empty preference list (i.e., marking all institutions as unacceptable). Note that the outcome of $IPDA(P \setminus \{d_*\})$ exactly corresponds to the outcome of $IPDA(\overline{P})$, and the institutions that prefer d_* over their match in the outcome of $IPDA(P \setminus \{d_*\})$ are exactly those that propose to d_* in $IPDA(\overline{P})$.

If d_* is unmatched in μ_0, then by the fact that μ_0 is the applicant-optimal stable outcome (Corollary F.3), she is unmatched in all stable matchings and in particular in $IPDA(P)$. Thus, the run of $IPDA(P)$ is identical to the run of $IPDA(\overline{P})$, and d_* does not get any proposals in $IPDA(\overline{P})$ that she deems acceptable by P. In particular, d_* is not matched according to Description 1.

Assume now that d_* is matched in μ_0 and let $h_* = \mu_0(d_*)$. We want to prove that d_*'s favorite proposal received in $IPDA(\overline{P})$ is exactly h_*. We do this in two steps: first, we show d_* gets a proposal from h_* in $IPDA(\overline{P})$, and second, we show d_* does not get any better proposal.

Consider $P' = P|_{d_*:\{h_*\}}$, i.e., preferences in which d_* submits a list consisting only of h_*. Observe that μ_0 is stable under preferences P'. Thus, by the Lone Wolf Theorem (Theorem F.6), d_* must be matched in $IPDA(P')$, and thus must be matched to h_*. Now, using the fact that DA is independent of the order in which we choose to make proposals (Corollary F.3), we can construct a run of $IPDA(\overline{P})$ by first executing $IPDA(P')$, then having d_* reject h_* and continuing the rejection process until the end. Thus, d_* gets a proposal from h_* in $IPDA(\overline{P})$, as desired.

Now, consider $P'' = P|_{d_*:L}$, where L is d_*'s preference list from P, but truncated right

before h_* (i.e., all institutions at or below h_* are removed from d_*'s list). It will suffice to prove the following:

Lemma C.1. *Applicant d_* is not matched in $IPDA(P'')$.*

Proof. Suppose for contradiction that d_* is matched in $\mu' = IPDA(P'')$. By the fact that DA always results in a stable matching under the reported preferences (Lemma F.1), we know that μ' is stable for P''. But by the fact that $APDA$ results in the applicant-optimal stable matching (Corollary F.3), we know that $\mu' \neq \mu_0$ is not stable for P. The only possible blocking pair under P (but not under P'') must involve applicant d_* (as their preference is the only one which changes between P and P''). Say (d_*, h) is this blocking pair, so $h \succ^P_{d_*} \mu'(d)$ under the original preferences. But $\mu'(d)$ is still on d_*'s truncated list in P'', and thus h is on this list in P'' as well. Thus, μ' is unstable for P'' as well, a contradiction to the fact that DA is stable. □

Because d_* accepts no proposals in the run of $IPDA(P'')$, this constitutes a run of $IPDA(\overline{P})$ as well. Thus, d_* is not matched in $IPDA(P'')$, and thus d_* does not receive any proposal better than h_* in $IPDA(\overline{P})$, as desired.[45] □

Remark C.2. *We remark here that Theorem 3.1 extends to many-to-one markets. To show this, it suffices to fix a many-to-one profile of preferences, capacities, and priorities P, and consider a one-to-one market P' where there are "independent copies" of each institution (where the number of copies equals the capacity of the institution, and 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) for each place where the original applicant ranked the original institution). Ignoring the difference between the copied institutions, the run of applicant-proposing DA with any preferences is equivalent under P and P'. Thus, an applicant's menu is equivalent in P and P', and a menu description can be given through institution-proposing DA under preferences P'.*

Theorem 6.2. *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 description of applicant i's DA menu is non-local.*

Proof. Assume for contradiction that D is a local, applicant-proposing menu description for

[45] Mathematically, an approach vaguely similar to the one used here (in terms of truncating lists in the way we do) has already been used to prove that (a variant of) DA is strategyproof in Hatfield and Milgrom (2005). It also builds on technical lemmas such as those in (Immorlica and Mahdian, 2005; Cai and Thomas, 2022).

some applicant d_* in a market with applicants d_1, d_2, and d_*. We first define priorities of three institutions h_1, h_2, h_3 as follows:

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

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

$$\succ_1 : h_1 \succ h_2 \succ h_3 \qquad \succ'_1 : h_1 \succ h_3 \succ h_2$$
$$\succ_2 : h_2 \succ h_1 \succ h_3 \qquad \succ'_2 : h_2 \succ h_3 \succ h_1$$

Consider executing D when preferences are $P = (\succ_1, \succ_2)$. 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 nodes queries applicant d_1, and thus learns the difference between $P = (\succ_1, \succ_2)$ and $P' = (\succ'_1, \succ_2)$. 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. But, because D is local, to correctly compute the menu, the description must update h_1's label in some successor node of v. This is a contradiction, and thus, no local applicant-proposing menu description of DA exists. □

Theorem 6.3. *If there are at least four applicants and three institutions, then there exist preferences of the applicants such that any institution-proposing description of the DA outcome matching 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 h_3 and We first define priorities of tdree applicants d_1, d_2, d_3 as follows:

$$d_1 : h_3 \succ h_1 \succ h_2$$
$$d_2 : h_1 \succ h_2 \succ h_3$$
$$d_3 : h_2 \succ h_3 \succ h_1$$
$$d_4 : (\text{any complete preference list})$$

Next, we consider two possible preference lists for each of h_1, h_2, h_3:

$$\succ_1 : d_3 \succ d_1 \succ d_2 \succ d_4 \qquad \succ'_1 : d_3 \succ d_1 \succ d_4 \succ d_2$$
$$\succ_2 : d_1 \succ d_2 \succ d_3 \succ d_4 \qquad \succ'_2 : d_1 \succ d_2 \succ d_4 \succ d_3$$
$$\succ_3 : d_2 \succ d_3 \succ d_1 \succ d_4 \qquad \succ'_3 : d_2 \succ d_3 \succ d_4 \succ d_1$$

Analogously to the previous proof, one can show from these preferences that no local institution-proposing outcome description exists. In particular, at the latest point where the execution of an extensive form description learns the difference between (without loss of generality) $P = (\succ_1, \succ_2, \succ_3)$ and $(\succ'_1, \succ_2, \succ_3)$, the state of the description requires a non-local update (because under P, the matching is $\{(h_1, d_2), (h_2, d_3), (h_3, d_1)\}$, and under P', the matching is $\{(h_1, d_1), (h_2, d_2), (h_3, d_3)\}$, but d_1 has already been queried from every institutions' list). □

Theorem 7.3. *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 description (computing the allocation), and for menu descriptions (computing one player's menu).*

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 following problem: 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 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.3. *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.4. *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.[46] □

D Global Structure of Matching Mechanisms

Our goal in this section is to formalize and prove the results discussed in Section 5.4. Recall that our motivation question is the following: for the individualized dictatorship description of TTC, can "stage (c)" of Definition 4.2 be "skipped" (i.e. performed without further access to t_i)?

Towards this goal, we recall and then introduce some general definitions. The "taxation complexity" of a social choice function counts the number of distinct menus the social choice function can have. At some intuitive level, this serves as a more "compact" representation of the complexity of the social choice function than our investigations into the complexity of

[46] 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).

computing menus yields (in fact, this definition has yielded impressive breakthroughs in algorithm mechanism design by capturing this notion of complexity (and, for example, proving that it bounds the *communication complexity* of the mechanism)).

Definition D.1 (Dobzinski (2016))**.** *The* taxation complexity $\mathrm{tc}(f)$ *of* f *is the logarithm of the maximum number of distinct menus which may be presented to any player. That is,*

$$\mathrm{tc}(f) = \log_2 \max_i \left|\left\{\mathtt{M}_{t_{-i}} \mid t_{-i} \in \mathcal{T}_{-i}\right\}\right|.$$

Recall that $\mathtt{M}_{t_{-i}} \subseteq A_i$ is a subset of the i-relevant outcome, not the overall outcomes A. For our purposes, we are interested in the complexity needed to achieve the full outcome in A at the end. Thus, we introduce the following definition:

Definition D.2 (Global taxation complexity)**.** *For any social choice function* f*, player* i*, and a type profile* $t_{-i} \in \mathcal{T}_{-i}$*, define the "global menu" presented to* i *by* t_{-i} *to be the function* $\mathtt{GM}_{t_{-i}} : \mathcal{T}_i \to A$ *given by* $\mathtt{GM}_{t_{-i}}(t_i) = f(t_i, t_{-i})$*.*

The global taxation complexity $\mathrm{gtc}(f)$ *of* f *is the logarithm of the maximum number of distinct global menus presented to any player. That is,*

$$\mathrm{gtc}(f) = \log_2 \max_i \left|\left\{\mathtt{GM}_{t_{-i}} \mid t_{-i} \in \mathcal{T}_{-i}\right\}\right|.$$

When we need to distinguish between global menus and "ordinary" menus, we refer to the latter as "local menus". Clearly, the local taxation complexity gives a lower bound for the global taxation complexity.

The connection between this new definition and our motivating question is the following: If a individualized dictatorship description is unable to query t_{-i} during the "remainder stage", then whenever the description presents the menu to player i, *it must in fact know the entire global menu*. Thus, a individualized dictatorship algorithm that cannot access t_{-i} during the "remainder" stage is essentially an algorithm for computing the global menu.

We now investigate the global taxation complexity of SD, DA, and TTC.

D.1 Serial Dictatorship

For any matching mechanism with n applicants and n institutions, any menu is simply a subset of institutions. There are at most 2^n such sets, and thus the local taxation complexity of any matching mechanism is at most n. We show that for the SD mechanism, this bound also holds (up to lower order terms) for the global taxation complexity. Intuitively, this follows because the SD mechanism works in such a simple way that any particular player can only "perturb" the matches of the other players in a "simple" (or "well-controlled") fashion.

Theorem D.3. *The global taxation complexity of SD is* $\widetilde{\Theta}(n)$.

Proof. The lower bound holds for all matching mechanism where a player might have any possible local menu. For the upper bound, it suffices to consider the first player in line – call them player 0, and fix a set of preferences of all other players. Conduct the following process: for each player $i \geq 1$ (in order), take the first element j (still) on their list, and remove it from the list of all agents $i' > i$. Do this recursively (considering the first element still on their filtered lists). Observe that, by the time we pass agent i in the mechanism, this institution j will certainly be taken (either by them if j is still available, or by an earlier agent if not). By induction this is true in the filtered preference lists for all players.

Lemma D.4. *Regardless of what institution player* 0 *picks from their menu (which is all institutions, as they are the first player in line), each agent will be matched to either their first or second institution in their filtered list.*

Proof. Suppose player 0 takes institution a. The mechanism will run initially with each agent taking their first choice (all of which are distinct), until some agent i which ranks a as their first choice. This agent will take their second choice b (which is yet unclaimed as each prior agent has taken their first choice). But, applying this argument inductively below i, we see that all agents will be matched to their first choice until i' which ranks b first, and i' will get their second choice. This applies recursively until the mechanism is finished. □

Thus, define p' by taking the first and second choice for each player from the filtered list. Then we know that p' presents the same global menu to player 0 as the original profile p. Because each list in p' is of size at most 2, it takes $\widetilde{O}(n)$ bits to write down p', and thus the global taxation complexity of serial dictatorship is $\widetilde{O}(n)$. □

Moreover, the above lemma proves that the global menus in SD have a very regular structure – there is some "default matching" of agents $i > 0$, and there is a directed acyclic graph of "trades" which might occur based on the choice of player 0 (and moreover, the priority ordering of SD gives a topological sort of this DAG).

We remark that while the global taxation complexity of SD is small, this does not necessarily capture why SD feels so intuitively "simple" – the reason for this foremost involves the fact that SD "visits each bidder at most once". This property can be almost unrelated to gtc. In fact, in Section D.3, we construct a mechanism called SD_{rot} which seems at first to just be a small variation on SD (maintaining most of its simplicity, and in particular visiting each bidder once only), yet has high gtc. This shows that gtc can be a rather "brittle" yardstick for measuring simplicity, necessitating more refined non-global approaches (such as individualized dictatorships).

D.2 Deferred Acceptance

For a crude *upper bound* on the global taxation complexity of any matching mechanism, observe that the number of distinct type profiles t_{-i} is at most $(n!)^{n-1} = 2^{\mathcal{O}(n^2 \log n)}$, and thus gtc is at most $\widetilde{\mathcal{O}}(n^2)$. The proof of Theorem 5.5 constructs a set of $2^{\Omega(n^2)}$ preference profiles, and shows that each of these preference profiles presents a distinct global menu to one player (specifically, Lemma 5.7 proves precisely this). Indeed, one way to summarize the proof of Theorem 5.5 using the language of this section is the following: Theorem 5.5 shows that the global taxation complexity is high, *and moreover* that you cannot avoid reading essentially the entire list of every player in order to compute the menu of a single player. This proves that the gtc(DA) is as high as possible (up to log factors), namely:

Corollary D.5. *The global taxation complexity of DA is* $\Omega(n^2)$.

D.3 Top Trading Cycles

We now work towards the main result of this section, which is that the global taxation complexity of TTC is large. This shows that the third phase of the individualized dictatorship description of TTC is absolutely crucial, and cannot avoid accessing t_{-i}.

To prove this result, we first define a mechanism SD_{rot} seems *nearly as simple* as SD, yet also has high gtc. This is already an interesting example of how the gtc might be high even though the mechanism has a applicant-linear individualized dictatorship description. Afterwards we will show that, intuitively, TTC can "simulate" SD_{rot}, and thus TTC has gtc at least as high as SD_{rot}.

Definition D.6. *Define* SD_{rot} *as follows: players and institutions are both labeled with* $[n]$. *Player* 1 *can pick any institution, call it institution* j. *After that,* player j *may pick any institution, followed by* $j+1$, *then* $j+2$, *etc, cyclically mod* n *(ignoring player* 1 *in this ordering).*

This mechanism is clearly strategyproof, and indeed is described above with a individualized dictatorship applicant-linear description. Despite this, we now show that the global taxation complexity of player 1 is high. Intuitively, this will follow because, for each player $i > 1$, player i may go *anywhere* in the ordering of the remaining agents, and thus *any* part of player i's preference list might matter for determining the final matching. We formalize this intuition in the proof.

Theorem D.7. *The global taxation complexity of* SD_{rot} *is at least* $n^2/9$.

Proof. Let the set of institutions and agents both be denoted $[n] = \{1, \ldots, n\}$. Let $k = n/3$, and for notational convenience, let $r_1, \ldots, r_k$ denote all institutions congruent to 1 mod 3,

let $x_1^0, \ldots, x_k^0$ denote those congruent to 2, and let $x_1^1, \ldots, x_k^1$ denote those congruent to 0. Let $a_1, \ldots, a_k$; $y_1, \ldots, y_k$; and $z_1, \ldots, z_k$ denote the corresponding sets of agents.

We define a collection of preference profiles of all agents other than 1 based on k^2 bits $b_{i,j} \in \{0, 1\}$ for $i, j \in [k]$. For such a bit vector $\vec{b}$, let $p_{\vec{b}}$ be defined as follows: For each $i \in [k]$ (other than $a_1 = 1$), we have

$$\begin{aligned} a_i: &\quad r_1 \succ r_2 \succ r_3 \succ \ldots \succ r_k \\ y_i: &\quad x_1^{b_{i,1}} \succ x_1^{1-b_{i,1}} \succ x_2^{b_{i,2}} \succ x_2^{1-b_{i,2}} \succ \ldots \succ x_k^{b_{i,k}} \succ x_k^{1-b_{i,k}} \\ z_i: &\quad x_1^{1-b_{i,1}} \succ x_1^{b_{i,1}} \succ x_2^{1-b_{i,2}} \succ x_2^{b_{i,2}} \succ \ldots \succ x_k^{1-b_{i,k}} \succ x_k^{b_{i,k}} \end{aligned}$$

In particular, the preference list of each a_i is identical, and for each y_i and z_i, all agents agree on the order up to flipping "adjacent places", as determined by the bits $b_{i,1}, b_{i,2} \ldots, b_{i,k}$.

Lemma D.8. *For each distinct bit vector $\vec{b}$, the preference profile $p_{\vec{b}}$ defines a distinct global menu.*

Proof. Consider two preference profiles p and p' induced by some distinct bit vectors $\vec{b}$ and $\vec{b}'$, and suppose $b_{i,j} = 0$ and $b'_{i,j} = 1$. It suffices to show that there exists some institutions which player 1 can clinch, such that a distinct matching will be produced when player other than 1 use p verses p'. We claim that r_{i-j+1} is such an institutions (with indices taken mod n). Proof: After player 1 selects r_{i-j+1}, agents in "block" $i - j + 1$, then $i - j + 2$, etc. will select institutions in order In particular, agents in block i will be the jth block to select their institution. Observe that, as all agents $y_{i'}$ and $z_{i'}$ agree on the ordering (except for adjacent flips), the ℓth block to pick will select institutions x_ℓ^0 and x_ℓ^1 (in some order). Thus, the favorite remaining institutions of block i will be exactly x_j^0 and x_j^1. When $b_{i,j} = 0$, agent y_i gets x_j^0, and when $b'_{i,j} = 1$, agent z_i gets x_j^0. Thus, p and p' present distinct global menus to player 1, as desired. □

Thus, there are at least 2^{k^2} distinct possible global menus, and the *gtc* of SD_{rot} is at least $k^2 = n^2/9$. □

Through a somewhat delicate construction, we now show that TTC can essentially "simulate" behavior similar to the above[47].

[47]While the construction is slightly involved, it is fairly easy to derive once the goal of "simulating SD_{rot}" is established. On the other hand, without using SD_{rot} as a stepping stone, it is unclear how to create a construction which works. For instance, if the "TTC graph" before agent 1 picks consists of a single directed path, then $\widetilde{\mathcal{O}}(n)$ many bits can be used to represent the global menu (because the remaining run of TTC will consist solely of agents pointing to things "above them" in the path, so at most one additional "edge" per agent could be needed). On the other hand, if all of the institutions point directly to agent 1, then when agent 1 picks all they can do is remove a single institution (which, as in the case of SD, should

Theorem D.9. *The global taxation complexity of TTC is* $\Omega(n^2)$.

Proof. We describe a set of priorities and a large collection of preference profiles which each present a distinct global menu to player 1. We do this in two steps. First, we construct a collection of preference profiles such that, for some large set of "primary" agents $p_1^A, \ldots, p_k^A$ who only rank "primary" institutions[48] $p_1^O, \ldots p_k^O$, the TTC mechanism can simulate each of the k different "serial dictatorships" $SD(p_i^A, p_{i+1}^A, \ldots, p_k^A)$ for $i \in \{1, \ldots, k\}$. The choice of this $i \in \{1, \ldots, k\}$ is based on agent 1's choice from their menu. Second, we show that this implies that the global taxation complexity of TTC is high.

Consider any k. We consider a market with $n := 2k+2$ agents and n institutions. There is agent 1 (to whom the menu is presented); $k+1$ agents $r_0^A, r_1^A, \ldots, r_k^A$ in the rotation gadget; and k primary agents $p_1^A, \ldots, p_k^A$. There is a "helper institution" h^O in the rotation gadget; $k+1$ institutions $r_0^O, r_1^O, \ldots, r_k^O$ in the rotation gadget; and k primary institutions $p_1^O, \ldots, p_k^O$. Given any preference profile Q of primary agents over primary institutions, we will define a preference profile P_Q in which the primary agents have preferences Q. All preferences and priorities other than those of primary agents are constant (including those of primary institutions). We now describe these preferences and priorities, as illustrated in Figure 12.

$$
\begin{array}{lll}
r_0^O: & 1 & \\
r_i^O: & r_{i-1}^A \succ p_i^A & \forall i \in \{1, \ldots, k\} \\
r_i^A: & r_i^O \succ h^O & \forall i \in \{1, \ldots, k\} \\
h^O: & r_0^A \succ r_1^A \succ \ldots \succ r_{k-1}^A \succ r_k^A & \\
p_i^O: & r_1^A \succ r_2^A \succ \ldots \succ r_{k-1}^A \succ r_k^A & \forall i \in \{1, \ldots, k\}
\end{array}
$$

In particular, each of the primary institutions have identical preference lists. Observe that each institution points transitively to player 1, so player 1's menu consists of all institutions.

Lemma D.10. *When player* 1 *picks some institution* r_i^O *for* $i \in \{0, \ldots, k-1\}$ *as their top choice, the primary agents are assigned to the primary institutions according to* $SD(p_{i+1}^A, p_{i+2}^A, \ldots, p_k^A)$.

Proof. We consider the run of TTC after player 1 points to institution r_i^O. First, player 1 and each agent $r_0^A, \ldots, r_{i-1}^A$ (there are no such agents if $i = 0$) is matched to their top institution in one cycle. Agent r_i^A is then matched to h^O in a one-agent cycle. Now, consider the

intuitively not allow them to effect the final matching all that much). The construction below, using one "long path" and many isolated nodes off of that path, conforms to the intuition of why TTC is hard – the long path gives agent 1 the ability to influence the rest of the run of TTC very significantly by dramatically changing the order in which future cycles (among the isolated nodes) are eliminated.

[48] In this proof, we denote institutions using a super-scripted "O", which stands for "object".

primary institutions. Each of them have the same preference list, and their favorite remaining agent is r^A_{i+1}. But r^A_{i+1} points to r^O_{i+1}, and r^O_{i+1} points to p^A_{i+1}. Thus, p^A_{i+1} will be matched to their top choice p^O_x of primary agents according to the primary agent preferences Q. The four agents and institutions $r^A_{i+1}, r^O_{i+1}, p^A_{i+1}, p^O_x$ are then removed in a cycle. Similarly, p^A_{i+2} is then transitively pointed to by all remaining primary institutions, and they are matched to their favorite remaining choice and removed. This happens successively for each primary agent $p^A_{i+1}, p^A_{i+2}, \ldots, p^A_k$. □

We now show that this implies that the global taxation complexity of TTC is high[49] For notational convenience, let k be even and let $x^0_1 = p^O_1, x^0_2 = p^O_3, \ldots, x^0_{k/2} = k-1$ denote those primary institutions p^O_i with i odd, and let $x^1_1 = p^O_2, x^1_2 = p^O_4, \ldots, x^1_{k/2} = p^O_k$ denote those p^O_i with i even. Let $y_1, \ldots, y_{k/2}$ and $z_1, \ldots, z_{k/2}$ denote the corresponding sets of primary agents (the y_js denoting those p^A_is with i odd and the z_js denoting those p^A_i with i even).

We define a collection of preference profiles of all primary agents based on $\binom{k/2}{2}$ bits $b_{i,j} \in \{0,1\}$ for $i \leq j \in [k/2]$. For such a bit vector $\vec{b}$, let $Q(\vec{b})$ be defined as follows: For each $i \in [k/2]$ we have

$$\begin{aligned} y_i: &\quad x_1^{b_{i,1}} \succ x_1^{1-b_{i,1}} \succ x_2^{b_{i,2}} \succ x_2^{1-b_{i,2}} \succ \ldots \succ x_i^{b_{i,i}} \succ x_i^{1-b_{i,i}} \\ z_i: &\quad x_1^{1-b_{i,1}} \succ x_1^{b_{i,1}} \succ x_2^{1-b_{i,2}} \succ x_2^{b_{i,2}} \succ \ldots \succ x_i^{1-b_{i,i}} \succ x_i^{b_{i,i}} \end{aligned}$$

In particular, all primary agents agree on the order up to flipping "adjacent places", as determined (for each y_i and z_i), by the bits $(b_{i,1}, b_{i,2} \ldots, b_{i,i})$.

Lemma D.11. *For each distinct setting of bit vector $\vec{b}$, the preference profile $P_{Q(\vec{b})}$ presents a distinct global menu to player* 1.

Proof. Let the bit vectors $\vec{b}$ and $\vec{b'}$ differ on index i, j, say $b_{i,j} = 0$ and $b'_{i,j} = 1$. This means that between $\vec{b}$ and $\vec{b'}$, the preferences of agents $y_i = p^A_{2i-1}$ and $z_i = p^A_{2i}$ differ on their preference over institutions x^0_j and x^1_j.

Suppose player 1's favorite institution is r^A_x for $x = 2(j-i)-1$. This will match primary agents to primary institution according to $SD(p^A_x, p^A_{x+1}, \ldots, p^A_k)$. In particular, a total of (*number*) many agents that go before y_i and z_i. Because all primary agents agree on the order of primary institution (up to flipping each x^0_ℓ and x^1_ℓ), the top remaining choices of y_i and z_i are x^0_j and x^1_j under both $\vec{b}$ and $\vec{b'}$. Under $\vec{b}$, agent y_i is matched to x^0_j, and under $\vec{b'}$, agent z_i is matched to x^0_j.

Thus, $P_{Q(\vec{b})}$ and $P_{Q(\vec{b'})}$ present distinct global menus to player 1 in TTC, as desired. □

[49] Essentially, this is just the proof that SD_{rot} has high taxation complexity, adapted to the fact that the "variable order" serial dictatorships are not "cyclic" but rather just "variable in starting point" (and this is certainly still rich enough to have high gtc).

Thus, there are at least 2 to the power of $\binom{k/2}{2}$ distinct global menus, and the global taxation complexity of TTC is at least $\Omega(k^2) = \Omega(n^2)$.

□

E Delicate Algorithms for Deferred Acceptance

In this section, we present additional algorithms for deferred acceptance. While technically interesting, we believe these algorithms 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".

E.1 Institution-proposing description of the matching of DA

First, we show that an *institution*-proposing variant of the DA algorithm can find the *applicant-optimal* stable matching (i.e. the outcome of the DA mechanism). Interestingly, this algorithm was already essentially known by Ashlagi et al. (2017) (henceforth, AKL).[50]

Theorem E.1 (Adapted from Ashlagi et al. (2017))**.** *Description 2 computes the applicant-optimal stable outcome. Moreover, Description 2 is an institution-proposing, $\widetilde{O}(n)$ memory algorithm.*

Proof. Description 2 follows the exact same order of proposals as "Algorithm 2 (MOSM to WOSM)", the men-proposing algorithm for the women-optimal stable matching, from AKL. The only difference apart from rewriting the algorithm in a more "pseudocode" fashion is that Description 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 $\widetilde{O}(n)$. □

[50] 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 2 An institution-proposing algorithm for finding the applicant-optimal stable outcome

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 μ be the result of institution-proposing DA
Let $\mathcal{D}_{\text{term}}$ be all applicants unmatched in μ ▷ $\mathcal{D}_{\text{term}}$ *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(μ, 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 μ* ◁
            WriteRotation(μ, V, d, h) ▷ *Updates the value of μ, V, and (possibly) h*
Return μ

function NextAcceptingApplicant(μ, 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(μ, 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 μ 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 μ* ◁
    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 $h \leftarrow h_1$

E.2 Applicant-proposing menu description of DA

In this section, we construct a preference-linear algorithm for computing one applicant's menu in DA. On an intuitive level, the algorithm works as per Example 2.4, but avoiding needing 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 2. The proof constricting this algorithm uses a sort of bijection between one applicant's menu in DA under some preferences, and the *institution*-optimal stable matching under a related set of preferences. Our preference-linear algorithm is then phrased as a variation of Description 2, which (reversing the roles of applicants and institutions from the presentation in Description 2) is able to compute the institution-optimal matching using an applicant-proposing algorithm.

Fix an applicant d_* and set P which 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 $h_i \in \mathcal{H}$:

$$d_i^{\text{try}} \ : \ h_i^{\text{try}} \succ h_i \succ h_i^{\text{fail}} \qquad\qquad d_i^{\text{fail}} \ : \ h_i^{\text{fail}} \succ h_i^{\text{try}}$$
$$h_i^{\text{try}} \ : \ d_i^{\text{fail}} \succ d_i^{\text{try}} \qquad\qquad h_i^{\text{fail}} \ : \ d_i^{\text{try}} \succ d_i^{\text{fail}}$$

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 in this augmented set of preferences P' will encode the menu, as we need.[51]

Proposition E.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 preferences P', we*

[51] 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 kicked back out from h_i after proposing, either immediately or after some "rejection rotation", 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 you "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.

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 E.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 kicked back out of 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 kicked back out of 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 a low memory (though rather complicated) applicant-proposing algorithm for the menu of (applicant-proposing) DA.

Theorem E.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 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 $h \in \mathcal{H}_{\text{open}}$ is immaterial—when such institutions receive a proposal from d_i^{try}, the algorithm can just query their lists for d_*. □

E.3 Institution-proposing menu-revealing description of DA

In this section, we construct an institution-proposing individualized dictatorship description of (applicant-optimal) DA.[52] 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 in Section E.2, our institution proposing menu-revealing description cannot be "reduced to" another algorithm such as Description 2, run on a related set of preferences. However, the algorithm is indeed a modified version of Description 2 which "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 their top choice from their menu, and "undo" all the rejections caused by d_* rejecting their choice.[53] To facilitate this, the algorithm has d_* reject institutions who 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.[54] The way this data structure works is involved, but one of the high-level keys to its correctness is a simple observation: *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.

[52]For some technical intuition on why such a description might exist, consider the construction used in Theorem 5.5, 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 5.5 shows that all of this information must thus be remembered in full. Now consider a run of Description 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 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).

[53]Description 2 is independent of the order under which proposals are chosen. Moreover, one can even show that d_* receives proposals from all h on their menu in Description 2. However, this does not suffice to construct our menu-revealing algorithm simply by changing the order of Description 2. The main reason is this: in Description 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 3 is inspired by the way Description 2 effectively "unrolls rejection chains" (by storing rejections in a list V and only writing these rejections to μ when you are sure they won't be "unrolled"), the bookkeeping of Description 3 is far more complicated (in particular, the algorithm maintains a DAG Δ instead of a list V).

[54]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 their menu. Instead, it is a "partial state" of Description 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 2 until we find the applicant-optimal outcome.

We present our algorithm in Description 3. 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 E.5. *The set $\mathcal{H}_{\text{menu}}$ output by Description 3 is the menu of d_* in (applicant-proposing) DA.*

Proof. Ignoring all bookkeeping, Phase 1 this algorithm corresponds to a run of $IPDA(P|_{d_*:\emptyset})$. The only thing changed is the order in which d_* makes these rejections happen, 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 E.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" $h \in \mathcal{H}$).

(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 hold 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 E.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).*

Description 3 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 μ'
Let $\mathcal{H}_*$ be all those institutions $h_i \in \mathcal{H}$ matched to d_i^{hold} in μ' ▷ *These institutions "currently sit at d_*"*
Let μ be μ', ignoring all matches of the form (d_i^{hold}, h)
Let $\mathcal{H}_{\text{menu}}$ be a copy of $\mathcal{H}_*$ ▷ *We will grow $\mathcal{H}_{\text{menu}}$*
Let Δ 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 Δ 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(μ, Δ, h)
        AdjustUnrollDag(μ, Δ, P, d, h) ▷ *Updates each of these values*
Return $\mathcal{H}_{\text{menu}}$
▷ ***Phase 2:*** *We now additionally have access to d_*'s menu* ◁
Permanently match d_* to their top pick h_{pick} from $\mathcal{H}_{\text{menu}}$
$(\mu, \mathcal{D}_{\text{term}}) \leftarrow$ UnrollOneChain$(\mu, \Delta, h_{\text{pick}})$
Continue running the Description 2 until its end, using this μ and $\mathcal{D}_{\text{term}}$, starting from Line 4
Return the matching resulting from Description 2

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 Δ 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 $(\mu, \mathcal{D}_{\text{term}})$

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

procedure AdjustUnrollDag(μ, Δ, 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 Δ
    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 Δ **then** ▷ *Here $h \succ_d \mu(d)$*
    Add $(d, \mu(d))$ to Δ ▷ *Record this in the rejection DAG*
    Add an edge from each $p \in P$ to $(d, \mu(d))$ in Δ, 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 Δ **then**
    AdjustUnrollDagCollision(μ, Δ, P, d, h) ▷ *Updates each of these values*

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

  ▷ *First, we drop all rejections from Δ 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 Δ 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 Δ, 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_{new} to Δ
  if $h_{\min} = h$ **then** ▷ *We replace p_1 with p_{new}*
    Add an edge from each $p \in P_1$ to p_{new}
    Add p_{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_{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*

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 3 and a point along that execution past such that the revealed truncated preferences are $\overline{Q}$, μ and $\mathcal{D}_{\text{term}}$ in Description 3 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 E.8. *Consider any moment where we query some institution's preferences list with-ing* NextInterestedApplicant *in Description 3. Let h be the just-queried institution, let d 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 3 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, say matched to some (d, h'), but $h' \succ_d h$. Because (μ, Δ) are unroll-correct for Q at $\overline{Q}$, and because Lemma E.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 E.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 24), and second, we consider the way that it creates a new node to handle d (starting on Line 27).

(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 2, the path from p_1 to p, and 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 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.[55]

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 27), 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 follows in the same way regardless of the previous cases regarding the existence of a path between p_1 and P, but still 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.

Now, 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

[55] This is the core reason why Description 3 cannot "unroll" to $IPDA(Q, d : \{h_i\})$—instead, it unrolls to a "partial state of AKL".

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 prefer 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 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 So (μ', Δ') is correct for $\overline{Q}'$ in this subcase.

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 UnrollOneChain(μ', Δ', 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', UnrollOneChain(μ', Δ', 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 $\overline{Q}'$ □

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

Lemma E.9. *Description 3 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 E.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 E.10. *Description 3 is a menu-revealing, institution-proposing, $\widetilde{O}(n)$ memory algorithm for (applicant-proposing) DA.*

Proof. We know Description 3 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 3, (μ, Δ) is

unroll-correct for Q at the truncated revealed preferences $\overline{Q}$ (for then, by definition, running Description 2 after 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 E.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. □

F Proofs of Known Results

This section is primarily dedicated to reproducing complete proofs from scratch of all lemmas we need related to DA and stable matchings. While this adds completeness to the paper, the primary purpose of including these is to illustrate the full theoretical underpinnings of our approach to DA (and how it may be considered to yield an easier to digest outline for a complete proof of the strategyproofness of DA).

F.1 Lemmas needed to prove Description 1 computes the menu

Lemma F.1. *The output of DA is a stable matching.*

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

Note that this algorithm gives us a very interesting existence result: it was not at all clear that stable matching existed before we had this algorithm.

Lemma F.2. *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 $d = \mu(h)$ 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. □

This immediately tells us that the result of applicant-proposing DA is the optimal stable outcome for the applicants, and that the result is independent of the order in which we chose applicant proposals.

Corollary F.3. *In the matching returned by APDA, every applicant is matched to their favorite stable partner.*

Corollary F.4. *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 F.5. *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 $h >_d \mu(d)$, because h is the d's favorite stable partner. If $d >_h \mu(h)$, then μ is not stable. Thus, h cannot be stably matched to any applicant they prefer less than d. □

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

Theorem F.6 (Lone Wolf / Rural Hospital Theorem)**.** *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 F.2, no stable matching pair any agent in $D \setminus D^0$. 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 F.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 F.7. *One can carefully check that all of our earlier proof work when we consider the match of an agent to be $\emptyset$ – this was intentional in order to make the above proof go through. Thus, these proofs seamlessly handle partial lists and market imbalance.*

F.2 Direct proof of the Strategyproofness of (Traditionally Described) DA

For a contrast between Section F.1 and Theorem 3.1, we also include the "shortest" proof we know of the strategyproofness of DA.

Lemma F.8. *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 themselves 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 $d' = \mu(h)$, who proposes to h during $APDA(P)$. In particular, h has already rejected $\mu'(h) \in T$ (who proposes to h in $APDA(P)$, because $h \succ_{\mu'(h)} \mu(\mu'(h))$), so $d \succ_h \mu'(h)$.

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

Theorem F.9. *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 the previous lemma. 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. $\square$

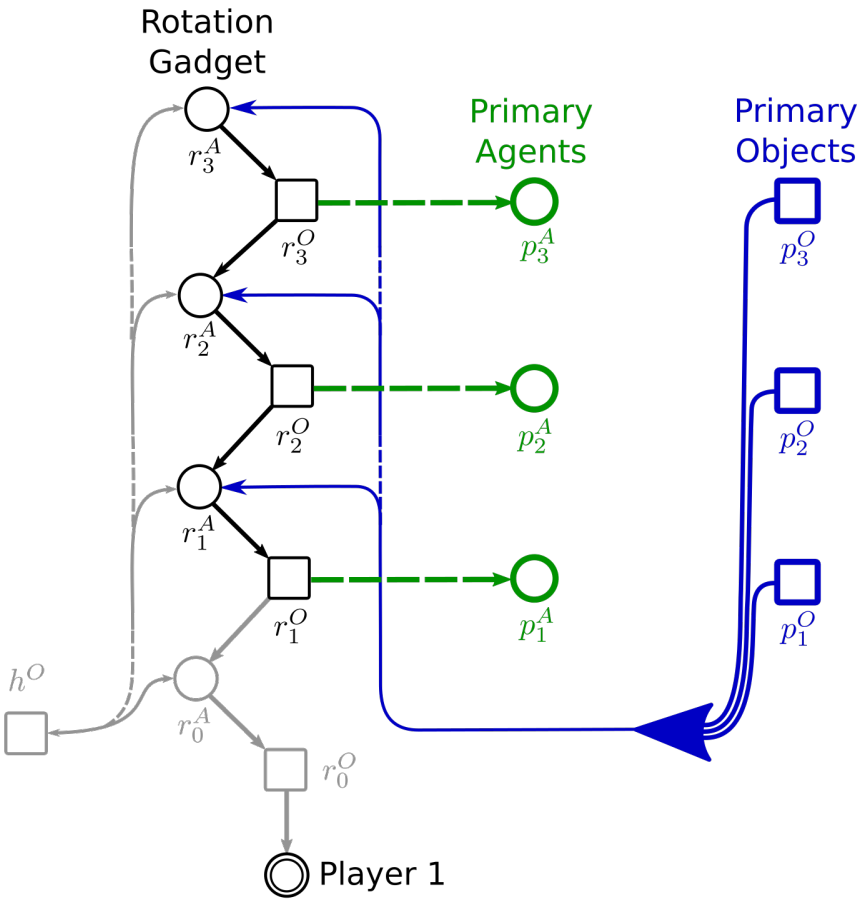


Figure 12: Illustration of the set of preferences witnessing that the global taxation complexity of TTC is $\Omega(n^2)$. Circular nodes represent agents and square nodes represent institutions. Primary agents have any preference order over primary institutions. The solid edges represent the top choice of an agent (with arrow directed to the choice). The dashed edges represent the second choice, and edges which are initially dashed (but later become solid) denote sequences of successively later choices.