

Self-Explanatory Strategyproof Mechanisms*

[MATCH-UP'22 submission; please do not distribute]

Yannai Gonczarowski[†]

Ori Heffetz[‡]

Clayton Thomas[§]

April 29, 2022

Abstract

A hallmark of the Deferred Acceptance (DA) algorithm is that it is strategy-proof. Alas, this property is far from apparent from the algorithm description, and despite many attempts at explaining it, there is a growing body of evidence that participants continuously attempt (unsuccessfully) to “game it.” We consider the problem of *explaining* the fact that a given mechanism such as DA is strategyproof, by changing the way the mechanism is *described*.

The *menu* of player i in a strategyproof mechanism is defined as the set of all possible outcomes that i might receive, holding the reports of all other players fixed. This gives rise (Hammond, 1979) to a two-step *menu description* of the mechanism for that player: 1) Using on the reports of players other than i , calculate player i ’s menu, and 2) Allocate player i to the outcome from their menu that they ranked highest. When a mechanism is *defined* to player i using a menu description, it is easy for her to see that the mechanism is strategyproof. Given any mechanism, its menu for player i can be calculated by repeated invocations of the mechanism. However, the resulting menu description can be vastly *harder to understand* than traditional descriptions, and therefore impractical for real-world use. In this paper we propose, rather than defining menus through directly defined mechanisms, defining mechanisms through directly defined menus. Thus, we seek *simple* and transparent menu descriptions. We give both positive results (i.e. new and appealing descriptions of known mechanisms) and negative results (i.e. formal barriers to even better descriptions).

For DA, we prove that any menu description must differ *drastically* from traditional descriptions. Specifically, we give a formal sense in which there is no menu description of DA that is a simple (or even “reasonable”) variation of the traditional DA description. This contrasts deferred acceptance with other popular matching mechanisms such as top trading cycles, where (perhaps surprisingly) we find a menu description that is in fact a slight natural variation of the traditional description. It also gives a barrier explaining one possible reason why previous attempts to concisely explain the strategyproofness of DA have failed.

We next propose a *new description of deferred acceptance*, which overcomes this barrier by deviating from traditional descriptions in crucial ways. Specifically, this new description can *explain the match of any single student* in the student-optimal outcome using a *school-proposing* algorithm. Despite the fact that this description is nearly as “easy to digest” as traditional descriptions, its strategyproofness comes clearly into focus. However, this new description comes at a cost: because it can only explain the match of one agent at a time, it is no longer clear why (for example) every agent will not receive the same match at the end, let alone why the outcome is stable. Nonetheless, we contend that strategyproofness is the most important property to emphasize to real participants (other properties, such as stability, are primarily the concern of policymakers, to whom these properties can be explained in more detail), and thus we view our description as an appealing real-life alternative over traditional descriptions of deferred acceptance.

To ground our study in reality, we conducted a laboratory experiment on a set of very simple mechanisms. We find significant evidence that menu descriptions can increase truth-telling rates (in a simple version of the median voting rule), providing a proof of concept for the real benefits of menu descriptions towards conveying strategyproofness.

*The authors thank Benjamin Enke, Xavier Gabaix, Nicole Immorlica, David Laibson, Shengwu Li, Assaf Romm, Ran Shorrer, Matt Weinberg, Leeat Yariv, Huacheng Yu, HUIJ’s JESC and Cornell’s BERG participants, and the 2020/2021 DIMACS Matching Reading Group participants 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. Thomas gratefully acknowledges the support of NSF CCF-1955205.

[†]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: clayton@princeton.edu.

1 Introduction

Mechanism design strives to achieve desirable outcomes in the presence of strategic behavior. An early cornerstone of the field was the advent of strategyproof social choice rules (i.e., those in which “truth-telling” is a dominant strategy), which eliminate any need for “perfectly rational” players to strategize. Unfortunately, real people are not perfectly rational, as standardly defined: significant and growing evidence shows that real people frequently do not play their theoretically dominant strategies in strategyproof scenarios¹.

Building above this cornerstone, recent works have investigated different potential explanations, including contingent-reasoning failures (Li, 2017) and expectations-based reference-dependent (EBRD) preferences (Dreyfuss et al., 2019). Several of these works have then proposed a more refined solution: *changing the interactive mechanism used to implement the social choice rule*.²

Figure 1: Two pairs of equivalent presentations of two mechanisms. Each mechanism is direct (in that agents simultaneously report all of their private information).

Traditional Description: The player who placed the <i>highest bid</i> will win the item. They will pay a price equal to the <i>second highest bid</i>.	Menu Description: Your “price to win” the item will be set to the <i>highest bid</i> placed by the <i>other</i> players. If your bid is higher than this “price to win”, then you will win the item and pay this price.
--	---

(a) Equivalent 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: The three votes will be sorted from low-est to highest, and the <i>middle vote</i> of the three will be elected.	Menu Description: The “obtainable candidates” will be the votes of the other two players, and all candidates between them. Out of these “obtainable candidates”, the one <i>closest to your own vote</i> will be elected.
--	---

(b) Equivalent descriptions of the median voting rule with **three** voters with single-peaked preferences. For each voter, voting for her favorite candidate is dominant. Each voter’s “menu” consists of all candidates between the votes of the other two voters.

Building yet another layer over these foundations, we investigate, theoretically and experimentally, the possibility that certain *descriptions* of the *same extensive-form mechanism* can better explain strategyproofness than others. Here, we mean “description” quite literally: we study a principled way to change the way a mechanism is *presented* without changing the outcome, its interactions with the agents, or any other aspect of it. We study this change of descriptions in a variety of contexts, and both theoretically and experimentally.

Different descriptions of the same mechanism. Consider the following three canonical mechanisms: a second-price auction for a single item, the median voting rule for voters with single peaked preferences, and the deferred acceptance matching rule. Each of these is strategyproof. But the difficulty of observing this strategyproofness varies wildly across the three, from a short classroom example for the second price auction, through a longer exercise for median voting, to a complex mathematical theorem for deferred acceptance.

In Figure 1, two equivalent descriptions of a second price auction and of median voting are given. The first set of descriptions, which we consider the “traditional” descriptions, simply convey the outcome of the mechanism. The second set of descriptions are intended to *expose and emphasize* the strategyproofness of the mechanism. Before we explain how, we mention that in a pre-registered lab experiment that we ran, we find higher rates of playing the dominant strategy in the second (new) description in Figure 1(b) than in the first (traditional) description in that figure (80 versus 70 percent, two-sided *t*-test *p*-value = 0.01, *N* = 100 observations each). In

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

²Informally, under these models, breaking up a “big decision” into “a sequence of small little ones”, or changing the timing of events to avoid disappointment, may alter behavior.

full disclosure, we find no such difference between the two descriptions for the simpler second price auction from Figure 1(a). See Section 2 for a detailed discussion of our experiment and results.

A major motivating question of this paper is **can such a “strategyproofness exposing” description be written for deferred acceptance³ as well?** The challenge of describing deferred acceptance to ordinary people is a particularly salient one: real public school districts use deferred acceptance to match students to schools, and students (and their parents) with no knowledge of economics or mechanism design must participate. From their point of view, likely the most important facet of this mechanism is its strategyproofness, yet school districts are rarely able to elucidate any sort of reason *why* the school choice mechanism should be strategyproof.⁴

Menu descriptions. Both of the “non-traditional” descriptions in Figure 1 share a common structure. For any player i (who reads the description), these descriptions contain two steps:

- (i) Using only the reports of *other players*, calculate a set of outcomes that player i might receive.
- (ii) Award player i her *favorite* outcome from this set (according to player i ’s report).

The set calculated in step (i) is called the *menu*, and we call a description with this structure a *menu description* for player i in the mechanism. When a mechanism is written this way, it is straightforward to observe strategyproofness: Player i ’s report cannot affect the menu in step (i), and reporting her “true type” will always select her favorite attainable outcome for her in step (ii).

It is no accident that both “non-standard” descriptions in Figure 1 have this structure. In fact, a mechanism is strategyproof if and only if it can be rewritten in this way (Hammond, 1979). Put another way, in any mechanism, the *menu presented to player i by reports t_{-i}* is the set of outcomes player i can receive when other players report t_{-i} ; Hammond (1979) showed that a mechanism is strategyproof if and only if each agent always receives her favorite outcome from her menu⁵.

Not all menu descriptions are created equally. While every strategyproof mechanism has a menu description, it is initially highly unclear *what such descriptions can look like*, as well as *how complicated such descriptions must be*. For some mechanisms, such as a second-price auction in Figure 1(a), a menu description can be provided almost entirely just by switching the order in which agent’s values are considered. For other mechanisms, such as the median voting rule in Figure 1(b), the menu description is non-trivially more complex, but still fairly understandable to a layperson. Yet for other mechanisms, such as deferred acceptance, it is far from clear how to proceed.

One approach to calculating the menu in any strategyproof mechanism f is the following: Given reports t_{-i} of the players other than i , calculate the outcome $f(t_i, t_{-i})$ (using any method for running the mechanism itself) for each possible report t_i of player i , and all of these outcomes precisely make up the menu. But this is clearly untenable as an approach to defining mechanisms for real people—even for a second price auction, this approach describes the mechanism as considering every possible hypothetical bid a player may place, remembering the price paid if the bidder wins at that hypothetical bid, and then picking the utility-maximizing hypothetical bid under the player’s report. Practitioners could never reasonably describe a mechanism with such an outline in settings such as school choice and expect people to get a sense for how the process works. So the real goal is to find *understandable* descriptions with *understandable* strategyproofness, and the real question is not “what can menu descriptions do”, but “what can ‘simple’ or ‘reasonable’ menu descriptions do?”

1.1 Impossibility result: the strategic complexity of deferred acceptance

Simple matching mechanisms and menus For another example of a mechanism where strategyproofness is easy to understand, consider the *serial dictatorship* (henceforth, SD) matching rule, which matches (say) *applicants*

³(Student-proposing) deferred acceptance is a matching rule that yields the student-optimal stable matching of students and schools. The matching is traditionally described via the “deferred acceptance algorithm”. Briefly, this algorithm has students iteratively “proposing” in order of their preferences over the schools, with schools “tentatively accepting” the best proposals they have seen so far (according to the “priority list” the school has over students).

⁴A common approach taken by real-life clearinghouses such as the NRMP (<https://www.nrmp.org/intro-to-the-match/how-matching-algorithm-works/>) is to rely on un-intuitive appeals to authority, such as:

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.

⁵Formally, “menu descriptions” of a strategyproof mechanism work as follows: for one player i , they calculate the menu using only t_{-i} , then allocate for player i her favorite outcome from that menu. On the other hand, “traditional descriptions” of a mechanism typically convey the final outcome of the mechanism (for all players) in as direct a way as possible. Thus, menu descriptions differ from traditional ones in two ways: the traditional description need not explicitly describe a menu, but a menu description only needs to describe the outcome for only one player.

to *institutions*⁶. In a direct implementation of SD, applicants are chosen in a fixed order, and when chosen, an applicant is permanently matched with her favorite (by her reported preferences) remaining “institution.” This “traditional” description of SD is already a menu description, and moreover, a menu description for every one of the players simultaneously.

The menu in a matching mechanism is an exceptionally natural object, as each applicant’s menu is always simply some subset of institutions. However, beyond SD, matching mechanisms are not typically described in a way that makes the menu apparent. Why is this the case? To delve into this question, we investigate (for two specific matching mechanisms of broad interest) *how much “traditional” descriptions must be changed in order to reveal an agent’s menu*.

A first step: the strategic simplicity of top trading cycles. We start by briefly considering a mechanism that initially appears quite difficult to understand, but for which *strategyproofness* turns out to be relatively simple. This mechanism is *top trading cycles* (henceforth, TTC). Roughly speaking, in TTC applicant match by trading (in cycles) the priorities institutions have for them. While TTC is not as ubiquitous as deferred acceptance, it is still used in practice and widely studied in theory.

Despite the apparent complexity of the TTC rule, one may give a fairly short and conceptually clear proof that TTC is strategyproof. This proof proceeds by considering, for an arbitrary player i , a “slightly modified” version of TTC in which *the menu of player i is calculated* (without using player i ’s report) *along the way to calculating the final matching*. Very briefly, this “modification” works by changing only the order in which the steps of TTC (the so-called “trading cycles”) are performed, but without changing how the steps actually work.⁷ This modified TTC first calculates the menu of player i , then matches i to her favorite institution on the menu, then proceeds to calculate the rest of the TTC outcome. Thus, a menu description of TTC can be constructed in a way that is surprisingly similar to how it can be constructed for a second price auction (in Figure 1(a))—by simply changing the order of the steps that calculate the outcome.⁸ If we describe this “slightly modified” TTC to an applicant instead of the similar traditional description, that applicant can understand the strategyproofness of TTC for themselves with comparative ease.

Deferred acceptance. We now proceed to our most important mechanism of study: deferred acceptance (henceforth, DA). Motivated by the above insight into TTC, we ask: would a similar approach give a simple argument for the strategyproofness of DA? That is, is there a “slightly modified” version of DA in which the menu of applicant i is calculated along the way to calculating the final matching? We prove that this cannot be. We show that *no small or even “reasonable”* modification of DA can reveal any agent’s menu along the way to describing the final matching.

A necessary condition for simplicity To formalize our impossibility result, we define a broad class of mechanism descriptions, to which any “small modification” of the traditional description of DA must surely belong. One way to motivate this is that the traditional description of DA is already one of the *most complex algorithms* one could possibly hope to explain to laypeople (e.g., school choice participants). We identify two major factors of DA which are crucial to this simplicity⁹:

- DA must take into account all applicants’ preference lists. However, it reads them in a very natural order: from favorite to least favorite (the order in which proposals are made in DA).
- DA *barely requires any bookkeeping* as it runs: it must keep track of a “tentative matching” (and perhaps a small amount of extra data¹⁰), but this is all; this is very little information “per applicant”, and virtually no more complex than the “final matching” that DA must compute.

⁶In all matching mechanisms that we consider, the only strategic players are the applicants. The institutions are non-strategic, and thus their “preferences” over the applicants are called “priorities.”

⁷In slightly more detail: after all possible trades not involving agent i are performed, every remaining institution is on agent i ’s menu (because i will create a cycle after pointing to any item). See Section 5 for details.

⁸One may suspect that this result somehow indicates that TTC is just an “objectively simple mechanism”. In Section 5.3, we delve further into the source of this simplicity result of TTC, and give a sense in which the above proof sketch captures the *precise way* in which the strategyproofness of TTC is easy (and it implies that TTC is quite complex from a “global perspective”).

⁹We emphasize that there is a crucial difference between what we aim to capture here and traditional notions of algorithm complexity: while traditional notions answer the question “given all of the inputs (e.g., preference lists) for the algorithm, how hard is it to run the algorithm and calculate the outcome?”, we target the question “with no inputs known yet, how hard is it to ex-ante internalize the algorithm and understand its qualitative behavior?”.

¹⁰For example, depending on implementation, the algorithm may keep track of to whom each applicant has already proposed.

We therefore consider an algorithm to be “far more complex than DA” if it either: 1) reads through applicants’ preference lists more than one time, or 2) stores vastly more information about the applicants’ lists than traditional DA. Either of these would yield a description much harder to grasp than the (already challenging) traditional one.¹¹ We thus consider these properties necessary conditions for the simplicity of a matching mechanism. Moreover, we consider (a necessary condition for) a “reasonably small modification” of DA to be a description that calculates the full matching of DA, while keeping this simplicity property.

An impossibility result Our first main result is that no description for DA can satisfy all of:

1. Calculates the menu for some player before accessing that player’s preferences.
2. Proceeds to calculate the final matching of all players.
3. Satisfies the above “necessary condition for simplicity.”

Thus, explaining strategyproofness via the menu is fundamentally incompatible with “simplicity” (desired for explaining DA to real people) and with “explaining the entire matching” (as traditional descriptions of DA do). In particular, any algorithm “nearly as simple” as DA that calculates the full matching cannot calculate a menu (that is, any description of DA satisfying the goal (2) and (3) cannot also satisfy (1)). This is the formal sense in which we claim that *no “small modification” of DA can reveal any participant’s menu*¹². This impossibility result gives striking evidence of the strategic complexity of DA: the way that traditional descriptions of DA work cannot reveal why the mechanism is strategyproof (in the canonical manner provided by menus). It also potentially offers a *formal reason* as to why previous attempts to provide a transparent explanation for the strategyproofness of DA have all failed: in a precise sense, *explaining the full matching* precludes explaining strategyproofness.

1.2 A simple menu description of DA nonetheless

A menu description of a different flavor. Given the stark impossibility barrier described above, we investigate the prospect of abandoning describing the full matching, and instead explaining the strategyproofness of the mechanism through the menu. Free from the need to describe the full matching, we present our second main result: *a novel, strikingly different description for DA*, presented in [Figure 2](#). Despite its differences from the traditional DA algorithm, we prove that [Figure 2](#) is indeed a menu description of DA (in particular, of *student-proposing* DA).¹³

For real-life clearinghouses wishing to emphasize strategyproofness, this new description of DA offers appealing distinct advantages. It is plausibly “digestible” by a layperson (very nearly as “digestible” as traditional descriptions of DA). It clarifies the role that an applicant’s rank-order-list plays in deciding their match (and the roll that submitting their true preference list plays in matching them to their highest-ranked obtainable school). And from the perspective of a student, it rephrases a process of “many different schools rejecting you” (a description which may highlight imagined “losses” or “opportunity costs”) into a process of “picking your best obtainable option.”¹⁴

Trade-offs. Another way to view our impossibility result for DA is as a formal investigation of *trade-offs*. Even for the simple mechanisms in [Figure 1\(a\)](#) and [Figure 1\(b\)](#), switching to the “strategically simpler” menu descriptions comes at a cost. For the second price auction, it is less clear in the menu description *why only one player can win the item*, and for median, it is initially very unclear *why each player can see the same description* (i.e., why the outcome chosen when the menu description is presented to one player is the same as when it is presented to another).

For DA, these trade-offs can become *severe*. For instance, in our menu description of DA presented in [Figure 2](#), it is highly unclear *why each player will be matched to a different institution* (or in the analogous menu description for the many-to-one setting of school choice, why no school will be matched to more students than its capacity), let alone why the matching will be stable. Another way to interpret our impossibility result for DA is that any “simple”

¹¹Indeed, this would make it much harder to mentally keep track of what next step could be consistent with a run so far.

¹²To begin to gain intuition into this result, consider the following: Like TTC, the DA algorithm is also independent of the execution order that the mechanism chooses (i.e., the order in which “proposals” are chosen). In contrast to TTC, however, allowing all other players to propose “before” player i proposes does *not* suffice to calculate the menu (briefly, because player i “proposing to” an institution can result in a “rejection cycle” that leads to i being rejected from that institution).

¹³Strictly speaking, this description does not precisely match our simplicity condition (as it replaces reading the applicants’ lists in order with reading the institutions’ lists in order). We in fact were able to also construct a menu description for DA that strictly adheres to our necessary condition for simplicity (and gives no information on other agent’s matching, of course). However, unlike our proposed menu description in [Figure 2](#), this description is very delicate and involved, more than anything underscoring the fact that our necessary condition for simplicity really is far from a sufficient condition.

¹⁴We are in the process of planning and executing an experiment to test the rates of straightforward behavior (i.e., “truth telling”) with this description compared to traditional ones. We ran a short pilot of this DA experiment on January 23, 2022.

Figure 2: Descriptions of the student-optimal (i.e., applicant-optimal) stable outcome in a school matching environment (both stated in terms of the traditional deferred acceptance algorithm). These two descriptions are equivalent according to [Theorem 6.1](#). We emphasize that in the menu description given here, the matching of “other students” used to calculate one student’s menu can be almost unrelated to the “final matching” that those students will receive by student-proposing DA. Indeed, if it were related, it would have given an avenue to calculating both the menu and the outcome in a simple manner, in contrast to our impossibility result.

Traditional Description: The students will be admitted to schools according to the <i>student-proposing deferred acceptance</i> rule.	Menu Description: All students <i>except you</i> will be “hypothetically” paired to schools according to the <i>school-proposing deferred acceptance</i> rule. You will “earn admission” at every school that is hypothetically paired to a student of lower priority than yourself at that school. You will be admitted to your <i>highest ranked</i> school where you “earned admission”.
---	---

menu description that “works like traditional descriptions of DA” *must* have this property, that is, *cannot* explain why other applicants will not match to the same institution as the player whose menu is described. (Formally, any description of DA satisfying goals (1) and (3) on Page 4 cannot also satisfy goal (2).) Thus, one cannot have their cake and eat it too: such a description must choose to explain strategyproofness (via menus) or to explain that the result is a matching, but not both. Traditional descriptions of DA may have been on the “wrong corner solution” of this trade-off. Choosing to explain strategyproofness over explaining that the result in a matching provides a direct benefit to applicants participating in DA.

Menu-based observations. So far, we have discussed the prospect of switching the presentation of a mechanism to a menu description to make strategyproofness easy to see. The concept of a menu also facilitates a completely different approach to conveying strategyproofness. To see this for the simple case of a second-price auction, consider the following “observation” or “advice,” which may *accompany* a traditional description of a second-price auction (such as the one in [Figure 1\(a\)](#)):

Observe that if you win the item, you will pay a price equal to the highest bid placed by any *other bidder*. Your own bid cannot affect this price. Moreover, you will win the item if and only if this price is lower than your bid.

This advice effectively identifies the menu (in terms of the “price to win”) via an easily observable and describable feature of the traditional description (the highest bid of another player). It then argues that you will always get your favorite outcome from the menu (either winning the item at its price, or winning nothing and paying nothing). Thus, if an agent understands this advice, and understands why any menu description is strategyproof, then they will also understand why the traditional description of a second price auction is strategyproof.¹⁵

A third interpretation of our impossibility result for DA is a strong sense in which such “menu-based observations” are *impossible* in DA. In fact, our result shows that any procedure that calculates the menu “along the way” to the final matching must be hopelessly complex compared to traditional descriptions (that is, any description that satisfies goals (1) and (2) on Page 4 cannot also satisfy goal (3)). Thus, no “simply observable feature” of the way that the traditional deferred acceptance algorithm works will suffice to describe the menu. In other words, any additional “observation” (to be added to the traditional description of DA) that can explain strategyproofness must be considerably more complex than the entire mechanism description that precedes it.

Given this, it is not surprising (in retrospect) that real-life clearinghouses using DA have failed to fully explain *why* DA is strategyproof—any description of the entire matching will make this difficult. It is our recommendation that want to convey strategyproofness first and foremost, to completely change their description of DA (but not change the implementation in any way) and replace it with our menu description from [Figure 2](#) or some variant thereof.

1.3 Related work

A major cornerstone of the contemporary “strategic simplicity” program in mechanism design was [Li \(2017\)](#), who introduces OSP mechanisms to formally explain why certain mechanisms implementing a social choice rule

¹⁵In contrast to this menu-based explanation/observation, a “more traditional” explanation of strategyproofness may go as follows: observe that bidding *less* than your value for the item can only result in you losing it when you could have won it for a profit, while bidding *more* than your value for the item can only result in you paying more for it than it is worth to you.

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, as [Li \(2017\)](#) showed for TTC, and as [Ashlagi and Gonczarowski \(2018\)](#) showed for DA, not all social choice functions have OSP mechanisms,¹⁶ which seems to limit 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) our results can actually achieve what essentially amounts to a *strongly* obviously strategyproof (a concept defined by [Pycia and Troyan \(2021\)](#)) description for the player at which they are aimed. This ability to target only one player fortunately brings the entire class of strategyproof social choice rules (not just those that are OSP implementable) back under our scope. Correspondingly, we achieve “strategically simple” descriptions of TTC and even DA.

The approach that we harness to proving the strategyproofness of TTC is so fundamental (and has been the “go-to intuition” for strategyproofness of TTC in the mind of one of the authors of this paper for so many years) that we are wary of claiming that it originates in this paper. [Leshno and Lo \(2021\)](#) use the menu in TTC (which they call a “budget set”) to calculate welfare; they also mention the fact that the budget set being independent of the player’s report implies strategyproofness. (As we note in [Section 3](#), as a general characterization of strategyproof mechanisms this fact goes as far back as [Hammond \(1979\)](#).) However, they do not discuss concrete ways of seeing the menu from the description of TTC, and their overall viewpoint is “ex-post verification” of the matching instead of “ex-ante understanding” of the matching algorithm.

Our theoretical analysis, to the best of our knowledge, is the first to analyze different ways to *describe* the menu of a mechanism. We offer a definition of simplicity of menus, which, despite borrowing concepts from computer science, is aimed at capturing *not* how complex it would be for a machine to perform the menu calculation ex-post (after type reports are in), but rather in terms of how simple it would be for a human reader to grasp the description of the menu ex-ante. There is some empirical overlap with the recent working paper [Katušćák and Kittsteiner \(2020\)](#), who earlier but independently suggested describing a matching mechanism (specifically, TTC) via menus. They use a single menu description that is very close to that of [Example 5.1](#), (essentially, calculate the menu by iterating over possible reports and running the standard TTC description for each report to see which outcomes are possible). In contrast, our main focus is finding, among the rich landscape of menu descriptions for each of a variety of social choice rules, and in particular those that satisfy what we called above our “necessary criterion for simplicity” (which, in particular, precludes iterating and running the standard description more than once). Further from us are papers that explore *advice* in strategyproof mechanisms: [Masuda et al. \(2020\)](#) consider the VCG auction with three bidder and two identical (“multi unit”) items. They describe the auction results and prices via a traditional description. Their main experiment centers around including (or not including) *advice* about strategyproofness, although they carefully phrase the advice to attempt to avoid the “experimenter demand effect.”

There is a growing literature within computer science looking at the structural complexity of menus (i.e., how many different entries are offered on the menu) in the context of the revenue-maximizing monopolist problem with one buyer ([Hart and Nisan, 2013](#); [Daskalakis et al., 2017](#); [Babaioff et al., 2017](#); [Saxena et al., 2018](#); [Gonczarowski, 2018](#)). Most relatedly, [Dobzinski \(2016\)](#) considers the properties of menus in multi-buyer mechanisms in some detail. While not directly related, we discuss some connections to this work in [Section 5.3](#). In particular, we prove that in contrast to our framework, the “taxation complexity” framework of this literature cannot separate TTC and DA.

2 Motivational Experiment

2.1 Experiment flow

Our experiment consists of two parts, for two mechanisms, in this order: voting in elections (henceforth, Median) and bidding in a second-price auction (henceforth, Auction). In each part, respondents are independently randomized into either Traditional (T) or Menu (M) treatments. T and M differ only in some of the language (which, in Median, includes visual illustrations) that respondents read and use to interact with the mechanism.

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. Respondents who, after three attempts,

¹⁶This was even shown for the median mechanism by [Bade and Gonczarowski \(2016\)](#).

fail to correctly answer a comprehension question are given the answer. They must enter it to proceed.¹⁷ All four setup screens are designed to convey, and confirm understanding of, the working of the mechanism—from inputs by the participant and other (computerized) players to outcomes and 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, they are reminded (1) how the mechanism works and (2) that the system randomly picks the votes of other two voters or bids of other four bidders.

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. The experiment is programmed on Otree (Chen et al., 2016).

2.2 Traditional and Menu Treatments

Figure 3 (on the pre-bibliography figures page) reproduces the main screen for Median 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. Its M version is shown below the T screen.

As the figure shows, respondents have 30 seconds to vote; can look again at (“click to expand”) examples; and are reminded that the other two votes are chosen from a uniform distribution.

The main screen in Auction is conceptually similar to the main Median screen in Figure 3, but: (1) “Your earnings table this round is: [table]” is replaced with “Your money payout in this round is: [payout]”; (2) “To select a vote ... click below:” is replaced with “Place your bid here: \$_____”; (3) “Remember ... Then:” is replaced with “Remember, the system will randomly pick the bids of four other “bidders”. Then:”; and (4) the T/M text to the right of the thick black line is replaced with roughly the text in Figure 1.¹⁸

2.3 Data

We conducted our experiment on February 4–5, 2021. 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.¹⁹ Median respondent age is 29; 51% are female; and median completion time is 20.6 minutes.²⁰ Subjects received \$2 for participation and earned an additional \$6.33 on average through their voting and bidding.

2.4 Results

Figure 4 presents our results. In Median, under T, participants vote for their highest-earning candidate 70% of the time; under M they do so 80% of the time. The difference, of 10 percentage points, is arguably large; a two-sided *t*-test for equality of means yields *p*-value = 0.01. Participants also earn 6 percent more in M (\$3.00) than in T (\$2.83), however this result is not statistically strong (*p* = 0.10). Comparing first to last five rounds to explore learning, fatigue, or boredom, we find essentially no difference.

In Auction, we find essentially no difference across the

	Trad. (T)	Menu (M)	<i>p</i> -value
Median Voting			
S.F. (frac.)	0.70 (0.03) <i>N</i> = 100	0.80 (0.03) <i>N</i> = 100	0.01
Earning (\$)	2.83 (0.07) <i>N</i> = 100	3.00 (0.07) <i>N</i> = 100	0.10
Second Price Auction			
S.F. (frac.)	0.37 (0.04) <i>N</i> = 100	0.34 (0.03) <i>N</i> = 100	0.55
Earning (\$)	3.40 (0.26) <i>N</i> = 100	3.40 (0.24) <i>N</i> = 100	0.98

Figure 4: Experiment Results by Treatment. S.F.: fraction of rounds with straightforward behavior. In T, S.F. means voting for the “ideal number”. In M, S.F. means bidding within \$0.10 of the private value. Earnings: participants’ average dollar amount earned across 10 rounds. Standard errors are in parentheses. *p*-values: two-sample, two-sided equality-of-means *t*-test.

¹⁷In Median, there are six comprehension questions. 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.

¹⁸T text: “You will win the auction if you placed the *highest bid*. / If you win, you will *pay a price* equal to the *second highest bid*”. M text: see Figure 1, but “item” is replaced with “auction” and “players” with “bidders.”

¹⁹Of 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).

²⁰Our (pre-registered) intention was to recruit a total of 200 participants. Due to miscommunication, our RA kept recruiting until we had 200 *completes*. Excluding these additional completes does not affect our results more than trivially; the Median *t*-test *p*-value reported in Figure 4 changes from 0.01 to 0.02.

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 (the figure shows S.F. results for $d = \$0.10$).

We view our results as providing sufficient motivation to keep developing our theory. While we can only speculate as to why our menu description “worked” in Median and not in Auction, we note that, to our knowledge, ours is the first attempt to develop menu descriptions for these mechanisms. Some traditional mechanism descriptions have been refined over years; we hope to similarly keep improving our new menu descriptions.

Our experiment shows both the promise and the challenges of menu descriptions. Ours are significantly longer than traditional descriptions; they also appear more difficult to grasp (as suggested by comprehension-question statistics). At the same time, our Median results suggest them as a worthy research direction, which we explore theoretically for the remainder of the paper.

3 Theoretical Preliminaries

This paper studies strategyproof social choice functions in different economic environments. We define all these terms fully in [Appendix A](#). Briefly, an *environment* has some set of possible outcomes A , and n agents (also known as players) with their own desires and incentives regarding the outcomes. These are modeled by “types” $t_i \in \mathcal{T}_i$ of the agents, which induce preferences $\succ_i^{t_i}$ over the outcomes.

Definition 3.1. A social choice function in some environment is any mapping $f: \mathcal{T}_1 \times \dots \times \mathcal{T}_n \rightarrow A$ from the types of all players to an outcome.

A social choice function is strategyproof if, for any $t_i, t'_i \in \mathcal{T}_i$ 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 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 lists. Only the applicants are strategic, where the institutions have fixed “priorities”. We study the canonical mechanisms of serial dictatorship (SD), top trading cycles (TTC), and deferred acceptance (DA) (all fully defined in [Appendix A](#)).

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 under the indifference of bidder i . (For a formal definition, see [Definition A.2](#).) In matching markets, agents are assumed to only care about their 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 as $[a]_i$, and the set of *i-outcomes* by A_i .

3.1 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 menu²¹ 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 3.2. For any social choice function f , the menu $M_{t_{-i}}$ presented to player i by types $t_{-i} \in \mathcal{T}_{-i}$ is the subset of all *i-outcomes* $a \in A_i$ such that there exists some $t_i \in \mathcal{T}_i$ such that $f(t_i, t_{-i}) \in a$. That is,

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

Theorem 3.3 ([Hammond \(1979\)](#)). A social choice function f is strategyproof if and only if each player i always receives (one of) their favorite *i-outcomes* from their menu (that is, for each $t_{-i} \in \mathcal{T}_{-i}$ and $t_i \in \mathcal{T}_i$, we have $f(t_i, t_{-i}) \succeq_i^{t_i} x$ for any $x \in M_{t_{-i}}$).

Proof. Suppose f is strategyproof and fix $t_{-i} \in \mathcal{T}_{-i}$. For any $t_i \in \mathcal{T}_i$, 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 their menu, then player i will always prefer submitting type t_i at least as much as any t'_i , so f is strategyproof. $\square$

²¹This definition has gone by a few different name: [Hammond \(1979\)](#) says that the menus “decentralize the mechanism”, and [Guesnerie \(1981, 1998\)](#) refer to [Theorem 3.3](#) as the “taxation principle”. [Leshno and Lo \(2021\)](#) call them “budget sets”. We follow the algorithmic mechanism design literature ([Dobzinski \(2016\)](#) and follow-ups) in calling these sets “menus”. This definition is distinct from a different definition of menu sometimes used in market design papers such as [Mackenzie and Zhou \(2020\)](#); [Bó and Hakimov \(2020\)](#).

4 Formal Model

4.1 Descriptions of Mechanisms

Mechanisms can be described to agents in a variety of ways. The present paper alone contains a range of examples: textual descriptions, examples, and algorithmic pseudocode. To abstract over all these possibilities, yet formally reason about the complexity of 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)²². Note, however, that the interpretation is very different than that of an extensive-form mechanism: The extensive-form description is a way of calculating the social choice function using the (directly reported) types of the players (and not an interactive process where the agents may act multiple times).

Definition 4.1. *An extensive-form description in some social choice environment is defined by a directed graph on some set of nodes V . There is a single source node $s \in V$, and the vertices of V are organized into “layers” $i = 1, \dots, L$ 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 node v , let $S(v)$ denote the edges directed out of v . Each node v is labeled with some player i and some function $\ell_v: \mathcal{T}_i \rightarrow S(v)$ from types of player i to edges out of v . (We also allow nodes with out-degree 1 which are not labeled with any player.)*

Any extensive-form description D defines a unique social choice function f_D as follows: to compute $f_D(t_1, \dots, t_n)$, start in the source vertex s , and construct a path where at every vertex v labeled by player i and ℓ_v the next vertex on the path follows the edge $\ell_v(t_i)$ exiting v . (We call this path the execution path on inputs $(t_1, \dots, t_n)$.) Eventually a sink vertex w labeled by some outcome a must be reached. This defines the result of f_D , i.e., $f_D(t_1, \dots, t_n) = a$. 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 a extensive-form description is the logarithm (base 2) of the maximum number of nodes in any layer of the node graph V .

We sometimes abuse notation and refer to an extensive-form description simply as a “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 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 and simply labeling each type profile with the right outcome²³. 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 Simple Descriptions of Matching Mechanisms

Each of the three matching mechanisms we consider (SD, TTC, and DA) are used in practical matching markets, and explained to real people with no background in economics. As discussed in the introduction, we identify two crucial properties which all of these mechanisms share which facilitate this explainability: 1) they only consider the preferences of applicants in a fixed, natural order (from favorite to least favorite), and 2) they require very little “book-keeping” as they run (little more than just the “book-keeping” required to remember a single matching).

Definition 4.2. *In a matching environment, an extensive-form description D is preference-read-once if it satisfies the following: For every agent i and every possible execution path through D , let $v_1, v_2, \dots, v_k$ denote the nodes along the execution path that query agent i . Then, for $j = 1, \dots, k$, the function $\ell_{v_j}: \mathcal{T}_i \rightarrow S(v_j)$ (which decides the next node in the execution path) depends only on the j th institution on agent i ’s preference list (possibly an “empty institution” if their list has fewer than j institutions on it).*

In a matching environment with n applicants and n institutions, an extensive-form description is preference-linear if it is preference-read-once, and additionally requires memory at most²⁴ $\tilde{O}(n)$.

²² 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 a extensive-form description is a finite automaton where state transitions are given by querying the types of players.

²³ 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).

²⁴ The notation $\tilde{O}(n)$ means $O(n \log^k n)$ for some constant k .

Note the definition of preference-read-once 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 $\tilde{O}(n)$ is exactly (up to lower order terms) the number of bits of memory required to describe a single matching (for instance, because there are $n! = 2^{O(n \log n)}$ distinct matchings on n applicants and positions). The name “linear” 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 typically defined using preference-linear descriptions:

Observation 4.3. *SD, TTC, and DA each have preference-linear extensive-form descriptions.*

4.3 Menu-Revealing Descriptions

Recall our definition of a “menu description” from [Section 1](#): A menu description 1) computes the menu presented to player i (using only t_{-i}), then 2) assigns i to their favorite outcome from the menu (according to t_i). For clarity, when we need to distinguish this with our next definition, we will refer to menu descriptions as *menu-only descriptions*²⁵. We enhance the definition of menu-only descriptions to *menu-revealing descriptions* in order to provide a lens into the formal simplicity of menu descriptions (as well as their relationship to traditional descriptions, and the tradeoffs involved with each).

Definition 4.4. *For any strategyproof social choice function f , a menu-revealing 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 $M_{t_{-i}}$ presented to player i by t_{-i} .*
2. *Using $t_i \in \mathcal{T}_i$, agent i is guaranteed their favorite i -outcome from $M_{t_{-i}}$.*
3. *Using both t_i and t_{-i} , calculate the full outcome $f(t_i, t_{-i})$.*

A menu-revealing extensive-form description of f for player i is an extensive-form description with some layer $k \in \{1, \dots, L\}$ such that: 1) every query in every node preceding layer k is made to some player other than i , and 2) every node v in layer k is labeled with some menu $M(v)$, such that if v is on the execution path on inputs $t_{-i} \in \mathcal{T}_{-i}$, then $M(v) = M_{t_{-i}}$.

In the most optimistic cases, such as a second-price auction, a menu-revealing description can be given for any player without increase the difficulty of understanding the mechanism. This is evidence both that the strategyproofness of the rule itself is easy to grasp, and that mechanism designers who wish to emphasize strategyproofness by portraying the mechanism in terms of menus do not face a strong cost of added complexity (an illustration of this for a second price auction is given in [Appendix G, Figure 9](#)).

5 Simple Menu-Revealing Mechanisms

Our goal in this section is to find *simple* menu-revealing descriptions, or prove that none exist. First, we give a *non-simple* menu-revealing description of a matching mechanism as a baseline:

Example 5.1 (A universal *non-simple* menu-revealing description for matching²⁶). *Consider any strategyproof matching mechanism f for n applicants and n institutions, and fix an applicant i . For any institution h , let $\{h\}$ denote the preference list of applicant i which ranks only h . For any extensive-form description D of f , consider the following procedure:*

1. *Given types $\succ_{-i}$, execute D on types $(\{h\}, \succ_{-i})$ for each institution h separately. If i is matched to h at the end of the corresponding run, then h is on the menu (by strategyproofness).*
2. *Match applicant i to their favorite institution from their menu, according to $\succ_i$.*
3. *Calculate the full matching by starting from scratch and using D to compute $f(\succ_i, \succ_{-i})$.*

²⁵We note the subtlety behind these definitions: a “menu description for player i in a mechanism” is a description *only* of one player’s menu (and how their outcome is picked from the menu, which (for strategyproof mechanisms) is determined only by the environment and not by the mechanism). On the other hand, a “menu-revealing description of a mechanism for player i ” is a complete description of the mechanism, which additionally reveals i ’s menu “along the way”.

²⁶This way of finding a menu description for any strategyproof matching rule was independently described by [Katušćák and Kittsteiner \(2020\)](#).

This menu description is sharply less intuitive than standard descriptions of mechanisms such as DA and TTC. Any information about agent i 's menu must be acquired “one institution at a time” by checking a separate instance of D , and there is no clear relationship between the match of i from step (2) and the final overall matching from step (3). Correspondingly, this strongly breaks our necessary criterion for simplicity [Definition 4.2](#).

5.1 Preference-linear menu-revealing descriptions for SD and TTC

We start with perhaps the simplest mechanism we consider in this paper, SD:

Fact 5.2. *For each agent i , SD has a preference-linear, menu-revealing description for player i . Moreover, there is a single preference-linear extensive-form description which constitutes a menu-revealing description for each player simultaneously.*

Perhaps somewhat surprisingly, TTC has a comparatively simple menu-revealing description as well. Unlike SD, this description can only work for a single player at a time²⁷. The *only goal* of the menu-revealing description is to explain the mechanism to one player, but it must explain both that one player's menu as well as the entire matching. We achieve this using the fact that TTC is *independent of the order in which trading cycles are executed*.

Theorem 5.3. *For each player i , TTC has a preference-linear, menu-revealing, extensive-form description for i .*

Proof. We will use the fact that by [Lemma A.6](#), TTC (as formally described in [Definition A.5](#)) is independent of the order in which we choose to “eliminate trading cycles”.

Start running TTC, and as long as possible, choose to only eliminate cycles that do not include i . This stage of TTC does not depend on i 's preferences,²⁸ 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 eliminated cycle *must* involve i , she would get this institution.

Afterwards, continue running TTC as usual using the rest of preferences to construct the final matching. This entire process constitutes a valid run of TTC, and thus correctly computes the final matching. $\square$

Note that, in addition to remaining preference-linear, this menu-revealing description changes TTC *only by modifying the order in which steps are executed*.

5.2 Impossibility result for DA

Theorem 5.4. *For any player i , there exist priorities of the institutions such that no preference-linear, menu-revealing description of DA exists. In fact, any menu-revealing extensive-form description for DA which is preference-read-once requires $\Omega(n^2)$ memory.*

The proof is in [Appendix B](#). Here, we give some preliminary intuition. The proof constructs (for a fixed “worst case” set of institution priorities) a very large set of applicant preferences such that: 1) to learn the menu of player i , a large fraction of each other applicant's preference list must be read, and 2) to correctly compute the final matching, all of the information from these applicant's lists must be remembered in full. The $\Omega(n^2)$ lower bound comes from this “large amount of information” which must be remembered (see [Figure 7](#)).

This theorem says that the following three requirements are not compatible for any algorithm which computes the matching of DA: preference-read-once, “reasonable” memory requirements, and menu-revealing. None of these three assumptions can be dropped. First, the traditional description is preference-read-once and requires low memory, but does compute the menu. Second, *any* function of the types can be computed with a preference-read-once description with $O(n^2)$ memory. Finally, we can show that an $\tilde{O}(n)$ memory menu-revealing description of DA exists which makes *just two* passes through the preference lists of all players (one pass before, and one pass after the menu is computed)²⁹.

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. For TTC, you have 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 the typical description of DA* will suffice for witnessing the strategyproofness of DA through menus.

²⁷This attribute is related on a technical level to the fact that SD is OSP ([Li, 2017](#)) and even SOSP ([Pycia and Troyan, 2021](#)), while TTC is not OSP-implementable.

²⁸Metaphorically, this can be thought of as running TTC but during this first stage not letting agent i point to any institution.

²⁹Towards this goal, we show in [Appendix F](#) that a preference-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 can be used to compute the rest of the matching.

5.3 Discussion: The Source of Simplicity in TTC

Comparing our finding contrasting DA and TTC, we ask *what is the true source of the “strategic simplicity” of TTC?* To probe into this question, consider a variant of the notion of menu-revealing description in which the description *cannot access* t_{-i} during stage 3 of Definition 4.4 (the stage where the matches of agents other than i is calculated). 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 description meeting this restricted definition still calculate both the menu and the matching in TTC? We prove that it cannot³⁰. We formulate the precise meaning of this result (and investigate the analogous questions for SD and DA) in Appendix C. Very briefly, this result holds because when agent i in TTC selects an item from their menu (in Theorem 5.3), this choice can dramatically effect the “landscape” of remaining applicants and institutions that will determine the rest of the matching, and in particular the *order* in which agents will be matched. Thus, the preference-linear menu-revealing description of TTC very precisely captures what is strategically simple about TTC.

6 A Simple Menu-Based Description of Deferred Acceptance

In Section 5.2, we give a precise sense in which the strategyproofness of DA is hard to convey. Moreover, we single out the requirement of *describing the entire matching* as a barrier to finding “better,” strategyproofness-exposing descriptions.

Prompted by this formal barrier, we looked for a menu description of DA which does not seek to explain the entire matching. To our surprise, we found a very appealing description, which call Strategyproofness-Exposing Deferred Acceptance (SEDA). This is given in Figure 5 (and also Figure 2 in the introduction). In terms of pure algorithmic simplicity, SEDA is quite comparable to traditional descriptions of DA; in fact, it simply adds a very straightforward “menu calculation and matching” step on top of (a traditional description of) DA.³¹ In terms of simplicity of understanding strategyproofness, though, the difference from traditional descriptions of DA is stark.

Fig. 5 . SEDA: Strategyproofness-Exposing Deferred Acceptance description for an applicant $d \in \mathcal{D}$

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 $d \succ_h \mu_{-d}(h)$.
 Match d to d ’s best ranked institution in M .

From the point of view of an applicant, this description is quite different from traditional descriptions of the same mechanism, i.e., 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. In SEDA, the situation completely reverses: strategyproofness is easy to observe (or derive, by explaining that one’s true preferences will always pick the best alternative from the menu). On the other hand, the reasoning behind why this procedure produces a matching (let alone a stable matching) becomes quite complex.³² In adopting a description along the lines of Figure 5 (by which we mean similar to the menu description from Figure 2), a clearinghouse could shift the “burden of complexity” from deducing strategyproofness (a primary concern of the applicants) to deducing that the result is a stable matching (an objective which primarily concerns the clearinghouse or policymakers, rather than the applicants themselves). Formally, our theorem giving the correctness of SEDA is the following:

Theorem 6.1. *SEDA describes the applicant-optimal stable matching. In particular, if every applicant is assigned to an institution according to Figure 5, then the result is the applicant-optimal stable matching (the same as in applicant-proposing DA).*

In other words, SEDA is a menu-only description of DA; we prove this in Appendix B. All told, SEDA provides a complete, simple, and accurate picture of how the match of each applicant is calculated, and in stark contrast to traditional descriptions one can see for themselves why this process is strategyproof. We thus contend that SEDA could provides a viable and useful approach to describing real-world stable matching algorithms.³³

³⁰In particular, this shows that the “pointing graph” in the proof of Theorem 5.3 just before agent i is matched does not contain *nearly enough* information calculate the rest of the matching.

³¹Interestingly though, note that *institution-proposing* DA must be used here to calculate one applicant’s outcome in the *applicant-proposing* DA outcome.

³²Indeed, the proof of Theorem 6.1 below is as complex as certain known proofs that DA (described traditionally) is strategyproof (such

Time left to complete this page: 0:01

ROUND 2

Your earnings table this round is:

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

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

[Submit vote](#)

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

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

► For example: (click to expand)
Remember: the other two citizens' votes are chosen at random, with each vote between 1 and 5 equally likely for each citizen.

(a) T treatment: entire screen

The "obtainable numbers" will be the votes of the two other citizens, and all the numbers between them.

Out of these "obtainable numbers", the one *closest to your own vote* will be elected.

(b) M treatment: changed text

Figure 3: Median Round Screen. The participant chose number 2 but has not yet submitted their vote.

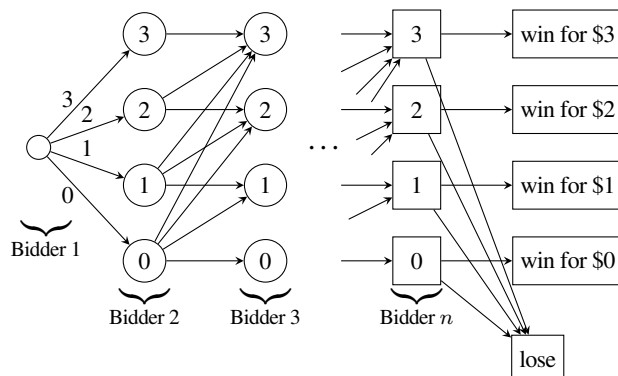


Figure 6: An extensive-form menu description for player n in a second price auction (i.e. an extensive-form formalization of the menu description in Figure 1(a)). The mechanism keeps track of only the highest bid seen so far. The memory required by this description is proportional only to the number of distinct bids that agents can place, and does not depend on the number of agents; this gives one sense in which this description is simple. The last layer is labeled with bidder n 's menu, which is defined by the "price they must pay to win the item".

References

- Itai Ashlagi and Yannai A Gonczarowski. 2018. Stable matching mechanisms are not obviously strategy-proof. *Journal of Economic Theory* 177 (2018), 405–425. Originally appeared in 2015.
- Itai Ashlagi, Yash Kanoria, and Jacob D. Leshno. 2017. Unbalanced Random Matching Markets: The Stark Effect of Competition. *Journal of Political Economy* 125, 1 (2017), 69 – 98. <https://EconPapers.repec.org/RePEc:ucp:jpolec:doi:10.1086/689869>
- Sepehr Assadi. 2020. Lecture 2: The Maximum Bipartite Matching Problem. (2020). <https://people.cs.rutgers.edu/~sa1497/courses/cs671-f20/lec2.pdf> Lecture notes for Rutgers course CS 671: Graph Streaming Algorithms and Lower Bounds.
- Moshe Babaioff, Yannai A. Gonczarowski, and Noam Nisan. 2017. 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*. 869–877. <https://doi.org/10.1145/3055399.3055426>
- Sophie Bade and Yannai A Gonczarowski. 2016. Gibbard-Satterthwaite success stories and obvious strategyproofness. In *EC*.
- Inácio Bó and Rustamdjan Hakimov. 2020. Pick-an-object mechanisms. *Available at SSRN* (2020).
- Yves Breitmoser and Sebastian Schweighofer-Kodritsch. 2019. *Obviousness Around the Clock*. Rationality and Competition Discussion Paper Series 151. CRC TRR 190 Rationality and Competition. <https://ideas.repec.org/p/rco/dpaper/151.html>
- Linda Cai and Clayton Thomas. 2019. Representing All Stable Matchings by Walking a Maximal Chain. (2019). <https://arxiv.org/abs/1910.04401> Mimeo.
- Daniel L. Chen, Martin Schonger, and Chris Wickens. 2016. oTree—An open-source platform for laboratory, online, and field experiments. *Journal of Behavioral and Experimental Finance* 9, C (2016), 88–97. <https://doi.org/10.1016/j.jbef.2015.12.00>
- Yan Chen and Tayfun Sönmez. 2006. School choice: an experimental study. *Journal of Economic theory* 127, 1 (2006), 202–231.
- Vincent P Crawford and Elsie Marie Knoer. 1981. Job matching with heterogeneous firms and workers. *Econometrica: Journal of the Econometric Society* (1981), 437–450.
- Constantinos Daskalakis, Alan Deckelbaum, and Christos Tzamos. 2017. Strong Duality for a Multiple-Good Monopolist. *Econometrica* 85, 3 (2017), 735–767.
- Gabrielle Demange and David Gale. 1985. The strategy structure of two-sided matching markets. *Econometrica: Journal of the Econometric Society* (1985), 873–888.
- Gabrielle Demange, David Gale, and Marilda Sotomayor. 1986. Multi-Item Auctions. *Journal of Political Economy* 94, 4 (1986), 863–872. <https://doi.org/10.1086/261411> arXiv:<https://doi.org/10.1086/261411>
- Shahar Dobzinski. 2016. Computational Efficiency Requires Simple Taxation. In *FOCS*.
- Bnaya Dreyfuss, Ori Heffetz, and Matthew Rabin. 2019. *Expectations-based loss aversion may help explain seemingly dominated choices in strategy-proof mechanisms*. Technical Report. National Bureau of Economic Research.

as that implied by Hatfield and Milgrom (2005)).

³³We mention here that Theorem 6.1 extends to many-to-one matching markets. Specifically, to adapt Figure 5, 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” they have available, and can intuitively be understood in terms of the “available seats” proposing to the applicants. All of the lemmas required for the proof of Theorem 6.1 hold in this setting.

- Yannai A. Gonczarowski. 2018. 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*. 123–131. <https://doi.org/10.1145/3188745.3188786>
- Roger Guesnerie. 1981. On taxation and incentives: further remarks on the limits to redistribution. (1981).
- Roger Guesnerie. 1998. *A contribution to the pure theory of taxation*. Number 25. Cambridge University Press.
- Pablo Guillen and Rustamdjan Hakimov. 2018. The effectiveness of top-down advice in strategy-proof mechanisms: A field experiment. *European Economic Review* 101 (2018), 505–511.
- Faruk Gul and Ennio Stacchetti. 1999. Walrasian equilibrium with gross substitutes. *Journal of Economic theory* 87, 1 (1999), 95–124.
- Faruk Gul and Ennio Stacchetti. 2000. The English auction with differentiated commodities. *Journal of Economic theory* 92, 1 (2000), 66–95.
- Dan Gusfield and Robert Irving. 1989. *The stable marriage problem: Structure and algorithms*. The MIT Press, Cambridge, MA.
- Rustamdjan Hakimov and Dorothea Kübler. 2019. *Experiments on matching markets: A survey*. Technical Report. WZB Discussion Paper.
- Peter J Hammond. 1979. Straightforward individual incentive compatibility in large economies. *The Review of Economic Studies* 46, 2 (1979), 263–282.
- Sergiu Hart and Noam Nisan. 2013. The menu-size complexity of auctions. In *the 14th ACM Conference on Electronic Commerce (EC)*.
- Avinatan Hassidim, Déborah Marciano, Assaf Romm, and Ran I Shorrer. 2017. The mechanism is truthful, why aren’t you? *American Economic Review* 107, 5 (2017), 220–24.
- John William Hatfield and Paul R Milgrom. 2005. Matching with contracts. *American Economic Review* 95, 4 (2005), 913–935.
- Nicole Immorlica and Mohammad Mahdian. 2005. 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*. 53–62. <http://dl.acm.org/citation.cfm?id=1070432.1070441>
- John H Kagel and Dan Levin. 1993. Independent private value auctions: Bidder behaviour in first-, second-and third-price auctions with varying numbers of bidders. *The Economic Journal* 103, 419 (1993), 868–879.
- Peter Katuščák and Thomas Kittsteiner. 2020. Strategy-Proofness Made Simpler. (2020). Mimeo.
- Alexander S Kelso Jr and Vincent P Crawford. 1982. Job matching, coalition formation, and gross substitutes. *Econometrica: Journal of the Econometric Society* (1982), 1483–1504.
- Donald E Knuth. 1974. The asymptotic number of geometries. *Journal of Combinatorial Theory, Series A* 16, 3 (1974), 398–400.
- Herman B Leonard. 1983. Elicitation of honest preferences for the assignment of individuals to positions. *Journal of political Economy* 91, 3 (1983), 461–479.
- Jacob D Leshno and Irene Lo. 2021. The cutoff structure of top trading cycles in school choice. *The Review of Economic Studies* 88, 4 (2021), 1582–1623.
- Shengwu Li. 2017. Obviously strategy-proof mechanisms. *American Economic Review* 107, 11 (2017), 3257–87.
- Andrew Mackenzie and Yu Zhou. 2020. Menu mechanisms. *Available at SSRN 3521257* (2020).

- Takehito Masuda, Ryo Mikami, Toyotaka Sakai, Shigehiro Serizawa, and Takuma Wakayama. 2020. The net effect of advice on strategy-proof mechanisms: An experiment for the Vickrey auction. *ISER DP* 1109 (2020).
- Takehito Masuda, Toyotaka Sakai, Shigehiro Serizawa, and Takuma Wakayama. 2019. A strategy-proof mechanism should be announced to be strategy-proof: An experiment for the Vickrey auction. *ISER DP* 1048 (2019).
- Peter McGee and Dan Levin. 2019. How obvious is the dominant strategy in an English Auction? Experimental evidence. *Journal of Economic Behavior & Organization* 159 (2019), 355–365.
- Marek Pycia and Peter Troyan. 2021. A theory of simplicity in games and mechanism design. *CEPR Discussion Paper No. DP14043* (2021). Available at SSRN: <https://ssrn.com/abstract=2853563>
- Alex Rees-Jones. 2018. Suboptimal behavior in strategy-proof mechanisms: Evidence from the residency match. *Games and Economic Behavior* 108, C (2018), 317–330. <https://doi.org/10.1016/j.geb.2017.04.011>
- Alvin E Roth and Andrew Postlewaite. 1977. Weak versus strong domination in a market with indivisible goods. *Journal of Mathematical Economics* 4, 2 (1977), 131–137.
- Raghuvansh R. Saxena, Ariel Schwartzman, and S. Matthew Weinberg. 2018. 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*. 2026–2035. <https://doi.org/10.1137/1.9781611975031.132>
- Lloyd Shapley and Herbert Scarf. 1974. On cores and indivisibility. *Journal of Mathematical Economics* 1, 1 (1974), 23–37.
- Ran I Shorrer and Sándor Sóvágó. 2017. Obvious mistakes in a strategically simple college admissions environment. (2017).

A Additional Preliminaries

A.1 Environments

First, we formally define an *environment*.

Definition A.1. An environment with n agents consists of a set A of outcomes, and sets $\mathcal{T}_1, \dots, \mathcal{T}_n$ of types of the n agents. For ordinal environments, each $t_i \in \mathcal{T}_i$ induces a weak order³⁴ $\succeq_i^{t_i}$ over the outcomes in A . For cardinal environments, each t_i induces a utility function $u_i(t_i, \cdot) : A \rightarrow \mathbb{R}$ (in these environments, we additionally define $a \succeq_i^{t_i} b$ to mean $u_i(t_i, a) \geq u_i(t_i, b)$).³⁵

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 A.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 .³⁶

A.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 A.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 A.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 $i = \pi(1), \pi(2), \dots, \pi(n)$, 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-optimal³⁷ 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

³⁴A 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 $b \succeq a$).

³⁵When no confusion can arise, we also write $\succeq_i^{t_i}$ in place of $\succeq_i^{t_i}$. We also write $a \succ_i^{t_i} b$ when $a \succeq_i^{t_i} b$, but we do not have $b \succeq_i^{t_i} a$.

³⁶When 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 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 .

³⁷A matching μ is Pareto-optimal for the applicants if there is no other matching μ' such that $\mu'(a) \succeq_i^{t_i} \mu(a)$ for all applicants a , and $\mu'(a) \succ_i^{t_i} \mu(a)$ for some applicant a .

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.

Definition A.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 A.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 A.7. *Deferred Acceptance (DA) 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 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”.

A.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 A.8. An auction environment for n players (known as *bidders*) and a set of items M is one in which the outcomes are allocations $(A_1, \dots, A_n)$ over M (i.e. each $A_i \subseteq M$ and $A_i \cap A_j = \emptyset$ for $i \neq j$), along with prices $(p_1, \dots, p_n)$ that the bidders must pay. The set of types $\mathcal{T}_1, \dots, \mathcal{T}_n$ are (arbitrary) sets of valuation function $v_i : 2^M \rightarrow \mathbb{R}_{\geq 0}$, which map subsets of M to a cardinal utility for that set of items. The utility of player i under an outcome with allocation $(A_1, \dots, A_n)$ and prices $(p_1, \dots, p_n)$ 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 $(A_1, \dots, A_n)$.

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 $(A_1, \dots, A_n)$, player i is charged

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

(where the max is taken over all allocations A' 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 $\{0, 1, \dots, K\}$, for some K that may be a function of m and/or n (note though that the “scale” of these utility values is arbitrary, so 1 unit can represent a single cent or even a smaller unit of currency).

Definition A.9. An additive valuation $v_i : 2^M \rightarrow \mathbb{R}_{\geq 0}$ 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 A.10. A unit demand valuation $v_i : 2^M \rightarrow \mathbb{R}_{\geq 0}$ 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 A.11. A gross substitutes valuation $v_i : 2^M \rightarrow \mathbb{R}_{\geq 0}$ 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 $GS = GS^{m,n}$.

A.4 Notation

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

B Missing Proofs

For the reader’s convenience, throughout the appendix, we restate theorems from the main body before giving their proofs.

Theorem 5.4. For any player i , there exist priorities of the institutions such that no preference-linear, menu-revealing description of DA exists. In fact, any menu-revealing extensive-form description for DA which is preference-read-once requires $\Omega(n^2)$ memory.

Proof. Fix an applicant d_* and consider any preference-read-once menu-revealing 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 $d_1, \dots, d_{n/4}$ must be read. More precisely, for each $i \in \{1, \dots, n/4\}$, the list of d_i must be read up to the position of a certain institution h'_i .
2. For each $i = 1, \dots, n/4$, each of the institutions in $\{h_{n/4+1}, \dots, h_{n/2}\}$ may or may not (each, independently) appear on d_i 's list (before h'_i appears). Let $T_i \subseteq \{h_{n/4+1}, \dots, h_{n/2}\}$ denote those institutions which actually appear.
3. Each set T_i (for each $i \in \{1, \dots, n/4\}$) 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 $i \in \{1, \dots, n/4\}$ we must read the entire set T_i during the “menu computation” phase of the menu-revealing description. Property (3) ensure that the entire set T_i must be remembered by the mechanism. Property (2) ensures 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 7). 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 $i = 1, \dots, n/4$) and “bottom” cycles (for $i = n/4 + 1, \dots, n/2$).

The institutions' priorities are fixed. For top cycles with $i \in \{1, \dots, n/4\}$, 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 $i \in \{n/4 + 1, \dots, n/2\}$, we have

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

For the top cycle applicants d_i with $i \in \{1, \dots, n/4\}$, 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 $i \in \{n/4 + 1, \dots, n/2\}$)) are fixed. Specifically, we have

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

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

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

where T_i is an arbitrary subset of $\{h_i | i \in \{n/4 + 1, \dots, n/2\}\}$ (the set of h_i from bottom tier 2-cycles) placed in any order. Any such collection of $\{T_i\}_{i=1, \dots, n/4}$ 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 list³⁸. 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 $i \in \{1, \dots, n/4\}$).

Lemma B.1. *In order to correctly compute the menu of d_* on $\mathcal{S}'$, any preference-read-once mechanism must query the entire preference list of each top cycle applicant d_i (up to the position of h'_i).*

³⁸This collection of preferences can also be constructed with full preference list by adding some unmatched “dummy institution” to represent truncating d_i 's list.

Proof. Consider the rejection chain that occurs when d_* submits a list containing only of 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 will 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 preference-read-once mechanism must read all of d_i ’s list before being able to correctly output the menu of d_* . $\square$

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 B.2. *For each distinct set of preference profiles t_{-*} in $\mathcal{S}$, the induced function $DA(\cdot, t_{-*}) : \mathcal{T}_* \rightarrow A$ 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. $\square$

Thus, at the moment any preference-read-once MTR decision DAG presents the menu to d_* , the decision DAG 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 decision DAG requires at least this many states, and thus uses space $\Omega(n^2)$. $\square$

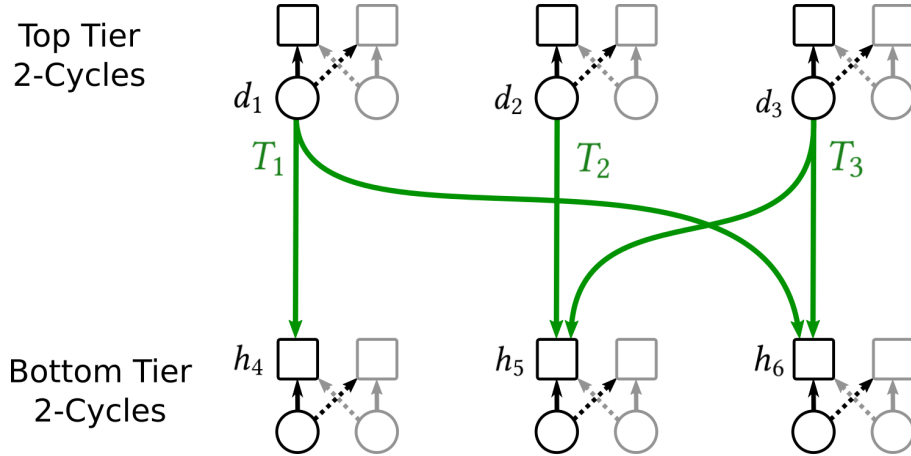


Figure 7: Illustration of the set of preferences witnessing that DA has no preference-linear menu-revealing description, with $n=12$. Circular nodes represent applicants and square nodes represent institutions. Dark nodes represent d_i or h_i for some i , and grey nodes represent d'_i or h'_i . The green arrows directed outwards from a top-tier d_i represent the sets T_i .

Remark B.3. *We make one remark here regarding the comparison between [Theorem 5.4](#) and [Theorem 5.3](#). Unlike TTC, the mechanism DA is bossy (that is, one player can impact the match of another player without changing their own match (for DA, this is in order to maintain stability)). Because TTC is non-bossy, the only information that the menu-revealing description in [Theorem 5.3](#) needs from player i ’s preference list is their choice from their menu.*

One might suspect that the bossiness of “picking from player i ’s menu” in DA is the reason for its increased

complexity (that is, because more information is needed from player i 's menu than just its choice from the menu). This is not the case. In fact, if we consider a variant of the DA environment in which player i (who is presented the menu) is only allowed to rank one institution, then DA will be nonbossy for player i . Still, the above proof of the complexity of a menu-revealing description for player i goes through (because [Lemma B.2](#) requires player i to make only one proposal).

Theorem 6.1. *SEDA describes the applicant-optimal stable matching. In particular, if every applicant is assigned to an institution according to [Figure 5](#), then the result is the applicant-optimal stable matching (the same as in applicant-proposing DA).*

Proof. Fix an applicant d_* and let $\mu_0 = APDA(P)$. Let $\bar{P} = P|_{d_*:\emptyset}$ 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 $IPDA(\bar{P})$, 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 $IPDA(\bar{P})$.

If d_* is unmatched in μ_0 , then by the fact that μ_0 is the applicant-optimal stable outcome ([Corollary E.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 $IPDA(\bar{P})$, and d_* does not get any proposals in $IPDA(\bar{P})$ that she deems acceptable by P . In particular, d_* is not matched in SEDA.

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 $IPDA(\bar{P})$ is exactly h_* . We do this in two steps: first, we show d_* gets a proposal from h_* in $IPDA(\bar{P})$, 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 E.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 E.3](#)), we can construct a run of $IPDA(\bar{P})$ 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 $IPDA(\bar{P})$, 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 B.4. *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 E.1](#)), we know that μ' is stable for P'' . But by the fact that $APDA$ results in the applicant-optimal stable matching ([Corollary E.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 $h \succ_{d_*}^P \mu'(d)$ 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. $\square$

Because d_* accepts no proposals in the run of $IPDA(P'')$, this constitutes a run of $IPDA(\bar{P})$ as well. Thus, d_* is not matched in $IPDA(P'')$, and thus d_* does not receive any proposal better than h_* in $IPDA(\bar{P})$, as desired.³⁹ $\square$

C Global Structure of Matching Mechanisms

Our goal in this section is to formalize and prove the results discussed in [Section 5.3](#). Recall that our motivation question is the following: for the menu-revealing description of TTC, can “stage 3” of [Definition 4.4](#) 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 *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)).

³⁹Mathematically, 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 ([Hatfield and Milgrom, 2005](#)).

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

$$\text{tc}(f) = \log_2 \max_i |\{M_{t_{-i}} \mid t_{-i} \in \mathcal{T}_{-i}\}|.$$

Recall that $M_{t_{-i}} \subseteq A_i$ is a subset of the *payoffs* of player i , 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 C.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 $\text{GM}_{t_{-i}} : \mathcal{T}_i \rightarrow A$ given by $\text{GM}_{t_{-i}}(t_i) = f(t_i, t_{-i})$.

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

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

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 menu-revealing 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 menu-revealing 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.

C.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 C.3. The global taxation complexity of SD is $\tilde{\Theta}(n)$.

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 C.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. $\square$

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 $\tilde{O}(n)$ bits to write down p' , and thus the global taxation complexity of serial dictatorship is $\tilde{O}(n)$. $\square$

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 C.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), yet has high *gtc*. This shows that *gtc* can be a rather “brittle” yardstick for measuring simplicity, necessitating more refined approaches (such as [Section 5](#) and [Appendix D](#)).

C.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 $\tilde{\mathcal{O}}(n^2)$. The proof of [Theorem 5.4](#) 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 B.2](#) proves precisely this). Indeed, one way to summarize the proof of [Theorem 5.4](#) using the language of this section is the following: [Theorem 5.4](#) 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 C.5. *The global taxation complexity of DA is $\Omega(n^2)$.*

C.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 menu-revealing 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 preference-linear menu-revealing 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 C.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 menu-revealing preference-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 C.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 $[n] = \{1, \dots, n\}$. Let $k = n/3$, and for notational convenience, let $r_1, \dots, r_k$ denote all institutions congruent to 1 mod 3, let $x_1^0, \dots, x_k^0$ denote those congruent to 2, and let $x_1^1, \dots, x_k^1$ denote those congruent to 0. Let $a_1, \dots, a_k; y_1, \dots, y_k; z_1, \dots, z_k$ 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

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

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 $b_{i,1}, b_{i,2}, \dots, b_{i,k}$.

Lemma C.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 versus 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 j th block to select their institution. Observe that, as all agents $y_{i'}$ and $z_{i'}$ agree on the ordering (except for adjacent flips), the ℓ th 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. $\square$

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

Through a somewhat delicate construction, we now show that TTC can essentially “simulate” behaviour similar to the above⁴⁰.

Theorem C.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 $p_1^A, \dots, p_k^A$ who only rank “primary” institutions⁴¹ $p_1^O, \dots, p_k^O$, the TTC mechanism can simulate each of the k different “serial dictatorships” $SD(p_i^A, p_{i+1}^A, \dots, p_k^A)$ for $i \in \{1, \dots, k\}$. The choice of this $i \in \{1, \dots, k\}$ 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 $r_0^A, r_1^A, \dots, r_k^A$ in the rotation gadget; and k primary agents $p_1^A, \dots, p_k^A$. There is a “helper institution” h^O in the rotation gadget; $k+1$ institutions $r_0^O, r_1^O, \dots, r_k^O$ in the rotation gadget; and k primary institutions $p_1^O, \dots, p_k^O$. 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 8.

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

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

Proof. We consider the run of TTC after player 1 points to institution r_i^O . First, player 1 and each agent $r_0^A, \dots, r_{i-1}^A$ (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 primary institutions. Each of them have the same

⁴⁰While 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 $\tilde{O}(n)$ 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 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.

⁴¹In this proof, we denote institutions using a super-scripted “ O ”, which stands for “object”.

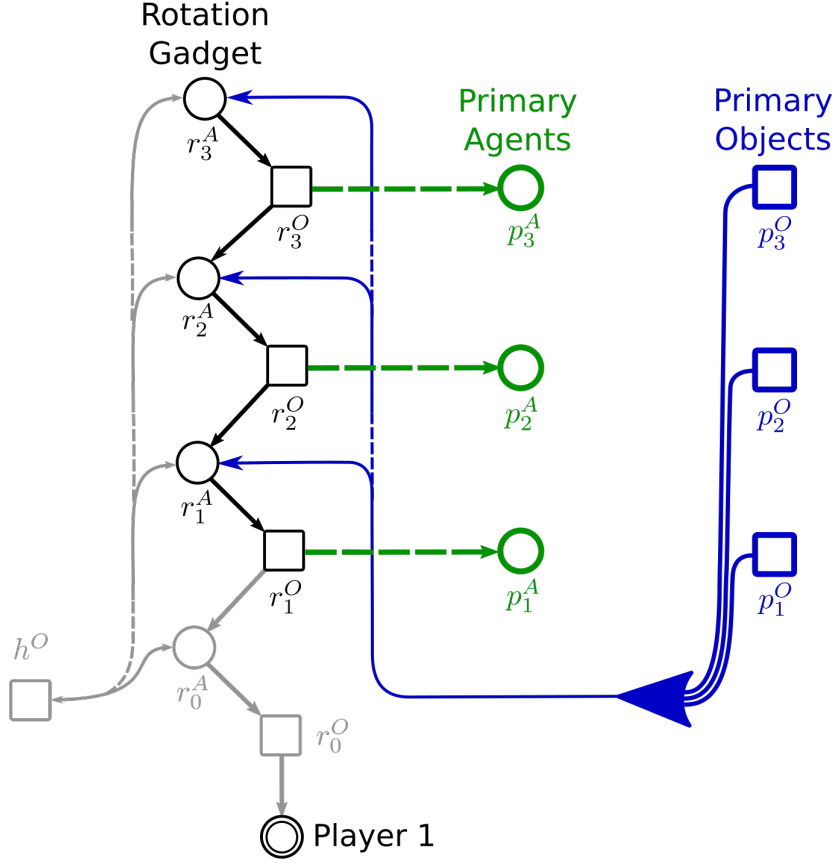


Figure 8: Illustration of the set of preferences witnessing that the global taxation complexity of TTC is $\Omega(n^2)$. Circular nodes represent agents and square nodes represent institutions. Primary agents have any preference order over primary institutions. The solid edges represent the top choice 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.

preference list, and their favorite remaining agent is r_{i+1}^A . But r_{i+1}^A points to r_{i+1}^O , and r_{i+1}^O points to p_{i+1}^A . Thus, p_{i+1}^A will be matched to their top choice p_x^O of primary agents according to the primary agent preferences Q . The four agents and institutions $r_{i+1}^A, r_{i+1}^O, p_{i+1}^A, p_x^O$ are then removed in a cycle. Similarly, p_{i+2}^A 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 $p_{i+1}^A, p_{i+2}^A, \dots, p_k^A$. $\square$

We now show that this implies that the global taxation complexity of TTC is high⁴². For notational convenience, let k be even and let $x_1^0 = p_1^O, x_2^0 = p_3^O, \dots, x_{k/2}^0 = p_{k-1}^O$ denote those primary institutions p_i^O with i odd, and let $x_1^1 = p_2^O, x_2^1 = p_4^O, \dots, x_{k/2}^1 = p_k^O$ denote those p_i^O with i even. Let $y_1, \dots, y_{k/2}$ and $z_1, \dots, z_{k/2}$ denote the corresponding sets of primary agents (the y_j s denoting those p_i^A s with i odd and the z_j s denoting those p_i^A 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

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

⁴²Essentially, 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).

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 $(b_{i,1}, b_{i,2}, \dots, b_{i,i})$.

Lemma C.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 $y_i = p_{2i-1}^A$ and $z_i = p_{2i}^A$ differ on their preference over institutions x_j^0 and x_j^1 .

Suppose player 1’s favorite institution is r_x^A for $x = 2(j-i) - 1$. This will match primary agents to primary institution according to $SD(p_x^A, p_{x+1}^A, \dots, p_k^A)$. 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_ℓ^0 and x_ℓ^1), the top remaining choices of y_i and z_i are x_j^0 and x_j^1 under both $\vec{b}$ and $\vec{b}'$. Under $\vec{b}$, agent y_i is matched to x_j^0 , and under $\vec{b}'$, agent z_i is matched to x_j^0 .

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

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 $\Omega(k^2) = \Omega(n^2)$. $\square$

D Simple Menu-Revealing Auctions

In this section, we investigate the prospect of simple menu-revealing descriptions of *auctions*.

D.1 Simple Descriptions of Ascending Auctions

In contrast to matching markets, the outcome of an auction is often described to the bidders directly, without any hint of a procedure given for *finding* that outcome. In the canonical case of the VCG mechanism which we consider here, this means that bidders are directly told that the allocation will maximize welfare, i.e. $\sum_i v_i(S_i)$ (with prices defined separately according to the VCG pricing rule).

However, there is an extremely natural way to compute the outcome of certain multi-item auctions: an *ascending-price auction* (with one price per item). We study additive (AD), unit demand (UD), and gross substitutes (GS) auctions in this model.

Definition D.1. *An ascending price auction (with one price per item) is an algorithm for computing the auction in which:*

- *Bidders valuations are only queried through demand queries. These take as input a bidder i and prices $p_1, \dots, p_m$ on each item, and return some⁴³ bundle of items S maximizing $v_i(S) - \sum_{j \in S} p_j$.*
- *Throughout the run of the auction, the prices used in the demand queries (to any bidder) are weakly increasing.*

Ascending-price auctions are widely studied, and celebrated positive and negative results exist (Gul and Stacchetti, 1999, 2000; Leonard, 1983; Demange and Gale, 1985; Demange et al., 1986; Kelso Jr and Crawford, 1982). For many of these positive results, there are additional ways in which these descriptions are simple. Similarly to the way matching mechanisms keep track of only a “tentative matching”, many of the most elegant ascending price auctions keep track of only a “tentative allocation of item” (and the prices). If there are m items, it requires $\tilde{O}(m)$ bits of memory to keep track of such an allocation. We formalize this condition as follows:

Definition D.2. *An extensive form description D of an auction mechanism with n players and m items is ascending-linear if:*

- *(Ascending-price) For every execution path $v_1, \dots, v_k$ through D and node v_j in this path, if v_j is labeled by player i , the function $\ell_{v_j} : \mathcal{T}_i \rightarrow S(v_j)$ depends only on the result of a demand query to bidder i ’s type. Moreover, for each item, the price attached to the item in these demand queries are non-decreasing along the path.*

⁴³For our model technical, ties among this value maximization must be broken in some deterministic, consistent way. This detail is immaterial, though.

- (Low memory requirement) The memory requirement of D is at most $\tilde{O}(m)$.

We also need to clarify one aspect of the definition of a *menu-revealing*, which was left ambiguous up until this point. Namely, after the menu is revealed to bidder i in an ascending-price menu-revealing description, we allow the auction to access the type of the bidder i in any way it likes. In the case of matchings, this was a non-issue, because the description could read (and/or store) the entire preference list of agent i . But in auctions, *the price of some items may have already risen* beyond bidder i 's willingness-to-pay, and we do not want to exclude learning about bidder i 's type for only this reason.

D.2 Simple menu-revealing ascending auctions

We start by showing ascending price auctions can compute additive auctions in a simple, menu-revealing way.

Theorem D.3. *For each bidder i , AD has an ascending-linear menu-revealing description for i .*

Proof. Fix a bidder i . Such a description can proceed by simply raising the price of all items (at any constant rate) and checking whether there are still more than two bidders who are willing to pay for the item⁴⁴. Once exactly two bidders are willing to pay for the item, record the second-to-last bidder willing to pay (and their value) and continue raising the price until you learn the value of the last bidder willing to pay. The current price of the item is now the price on the menu.

The final outcome can be computed using only the as follows: i will be awarded each item if their value is above its price on the menu. If i is not awarded the item, it will be given to the highest bidder (at a price equal to the max of the highest previous bid, and player i 's bid). $\square$

While this ascending auction for AD may be somewhat contrived, UD has a celebrated ascending-price description (Crawford and Knoer, 1981) which gives an extremely natural way to compute the allocation. Because this auction only needs to keep track of a “tentative allocation” (and prices), it is also ascending-linear. Unfortunately for the strategic simplicity of UD, we prove that this auction cannot be made *menu-revealing*.

Theorem D.4. *UD has no ascending-linear menu-revealing description. In fact, any ascending-price menu-revealing extensive-form description of UD requires memory at least $\Omega(m^2)$.*

Proof. We phrase this result conveniently in the language of global taxation complexity defined in Appendix C.

Recall that the social choice function UD must output both a maximum weight matching between bidders and items, as well as the VCG prices charged to each bidder. Consider a market with $m+1$ bidders and m items. We construct a large set of profiles of valuations of the m bidders (other than 0) over m items with the following two properties:

- In order to calculate the menu of bidder 0, the price of each item must raise above each bidders value for each item.
- The global menu presented by each distinct valuation profile in the set is distinct (due to the prices UD must charge bidders other than 0).

Let $m=2k$. Divide the bidders and items other than 0 into two groups, “left” bidders $\ell_1^B, \dots, \ell_k^B$ / items $\ell_1^I, \dots, \ell_k^I$ and “right” bidders $r_1^B, \dots, r_k^B$ / items $r_1^I, \dots, r_k^I$. For any bit vector $\vec{b} = \{b_{i,j}\}_{i,j \in \{1, \dots, k\}} \subseteq \{0,1\}^{k^2}$, we define a preference list $P_{\vec{b}}$. The set of all such profiles will satisfy the above two properties.

In each $P_{\vec{b}}$, the preferences of each $\ell_1^B, \dots, \ell_k^B$ are fixed, and each such ℓ_i^B values ℓ_i^I at 10 and ℓ_i^B values all other items at 0. Each r_j^B values r_j^I at 10, and each r_i^I at 0 for each $i \neq j$. Additionally, r_j^B will value the ℓ_i^I at either 0 or 1, depending on the bit vector $\{b_{i,j}\}_{i,j}$. Specifically, r_j^B will value ℓ_i^I at exactly $b_{i,j} \in \{0,1\}$.

First, we claim that in order for an ascending price auction to know the menu presented to player 0, the price of each item must rise above 10. Proof: consider an item x , which may be on the left or on the right. Exactly one bidder values x at 10. Thus, the price for x on bidder 0's menu will be exactly 10 (as one can check via the VCG pricing rule). But, if the bidder who values x at 10 instead valued x at 10.5, then the price for x on bidder

⁴⁴An additive bidder is still willing to pay for the item if and only if they include it in their demand set at the current prices. This bookkeeping be done using an $O(\log n)$ amount of memory per item as follows: Use one bit to track if more than two bidders still demand the item, and use $2\log n$ additional bits to keep track of two arbitrary bidders who still demand the item (in case every remaining bidder for an item drops out at the same price).

0's menu would instead be 10.5. It is easy to see that the only way an ascending price auction can differentiate between these two cases is if the price of item x has risen above 10.

Thus, at the moment when an ascending price auction presents the menu to bidder 0, the price of all items must be above 10. But, for this reason, for each $i, j \in \{1, \dots, k\}$, the auction can never again learn whether bidder r_j^B values ℓ_i^I or not by using a demand query, because the price of ℓ_i^I is too high for r_j^B to ever demand it (under a profile of the form $P_{\vec{b}}$).

Now, we claim that, when preferences are $P_{\vec{b}}$ and when player 1 takes item r_j^I (say, when player 1 values r_j^I at 100 and all other items at 0), bidder ℓ_i^B will pay exactly $b_{i,j}$. Proof: The price bidder ℓ_i^B pays is exactly the utility welfare in the optimal matching without ℓ_i^B (but including bidder 1) minus the utility that other bidders (including bidder 1) actually receive. In the allocation without ℓ_i^B , each bidder (other than 1 and r_i^B) will match to the (unique) item they value at 10. The only remaining available item is ℓ_i^I , which bidder r_j^B values at $b_{i,j}$. The allocation including ℓ_i^B matches all these pairs, with one exception: ℓ_i^B is matched to ℓ_i^I and r_j^B is unmatched. Thus, the presence of bidder ℓ_i^B causes a loss in welfare among the other bidders of exactly $b_{i,j}$.

Thus, each bit vector $\{b_{i,j}\}_{i,j}$ defines a distinct global menu, and the gtc of UD is $\Omega(m^2)$. Moreover, when the auction presents the local menu to bidder 0, it must in fact know the global menu (as it cannot further query for information on $\vec{b}$). Thus, the memory requirement of any menu-revealing ascending-price auction for UD is $\Omega(m^2)$. $\square$

The previous result shows that UD remains very “globally complex” (similar to matching rules such as TTC and DA), despite the fact that the “local” menus in UD are just as simple as those of AD (i.e. for each bidder there is one exogenous-determined price per item). In [Section D.3](#), we discuss a canonical case where even the local structure of the menus is far too complex to facilitate an approach to simplicity.

D.3 Complex auctions

For the more complicated class of GS auctions, ascending auctions (with one price per item) can compute the allocation ([Gul and Stacchetti, 1999](#)), but are completely unable to compute the prices (regardless of space requirements *and* regardless of the need to compute the menu) ([Gul and Stacchetti, 2000](#)).

In terms of menus, gross substitutes auctions are much more complex than AD or UD auctions.

Proposition D.5 ([Dobzinski \(2016\)](#), [Knuth \(1974\)](#)). *It takes $\tilde{\Omega}(2^m)$ many bits to even write down the menu presented in GS auctions.*

The proof is already discussed in [Dobzinski \(2016\)](#) in the context of *communication complexity*. Briefly, the result follows from the fact that for a two-player combinatorial auction, the menu presented to each player is nearly as “complex” as the valuation function of the other player, and the fact that there are doubly exponentially many gross substitutes valuation functions ([Knuth \(1974\)](#)). Because the full proof is a bit hard to find in the literature, we include the proof in [Section E.3](#).

D.4 An Alternative Simplicity Condition for Auctions

In this section, we give another independent sense in which auctions with additive bidders (AD) are simpler than auction with unit demand bidders (UD).

Definition D.6. *An extensive-form description of a mechanism in an auction environment is bidder-read-once if along each execution path, it queries the types of bidders at most once each.*

A bidder-linear description is a bidder-read-once description with memory requirement at most $\tilde{O}(m)$.

Exactly as discussed in [Section 4.2](#) for matching, the bidder-read-once definition will only have bite if additionally complemented with restrictions on the memory used by the algorithm. As for ascending-linear auctions ([Definition D.2](#)), the memory requirement of $\tilde{O}(m)$ is (up to lower order terms) the memory required to write down a “tentative allocation” of the items.

Our first result is that AD has a bidder-linear description, and moreover, this description can present the menu.

Theorem D.7. *For any bidder i , AD has a bidder-linear menu-revealing description for bidder i .*

Proof. The MTR description for agent i simply needs to visit the bidders other than i in any order, and for each item j , keep track of the highest and second highest bid for item j , as well as the identity of the bidders who placed those bids (this uses $\mathcal{O}(\log n + \log K) = \mathcal{O}(\log m)$ bits per item, and thus $\tilde{\mathcal{O}}(m)$ bits of memory overall). The menu presented to bidder i is determined by the highest bid placed on each item (formally, the menu allows the bidder i to decide, for each item separately, whether to take the item for price equal to the highest bid, or to leave the item and pay nothing for it). After receiving bidder i 's type, we allocate to bidder i each item that they value above its price. Moreover, because we store identity of the highest bidder, we can correctly allocate any item that bidder i does not want. Finally, because we store the second highest bid as well, we can charge these other bidders the correct price. $\square$

Again, the situation for AD is similar to that of TTC – a fairly straightforward modification to the typical implementation of AD demonstrates that AD is strategyproof (by presenting a menu to player i). Indeed, this shows how the mechanism (and its strategyproofness) can be understood in terms of the (separate) strategyproofness of m separate single item auctions⁴⁵. All told, we believe the description of AD given by **Theorem D.7** is an easily understood description which cleanly and intuitively emphasizes the strategyproofness of the mechanism.

For UD, under the bidder-read-once simplicity condition things become more complicated *even for the problem of computing the matching*. Indeed, a bidder-linear description cannot even compute the matching⁴⁶ in UD (disregarding the menu, or even the prices charged to any agent).

Theorem D.8. *No bidder-item-simple description of UD exists (even for the outcome of UD alone, without regards to the menu). In fact, any bidder-read-once mechanism for UD requires memory $\Omega(m^2)$.*

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. We show that even this requires memory $\Omega(n^2)$ with a bidder-read-once algorithm.

Let $n = 2k$, and consider any bidder-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 items into two classes, $v_1, \dots, v_k$ and $w_1, \dots, w_k$. For each $i \in [k]$, let b_i denote the bidder the which is queried i th by the algorithm. We give b_i values as follows: b_i wants v_i (but no other v_j for $j \neq i$), and for each j such that $x_{i,j} = 1$, we let b_i demand w_j .

Now, consider a pair of indices $(p, q) \in [k]$. For $i \in \{n/2 + 1, \dots, n\}$, let b_i denote the i th bidder queried. For the $k - 1$ such bidders with $i < n$, let each b_i demand exactly one item in $\{w_1, \dots, w_k\}$, such that only w_q is not demanded by any bidder. For $i = n$, let b_n demand only the item v_p .

Lemma D.9. *There exists a perfect matching (of bidders to items which they demand) if and only if $x_{p,q} = 1$.*

Proof. All of the bidders b_i with $i \in \{n/2 + 1, \dots, n\}$ demand exactly one item, and thus must match there if there is a perfect matching. In particular, v_p must be taken by b_n and each w_j for $j \neq q$ must be taken by $b_{n/2+j}$. Each bidder b_i for $i \neq p$ has the option to match to v_i . If $x_{p,q} = 1$, then b_p can match to w_q to complete the matching. On the other hand, if some perfect matching exists, then every item v_i for $i \neq p$ must be taken by some bidder, but this bidder must of course be b_i . Thus, b_p must go to w_q , and we must have $x_{p,q} = 1$. $\square$

Lemma D.10. *After the first $n/2$ bidders are queried in the above process, the mechanism 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

⁴⁵ As previously mentioned, is in stark contrast to the simplicity theory of OSP matching mechanisms. While an ascending-price auction OSP implements a single item auction (Li, 2017), no OSP mechanism implementing AD exists even for $m = 2$ (Bade and Gonczarowski, 2016).

⁴⁶ This result is unlike most of our results (which hold only for menu-revealing descriptions or otherwise explore the relationship between the menus and the overall mechanism). 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).

for both inputs, a contradiction. $\square$

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. $\square$

E 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).

E.1 Lemmas needed to prove SEDA computes the menu

Lemma E.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 $\mu(h)$. $\square$

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 E.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 $\tilde{d} \neq d$ (either because $\tilde{d}$ proposed to h , or because $\tilde{d}$ was already matched to h and d proposed). We have $\tilde{d} \succ_h d$. We have $\mu(\tilde{d}) \neq h$, simply because μ is a matching. Because this is the first time any applicant has been rejected by a match from μ , $\tilde{d}$ has not yet proposed to $\mu(\tilde{d})$. This means $h \succ_{\tilde{d}} \mu(\tilde{d})$. However, this means μ is not stable.

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

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

Corollary E.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 E.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 $h \succ_d \mu(d)$, because h is the d 's favorite stable partner. If $d \succ_h \mu(h)$, then μ is not stable. Thus, h cannot be stably matched to any applicant they prefer less than d . $\square$

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 E.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 E.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 E.5](#) implies that each agent in H^0 is matched in every stable outcome, so $H^0 \subseteq H^\mu$. But then we have $|D^0| = |H^0|$ as well as $|D^0| \geq |D^\mu| = |H^\mu| \geq |H^0|$, so the same number of agents (on each side) are matched in μ as in APDA. Thus, $D^0 = D^\mu$ and $H^0 = H^\mu$. $\square$

Remark E.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.

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

For a contrast between [Section E.1](#) and [Theorem 6.1](#), we also include the “shortest” proof we know of the strategyproofness of DA.

Lemma E.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 $d \notin T$. $\square$

Theorem E.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. $\square$

E.3 Auctions: High complexity of GS valuations

The following theorem is somewhat hard to find in the literature, so we reproduce it.

Proposition D.5 ([Dobzinski \(2016\)](#), [Knuth \(1974\)](#)). *It takes $\tilde{\Omega}(2^m)$ many bits to even write down the menu presented in GS auctions.*

Proof. We prove that this holds even for the special case of $n = 2$, i.e. in an auction with two players. Observe that in this case, the menu of bidder 1 is given by the following: any bundle S of items is attainable, for a price given by $v_2(M) - v_2(M \setminus S)$, where v_2 is the valuation function (i.e. type) of player 2. First, we claim that for any two distinct $v_2 \neq v'_2$, the menu presented by these two valuation functions is distinct. Proof: If $v_2(M) \neq v'_2(M)$,

then the price of M in the two menus is different (as we assume $v_2(\emptyset) = v'_2(\emptyset) = 0$). On the other hand, if $v_2(M) = v'_2(M)$, then $v_2(S) \neq v'_2(S)$ for some $\emptyset \subsetneq S \subsetneq M$ (as we assume $v_2 \neq v'_2$), and thus the price of S is different in the two menus. (As mentioned previously, this reduction is given in [Dobzinski \(2016\)](#) in the context of communication complexity.)

Now, we show that there are *doubly* exponentially many distinct gross substitutes valuation function (using the fact that any matroid rank function is, in particular, gross substitutes). The proof is adapted from [Knuth \(1974\)](#), and given in [Lemma E.10](#) below. $\square$

Lemma E.10 ([Knuth \(1974\)](#)). *There are at least $2^{\tilde{\Omega}(2^m)}$ distinct matroid rank functions on $[m]$.*

Proof. We show that this claim holds for the special case of “paving matroids” arising from “ d -partitions” (note that different, older terminology is used in [Knuth \(1974\)](#)). These matroids are defined with respect to other set systems which induces a partition of the subsets of size d , according to the following definition.

Definition E.11. A d -partition $\mathcal{D} \subseteq 2^{[m]}$ on $[m]$ is a family of sets of size $\geq d$ with the following property: for any $S \subseteq [m]$ of size exactly d , there exists a unique $D \in \mathcal{D}$ such that $S \subseteq D$.

The paving matroid $M_{\mathcal{D}}$ induced by $\mathcal{D}$ is defined as follows: $I \subseteq [m]$ is independent in $M_{\mathcal{D}}$ if and only if either $|I| \leq d$, or $|I| = d+1$ and I is not contained in any element of $\mathcal{D}$.

Lemma E.12. *For any d -partition $\mathcal{D}$, the set system $M_{\mathcal{D}}$ is a matroid.*

Proof. It is clear that the downwards-closed property holds in $M_{\mathcal{D}}$. To verify that the independent set exchange property holds, the only nontrivial case is if I and J are independent and $|I| = d$ and $|J| = d+1$. Because $\mathcal{D}$ is a d -partition, there is a unique $D \in \mathcal{D}$ where $I \subseteq D$. By the definition of $M_{\mathcal{D}}$, J is not contained in any element of $\mathcal{D}$, so in particular there is some $j \in J \setminus D$. But then $I \cup \{j\}$ cannot be contained in any element of $\mathcal{D}$, and is thus independent, as desired. $\square$

We now introduce yet another family of sets. We will show that, for any family of sets $\mathcal{T}$ which meets the assumptions of the next lemma, there exists a distinct paving matroid which arises from each *sub-family* $\mathcal{T}'$ of $\mathcal{T}$. Later, we will show that we can find a $\mathcal{T}$ which is very large, thus showing that there are many distinct paving matroids. For simplicity, we now assume that n is even.

Lemma E.13. *Suppose $\mathcal{T} \subseteq 2^{[m]}$ is a family of sets of size $n/2$ with the following property: for any $S, T \in \mathcal{T}$ with $S \neq T$, we have $|S \cap T| \leq n/2 - 2$ when $|S \cap T| \leq n/2 - 2$, we say that S and T “differ in at least two elements of $[m]$ ”. Let $\mathcal{D}(\mathcal{T})$ be the set family defined by taking $\mathcal{T}$, and additionally adding all sets of size $n/2 - 1$ which are not contained in some element of $\mathcal{T}$. Then $\mathcal{D}(\mathcal{T})$ is a $(n/2 - 1)$ -partition.*

Proof. Consider any $S \subseteq [m]$ of size $n/2 - 1$. If S is not contained in any member of $\mathcal{T}$, then S itself is a member of $\mathcal{D}(\mathcal{T})$ and certainly this is the unique element of $\mathcal{D}(\mathcal{T})$ containing S . Otherwise, $S \subseteq T \in \mathcal{T}$. If any other distinct $T' \in \mathcal{T}$ were such that $S \subseteq T'$, then we would have $|T \cap T'| \geq |S| = n/2 - 1$, contradicting our assumption on $\mathcal{T}$. Thus, we know T is the unique element of $\mathcal{T}$ with $S \subseteq T$, as needed. $\square$

Lemma E.14. *Suppose (as above) that $\mathcal{T} \subseteq 2^{[m]}$ is a family of sets of size $n/2$ such that any two elements of $\mathcal{T}$ differ in at least two elements of $[m]$. Then, for any two subsets $\mathcal{T}_1, \mathcal{T}_2 \subseteq \mathcal{T}$ with $\mathcal{T}_1 \neq \mathcal{T}_2$, the matroids $M_{\mathcal{D}(\mathcal{T}_1)}$ and $M_{\mathcal{D}(\mathcal{T}_2)}$ are distinct.*

Proof. Suppose without loss of generality that $T \in \mathcal{T}_1 \setminus \mathcal{T}_2$, and recall that T is a set of size $n/2$. Then T is not independent in $M_{\mathcal{T}_1}$, by definition. On the other hand, T certainly cannot be contained in any element of $\mathcal{T}_2$, so T is independent in $M_{\mathcal{T}_2}$, as needed. $\square$

Now, to conclude the proof it suffice to construct a large family of sets $\mathcal{T}$.

Lemma E.15. *There exists a family $\mathcal{T} \subseteq 2^{[m]}$ of at least $\binom{m}{m/2} / (m/2)^2 = \tilde{\Omega}(2^m)$ sets of size $n/2$ such that any two elements of $\mathcal{T}$ differ in at least two elements of $[m]$.*

Proof. Consider the following “greedy algorithm” for constructing $\mathcal{T}$: Start with $\mathcal{T} = \emptyset$, and (as long as it is possible), add any set S of size $n/2$ such that no set T previously in $\mathcal{T}$ is such that $|S \cap T| \geq n/2 - 2$. Observe the following: for every set T currently in $\mathcal{T}$, there are exactly $\binom{m}{m/2}^2$ distinct subsets S of $[m]$ of size $m/2$ such that $|S \cap T| = m/2 - 1$ (each such S removes exactly one element of T , then adds back an element of $[m] \setminus T$). $\square$

□

F Continued DPDA menu computation

In this section, we construct a preference-linear algorithm for computing one applicant's menu in DA. The resulting algorithm is somewhat involved and not very intuitive; correspondingly, we are mostly interested in this algorithm for technical reasons, and do not propose this algorithm as any sort of “simple” approach to DA. The algorithm essentially works as per [Example 5.1](#), but avoiding needing the need to “restart many times” by using careful bookkeeping (to intuitively “simulate all of the separate runs of the brute force description on top of each other”).

Before describing the algorithm, we start by describing the main part of our proof of correctness. This helps to illustrate why the algorithm works the way it does. The proof involves constructing a sort of bijection between one applicant's menu in DA, and the *institution-optimal* stable matching under a related set of preferences. Our preference-linear algorithm can be viewed as an algorithm for institution-optimal stable matching under this related set of preferences. The difficulty arises because we must do this in a “applicant-proposing way” (while additionally only storing a small amount of data, of course). Fortunately, an applicant-proposing procedure for calculating the institution-optimal outcome is already known from [Ashlagi et al. \(2017\)](#)⁴⁷.

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 start by defining some notation that will be used throughout the section. Consider the applicant-optimal matching without d_* , i.e. $\mu_{\text{app}} = \text{APDA}(\mathcal{D} \setminus \{d_*\}, \mathcal{H})$. Recall that DA is independent of the order in which proposals are chosen (a fact that we lean heavily on throughout this section). Let $\mathcal{H}_{\text{term}}$ denote all those institutions unmatched in μ_{app} . These institutions are certainly on d_* 's menu. Let $\mathcal{H}_{\text{picky}}$ denote all those matched institutions h such that $\mu_{\text{app}}(h) \succ_h d_*$. We can already tell that each institution in $\mathcal{H}_{\text{picky}}$ is *not* on d_* 's menu. Let $\mathcal{H}_{\text{open}} = \mathcal{H} \setminus (\mathcal{H}_{\text{picky}} \cup \mathcal{H}_{\text{term}})$ denote all the remaining institutions – these may or may not be on d_* 's menu, because $h \in \mathcal{H}_{\text{open}}$ will accept a proposal from d_* (if the tentative matching is μ_{app}) but a “rejection cycle” may result in d_* getting rejected from h eventually.

We can now define the “related set of preferences” mentioned above. Define the *augmented preference list* P' as follows: For each $h_i \in \mathcal{H}_{\text{open}}$, 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 are as follows: for each $h_i \in \mathcal{H}_{\text{open}}$:

$$\begin{array}{ll} d_i^{\text{try}} : h_i^{\text{try}} \succ h_i \succ h_i^{\text{fail}} & d_i^{\text{fail}} : h_i^{\text{fail}} \succ h_i^{\text{try}} \\ h_i^{\text{try}} : d_i^{\text{fail}} \succ d_i^{\text{try}} & h_i^{\text{fail}} : d_i^{\text{try}} \succ d_i^{\text{fail}} \end{array}$$

We need to modify the preference lists of the pre-existing institutions as well. But this modification is simple: for each $h_i \in \mathcal{H}_{\text{open}}$, replace d_* with d_i^{try} . (For each $h \notin \mathcal{H}_{\text{open}}$, you can remove d_* from their list.) The institution-optimal matching in this augmented set of preferences P' will encode the menu, as we need⁴⁸.

Proposition F.1. *An institution $h_i \in \mathcal{H}_{\text{open}}$ 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 have h_i^{try} matched to d_i^{try} .*

Proof. For both directions of this proof, we use the following lemma from [Immorlica and Mahdian \(2005\)](#); [Cai and Thomas \(2019\)](#): 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}), h_i^{try} goes unmatched (say, in the applicant-optimal matching). Note that

⁴⁷For 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 \(2019\)](#)).

⁴⁸For 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 $h_i \in \mathcal{H}_{\text{open}}$, i.e. “trying to see if h_i is on their menu”. If d_* would be kicked back out from h_i after proposing, so will d_i^{try} (because they serve the same role as d_* at h_i). Thus, d_i^{try} will propose to h_i^{fail} , and end up “completing a rotation” (among h_i^{try} and h_i^{fail} , and possibly other “regular institutions” as well). 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.

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}).

($\Leftarrow$) 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 original doctors 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. $\square$

With this lemma in hand, we can present our algorithm as [Algorithm 1](#). We show that it is a preference-linear applicant-proposing algorithm, and that it can be viewed as a special case of an algorithm from [Ashlagi et al. \(2017\)](#) which computes the institution-optimal stable matching. For this section, we call this the “AKL algorithm”. A closely related algorithm is presented in [Cai and Thomas \(2019\)](#) for computing matches “between” the applicant and institution optimal (via the “rotation poset”).

Proposition F.2. *The set of rejections tracked in [Algorithm 1](#) is a special case of the rejections found in a run of the AKL algorithm on the augmented preferences P' . In particular, each $h_i \in \mathcal{H}_{\text{open}}$ will be added to $\mathcal{H}_{\text{menu}}$ if and only if h_i^{try} has no stable partner better than d_i^{try} in P' .*

Proof. The outermost while loop in [Algorithm 1](#) proceeds by picking a $\hat{h} = h_i \in \mathcal{H}_{\text{open}}$ and having them reject their current partner if they prefer d_* to their current partner. This exactly corresponds to what would happen in the AKL algorithm under preferences P' if institution h_i^{try} rejected d_i^{try} . The remainder of the tracking of the rejection chains are the same as well. First, all rejections relative to μ are tracked in V , and any “internal improvement cycles” is written into μ in the branch starting at [Line 20](#). If d_i^{try} is tentatively accepted at h_i but later kicked back out, then this is detected at [Line 32](#), and h_i is removed from consideration for $\mathcal{H}_{\text{menu}}$. On the other hand, in the block surrounding [Line 36](#), we have found what AKL calls a “terminal phase” – this tells us that h_i^{try} was already at their optimal stable partner when they were matched to d_i^{try} . We avoid ever writing proposals from V into μ – this serves the same function as the operation which AKL calls “rolling back” the matching, and (like AKL) we keep track of a set of institutions where accepting a proposal will certainly imply that we are in a “terminal phase” (we call this set $\mathcal{H}_{\text{term}}$, where AKL calls that set S).

The AKL algorithm can be used to compute the institution-optimal outcome for all institutions $\{h_i^{\text{try}} \mid h_i \in \mathcal{H}_{\text{open}}\}$ (if restricted to only starting rejection chains from $h_i \in \mathcal{H}_{\text{open}}$, as we do (see also [Cai and Thomas \(2019\)](#) for how this works for “intermediate” matchings such as this)). Thus, [Algorithm 1](#) implicitly calculates the institution-optimal stable match of each institution h_i^{try} in P' , and adds h_i to $\mathcal{H}_{\text{menu}}$ if and only if h_i^{try} has no stable partner better than d_i^{try} in P' . $\square$

Proposition F.3. *[Algorithm 1](#) is preference-linear.*

Proof. We must show that 1) applicant’s preferences are only queried “in order” (from favorite to least), and 2) the total memory requirement of the algorithm is $\tilde{O}(n)$.

Applicant’s preferences are accessed only via *APDA* (on [Line 1](#)) or when finding the “next tentative acceptance” on lines [14](#) to [16](#). Each of these simply gets the next “proposal” of the corresponding applicant.

We now consider the space complexity of [Algorithm 1](#). The algorithm stores sets $\mathcal{H}_{\text{menu}}$, $\mathcal{H}_{\text{mys}}$, and $\mathcal{H}_{\text{term}}$, and each of these are subsets of institution (and thus may be represented in space $O(n)$). The matching μ is also stored and continually updated (which takes space $O(n \log n)$). The only additional data structure used is the “rejection chain” V . This is an ordered list of pairs (h, d) , but each institution may occur at most once in V at any given point during the execution of the algorithm (as we will never insert a institution more than once, because if we h is ever the “next institution” tentatively accepting a proposals, but h already appears in V , we will remove the earlier occurrence of h via the if branch starting at [Line 20](#)). Thus, V contains $O(n)$ entries, and all data

Algorithm 1 An applicant-proposing algorithm for finding the menu of a single applicant

Input: A doctor d_* and preferences of applicants $\mathcal{D} \setminus \{d_*\}$ and institutions $\mathcal{H}$

Output: The menu $\mathcal{H}_{\text{menu}}$ presented to d_*

```

1: Let  $\mu_0$  be the doctor-optimal stable matching for doctors in  $\mathcal{D} \setminus \{d_*\}$ , i.e.  $DPDA(\mathcal{D} \setminus \{d_*\}, \mathcal{H})$ 
2: Let  $\mathcal{H}_{\text{term}}$  be all institutions unmatched in  $\mu_0$ 
3: Let  $\mathcal{H}_{\text{open}}$  be all  $h \in \mathcal{H} \setminus \mathcal{H}_{\text{term}}$  such that  $d_* \succ_h \mu_0(h)$ ,
   and let  $\mathcal{H}_{\text{picky}}$  be all  $h \in \mathcal{H} \setminus \mathcal{H}_{\text{term}}$  such that  $\mu_0(h) \succ_h d_*$   $\triangleright$  These partition  $\mathcal{H} \setminus \mathcal{H}_{\text{term}}$ 
4: Let  $\mu$  be a copy of  $\mu_0$  and let  $\mathcal{H}_{\text{menu}} = \emptyset$   $\triangleright$  The algorithm slowly fills  $\mathcal{H}_{\text{menu}} \subseteq \mathcal{H}_{\text{open}}$ 
5: while  $\mathcal{H}_{\text{open}} \neq \emptyset$  do
6:   Pick any  $\hat{h} \in \mathcal{H}_{\text{open}}$ 
7:   Let  $h = \hat{h}$ 
8:   Let  $d = \mu(h)$ 
9:   if  $d \succ_h d_*$  then  $\triangleright \hat{h}$  is not on the menu
10:    Remove  $\hat{h}$  from  $\mathcal{H}_{\text{open}}$   $\triangleright$  Pick a new  $\hat{h}$ 
11:    GOTO Line 5
12:   Set  $V = [(h, d)]$ 
13:   while  $h \notin \mathcal{H}_{\text{term}}$  and  $h \neq \emptyset$  do
14:     Query  $d$ 's preference list to get their next choice  $h$ 
15:     while  $h \neq \emptyset$  and  $\mu(h) \succ_h d$  do
16:       Query  $d$ 's preference list to get their next choice  $h$ 
17:       if  $h \neq \emptyset$  and  $h$  does not already appear in  $V$  then  $\triangleright$  Record this in the rejection chain
18:         Add  $(h, \mu(h))$  to the end of  $V$ 
19:         Update  $d$  to  $\mu(h)$ 
20:       else if  $h \neq \emptyset$  then  $\triangleright h$  appears in  $V$ , and we've found a new "rejection rotation"
21:         Let  $(h_1, d_1), \dots, (h_k, d_k)$  be the suffix of  $V$  starting with the first occurrence of  $h = h_1$ 
22:         Update  $\mu$  such that  $\mu(h_i) = d_{i-1}$  (for each  $i = 1, \dots, k$ , with indices taken mod  $k$ )
23:         Update  $V$  by removing  $(h_1, d_1), \dots, (h_k, d_k)$  from the end of  $V$ 
24:         if  $V = \emptyset$  then Let  $d_0 = d_*$ 
25:         else Let  $(h_0, d_0)$  denote the final entry remaining in  $V$ 
26:         if  $d_0 \neq d_*$  and  $d_k \succ_{h_1} d_0$  then
27:           Set  $d = d_0$ 
28:         else if  $d_0 \succ_{h_1} d_1$  then  $\triangleright d_0$  may or may not be  $d_*$  here
29:           Add  $(h_1, d_k)$  to the end of  $V$ 
30:           Set  $d = d_1$ 
31:         else  $\triangleright$  In this case  $d_0 = d_*$  and  $d_k \succ_{h_1} d_*$  (and  $V = []$ )
32:           Remove  $\hat{h}$  from  $\mathcal{H}_{\text{open}}$   $\triangleright \hat{h}$  is not on the menu
33:           GOTO Line 5  $\triangleright$  Pick a new  $\hat{h}$ 
34:    $\triangleright$  If this line is reached, then  $V \neq []$ , and  $\hat{h}$  is on the menu  $\triangleleft$ 
35:   For every institution  $h'$  such that  $(h', d')$  is in  $V$  for some doctor  $d'$ , add  $h'$  to  $\mathcal{H}_{\text{term}}$ 
36:   Add  $\hat{h}$  to  $\mathcal{H}_{\text{menu}}$ 
37:   Remove  $\hat{h}$  from  $\mathcal{H}_{\text{open}}$ 
38: Return  $\mathcal{H}_{\text{menu}}$ 

```

structures used in **Algorithm 1** are of size at most at most $O(n \log n)$, as needed. $\square$

Corollary F.4. *Algorithm 1 is a preference-linear algorithm for computing the menu.*

G Additional figures

Menu-revealing description:

1. Look through the bidders $j \neq i$ in any order, keeping track of the highest bid p_1 and the second-highest bid p_2 (and the bidder who placed them).
2. Bidder i will win the item if their bid is greater than p_1 . If they win, they'll pay p_1 .
3. Otherwise, the bidder who placed p_1 will win the item (and pay a price which is the max of p_2 and bidder i 's bid).



Traditional description (with “menu advice”):

The highest bidder will win the item and pay the second-highest price.

Note: Observe that you cannot influence the price you would have to pay to win the item, and you will win if and only if your bid is above this price.



Menu description (with “consistency advice”):

Your “price to win” the item will be set to the highest bid placed by any other player.

If your bid is above this price, you will win the item and pay this price.

Note: Observe that in order for any bidder to win the item, their bid must be the highest. Thus, only one player can win the item.

Figure 9: A menu-revealing description of a single-item second price auction for player i .

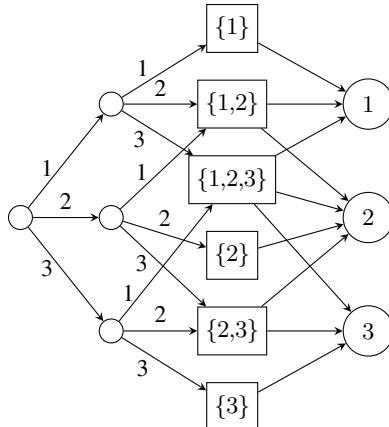


Figure 10: A complete depiction of a menu-revealing descriptive mechanism for median voting with three voters (and possible outcomes 1,2, and 3). The votes of the first two players are queried in the first two layers, and the third layer is labeled with the menu of the third player. The fourth layer gives the outcome.