Strategyproofness-Exposing Descriptions of Matching Mechanisms*

Yannai A. Gonczarowski† Ori Heffetz‡ Clayton Thomas§

November 22, 2024

[Click here for the latest version]

Abstract

A *menu description* exposes strategyproofness by presenting a mechanism to player i in two steps. Step (1) uses others' reports to describe i's *menu* of potential outcomes. Step (2) uses i's report to select i's favorite outcome from her menu. How can canonical matching mechanisms be explained using menu descriptions? We provide simple menu descriptions of Deferred Acceptance (DA) and Top Trading Cycles (TTC). For TTC, our description additionally exposes a simple argument for its strategyproofness. For DA, we prove that this is impossible, and that simple descriptions face a tradeoff between conveying the menu and conveying the matching.

*An earlier version of this paper circulated under the name "Strategyproofness-Exposing Mechanism Descriptions", a one-page abstract of which appeared in the proceedings of EC 2023. For helpful comments and discussions, the authors thank Itai Ashlagi, Tilman Börgers, Eric Budish, Robert Day, Benjamin Enke, Xavier Gabaix, Nicole Immorlica, Guy Ishai, David Laibson, Jacob Leshno, Irene Lo, Kevin Leyton-Brown, Shengwu Li, Paul Milgrom, Michael Ostrovsky, Assaf Romm, Al Roth, Fedor Sandomirskiy, Ilya Segal, Ran Shorrer, Pete Troyan, Matt Weinberg, Leeat Yariv, Huacheng Yu, and participants in HUJI's JESC, Cornell's BERG, Harvard's Economic Theory, Behavioral Economics, and Computer Science Theory Seminars, Stanford GSB's Economics Seminar, WALE 2022, Noam Nisan's 60th Birthday Conference, the INFORMS Workshop on Market Design at EC 2022, MATCH-UP 2022, UMich's Economic Theory Seminar, INFORMS 2022, the 2022 NBER Market Design Working Group Meeting, the UTMD Rising Stars in Market Design workshop, UCI's ACO Seminar, UCSD's Economic Theory/Behavioral/Experimental Seminar, the Bellairs Workshop on Multi-Agent Systems 2023, EC 2023, and the 2023 WZB Matching Market Design conference. Zoë Hitzig provided valuable research assistance. Gonczarowski gratefully acknowledges research support by the National Science Foundation (NSF-BSF grant No. 2343922), Harvard FAS Inequality in America Initiative, and Harvard FAS Dean's Competitive Fund for Promising Scholarship; part of his work was carried out while at Microsoft Research. Heffetz gratefully acknowledges research support by the Israel Science Foundation (grant No. 2968/21), US-Israel Binational Science Foundation (NSF-BSF grant No. 2023676), and Cornell's Johnson School; part of his work was carried out while visiting Princeton's School of Public and International Affairs. Thomas gratefully acknowledges the support of NSF CCF-1955205, a Wallace Memorial Fellowship in Engineering, and a Siebel Scholar award; part of his work was carried out while at Princeton University.

†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.

§Microsoft Research — *E-mail*: thomas.clay95@gmail.com.

1 Introduction

Strategyproof mechanisms are often considered desirable. Under standard economic assumptions, these mechanisms eliminate the need for players to strategize, since straightforward play is a dominant strategy.[1] In practice, however, real participants in strategyproof mechanisms often play theoretically dominated strategies, raising the possibility that they do not perceive the mechanisms as strategyproof.[2]

In this paper, we posit that the way mechanisms are *described* can influence the extent to which participants perceive strategyproofness. In contrast to other recent works, which have sought to implement a given choice rule via different interactive mechanisms in order to discourage non-straightforward play arising from behavioral factors,[3] we propose only changing the ex ante description of the static, direct-revelation mechanism. We propose a general outline—called *menu descriptions*—for presenting a mechanism to one player at a time in a way that exposes strategyproofness; namely, making it hold via an elementary, one-sentence argument.

Our focus is on matching, particularly two canonical mechanisms: the stable Deferred Acceptance (henceforth DA) and the Pareto efficient Top Trading Cycles (henceforth TTC). While these mechanisms are quite successful, we posit that their traditional *outcome descriptions*—detailed and explicit algorithms for calculating the matching outcome—may not expose strategyproofness, since showing this property conventionally requires delicate and technical mathematical proofs.

Our paper includes four main results. The first two present novel menu descriptions of DA and TTC, which expose their strategyproofness while maintaining complexity comparable to that of the traditional descriptions. For TTC, our description is additionally sufficiently similar to the traditional description that it yields a simple proof that the traditional description is strategyproof. Our third and fourth main results are negative findings for DA. They formally show that, in contrast to TTC, *no* menu description of DA yields a simple proof for its traditional description's strat-

[1] We use the term "straightforward" to describe the strategy an agent would play under classic economic assumptions. While often referred to as the "truthtelling" strategy, we avoid this morally laden term, since deviations from this strategy should not be thought of as dishonesty.

[2] Evidence of such non-straightforward behavior comes both from the lab and the field (where participants are typically explicitly informed of the mechanism's strategyproofness). For surveys of the literature, see Hakimov and Kübler (2021); Rees-Jones and Shorrer (2023).

[3] These factors include contingent-reasoning failures (Li, 2017; Pycia and Troyan, 2023) and expectations-based loss aversion (Dreyfuss et al., 2022; Meisner and von Wangenheim, 2023).

egyproofness, and that simple descriptions of DA face a trade-off between conveying the outcome matching and conveying strategyproofness. This illustrate the (perhaps surprising) strategic complexity of DA, and further motivates our menu description.

Before proceeding, we consider a foundational, illustrative example: the Serial Dictatorship (henceforth, SD) mechanism. When matching applicants to institutions,[4] the traditional description of SD is as follows: in some order, say $i = 1, \ldots, n$, applicant i is matched to her highest-ranked not-yet-matched institution. Strategyproofness is quite evident from this description: Applicant i cannot influence the set of not-yet-matched institutions, and straightforward reporting guarantees i her favorite not-yet-matched institution. Our paper asks: To what extent can we find descriptions of other mechanisms that make strategyproofness as evident as in SD?

Our paper begins in Section 2 with preliminaries. We study descriptions in terms of the classic notion of a *menu* (Hammond, 1979)—the set of all institutions an applicant might match to, given others' reports. In particular, a *menu description* for applicant i has the following two-step outline:

Step (1) uses only the reports of other applicants to describe i's menu.

Step (2) matches applicant i to her highest-ranked institution from her menu.

In contrast with some traditional descriptions, strategyproofness can be readily seen from any menu description: Applicant i cannot effect her menu in Step (1), and straightforward reporting guarantees i her favorite possible institution in Step (2).

In Section 3, we present our first main result: A novel menu description of DA, summarized in Table 1. It describes applicant i's menu as all institutions that prefer i to their outcome in "flipped-side-proposing" deferred acceptance, excluding i.

In many ways, our menu description in Table 1 is comparable in complexity to the traditional description of DA, an important consideration for use with real-world participants.[5] Prior to our work, it was not clear how to construct DA's menu, except via a trivial solution involving running the traditional description many times to separately check if different institutions are on i's menu—an approach to describing mechanisms that might be viewed as complicated, implausible, or confusing.

[4] In all mechanisms we consider, the only strategic players are the applicants. The institutions are not strategic, and thus their preferences over the applicants are by convention called *priorities*. We consider one-to-one matching for concreteness, but our results generalize substantially; see Section 3.

[5] We also note that Table 1 gives a useful theoretical tool for reasoning about DA. For example, it immediately implies that when one applicant's priority improves at a set of institutions, her match cannot become worse (Balinski and Sönmez, 1999).

Table 1: Two equivalent descriptions of DA (the applicant-optimal stable match)

Traditional Descr.:	Menu Description:
The applicants and institutions will be matched using the *applicant-proposing deferred acceptance* algorithm.	We will run *institution-proposing deferred acceptance* with all applicants *except you*, to obtain a hypothetical matching. Your menu consists of every institution that ranks you higher than its hypothetically matched applicant. You will be matched to the institution that you *ranked highest* out of your menu.

Note: In the menu description, other applicants' hypothetical match is not their match in DA.

In Section 4, we present our second main result: A novel menu description of TTC, which is in fact contained within a slight tweak of the traditional description. In particular, it features an extra third step after the menu description for applicant i:

Step (1) uses only reports of applicants other than i to calculate i's menu.

Step (2) uses i's report to match i to her top-ranked institution on her menu.

Step (3) uses all reports to calculate the rest of the matching (for all applicants).

We call a description with this three-step outline—i.e., an outcome description that contains a menu description—an *individualized dictatorship*.

Our new description of TTC gives a simple, intuitive proof that TTC is strategyproof. The traditional description of TTC works in terms of "eliminating trading cycles," and it is well known that these cycles can be eliminated in any order. Our new description differs from the traditional one only by eliminating the cycle involving applicant i as late as possible. Combining the above, the match of applicant i equals her match in a menu description, making TTC's strategyproofness clear.[6]

We find our menu description of DA (Section 3) nearly as appealing as our individualized dictatorship for TTC (Section 4). However, one may wonder: Like TTC, does DA have an individualized dictatorship that is a slight tweak of, and results in an intuitive proof of the strategyproofness of, its traditional description?

[6] Following the appearance of our paper, the survey article Morrill and Roth (2024) used our individualized dictatorship to give a self-contained proof that TTC is strategyproofness. Regarding the potential real-world adoption of TTC for public school choice, Morrill and Roth write:

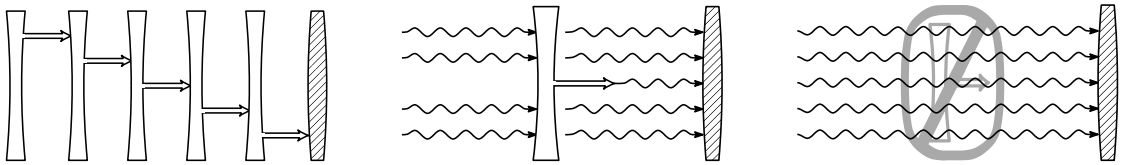
> "Our experience [...] taught us that when we worked with school districts, we should help design not just a mechanism, but also the communication package that explained that mechanism [...]. Perhaps if we had already known of the proof of [Gonczarowski, Heffetz, and Thomas] we could have explained [TTC's strategyproofness] more clearly."

To answer the above question, we examine formal simplicity desiderata capturing a notion of "slight tweaks" of traditional descriptions of many matching mechanisms. First, such descriptions are "applicant-proposing"—i.e., they look at applicants' preferences in favorite-to-least-favorite order. Second, they construct the outcome by keeping track of (and iteratively modifying) a single tentative matching, without performing more complicated bookkeeping—we capture (a flexible, necessary condition for) this property by considering descriptions that keep track of an amount of memory roughly linear in the number of applicants. We adopt the position that any slight tweak of the traditional description of DA should, like our individualized dictatorship for TTC, maintain (at least) these two properties.

In Section 5, we present our third main result, which is our main impossibility theorem: In contrast to (SD and) TTC, no individualized dictatorship for DA is a slight tweak of its traditional description. In fact, we prove that any applicant-proposing individualized dictatorships for DA must use *quadratic* memory—roughly equivalent to keeping track of a separate matching for each applicant.

Our second and third main results complete a striking trichotomy among SD, TTC, and DA. In SD, strategyproofness is exposed "automatically." For TTC, this is not the case, but a slight tweak of the traditional description suffices to expose strategyproofness for one applicant. For DA, a comparable result is impossible, in the precise and strong sense discussed above. See Figure 1 for an illustration.

Figure 1: Trichotomy for (tweaks of) traditional descriptions of SD, TTC, and DA



(a) SD (b) TTC (c) DA

Notes: Each figure depicts an applicant-proposing and linear-memory description. Each description progresses from left to right and calculates the outcome matching, which is depicted as a shaded rectangle. Light rectangles and doubled arrows depict Step (2) of a menu description. Wavy arrows depict other computations. Panel (c) depicts our main impossibility theorem, which proves that DA cannot be described in finer detail as in the other two panels.

Our main impossibility result (Section 5) rules out simple (linear-memory) descriptions of DA of a specific type (applicant-proposing individualized dictatorships). However, one may wonder: Are there simple descriptions of DA—beyond the traditional one and our menu description—that are not ruled out by this impossibility?

To answer the above question, we examine descriptions of DA of all classes implied by our framework. We consider applicant-proposing descriptions (like the traditional one), and institution-proposing descriptions (like our new menu description of DA). Furthermore, we consider three description outlines: Menu descriptions, outcome descriptions, and individualized dictatorships.

In Section 6, we present our fourth main result, which extends our main impossibility theorem with an additional negative finding: Among all the classes of descriptions of DA discussed above, only the traditional description and our novel menu description have descriptions which are sufficiently simple for real-world use. Using a more-refined simplicity desideratum called *locality*, which captures another feature of many traditional descriptions, we prove that any additional description of DA in this classification must, in a precise sense, be a delicate and technical algorithm.[7]

Our fourth main result shows that simple descriptions of DA face a trade-off between conveying strategyproofness and conveying the full matching outcome. DA's traditional (outcome) description, used in countless real-life matching markets, is at one corner of this tradeoff. Only our new menu description is at the other corner, exposing perhaps the most relevant property for applicants: Strategyproofness.

Table 2 summarizes our four results, which construct and classify strategyproofness-exposing descriptions of DA and TTC. In Section 7, we review related work, including our own broader empirical research agenda investigating menu descriptions. We conclude in Section 8, where we also discuss potential practical concerns.

2 Preliminaries

2.1 Mechanisms

This paper studies (static, direct-revelation) strategyproof matching mechanisms. This environment consists of n applicants $\{1, \ldots, n\}$ who are matched to institutions.[8] Applicant i has a strict ordinal preference $\succ_i$ over institutions, also called her

[7] In fact, in Section 6, we construct nontrivial (indeed, linear-memory) descriptions of DA of every class not ruled out by our main impossibility theorem. Unfortunately, each such additional description is exceedingly convoluted and impractical. Our additional impossibilities show that such algorithms must be non-local. Roughly speaking, this means that such descriptions must update bookkeeping concerning some agents when making queries that seem unrelated to them.

[8] We denote applicants i whenever the mechanism (or a concept such as strategyproofness, or a menu) is described specifically to applicant i; we otherwise denote applicants by d (mnemonic:

Table 2: Summary of main results

Section	Description Type(s)	Result and Interpretation
Sec. 3	Institution-proposing menu description for **DA**	**Positive Result**: Description 1 is an alternative description of an applicant's match in DA that exposes strategyproofness
Sec. 4	Applicant-proposing individualized dictatorship for **TTC**	**Positive Result**: Description 2 gives the above (↑) interpretation, and in addition, exposes strategyproofness of traditional description of TTC
Sec. 5	Applicant-proposing individualized dictatorship for **DA**	**Impossibility Result:** There are no such linear-memory descriptions. Hence, menu descriptions cannot convey the strategyproofness of the traditional description of DA
Sec. 6	One-side-proposing, menu, outcome, or individualized-dictatorship for **DA**	**Impossibility Result:** Except the traditional description and Description 1, there are no such local descriptions. Hence, can convey either strategyproofness (menu description) or full matching (outcome description), but not both (individualized dictatorship)

report or her *type*.[9] We let $\mathcal{T}_i$ denote the set of possible types of applicant i, and let A denote the set of matchings.[10] Following much of the matching literature, we treat (only) the applicants as strategic agents who submit preferences.

Definition 2.1. A *(matching) mechanism* is any mapping $f : \mathcal{T}_1 \times \ldots \times \mathcal{T}_n \to A$ from the types of all applicants to a matching.[11] A mechanism is *strategyproof* if, for every $\succ_i, \succ_i' \in \mathcal{T}_i$ and $\succ_{-i} \in \mathcal{T}_{-i}$, we have $f_i(\succ_i, \succ_{-i}) \succeq_i f_i(\succ_i', \succ_{-i})$.

We study the canonical strategyproof mechanisms SD, TTC, and DA. These mechanisms are defined with respect to *priorities* of the institutions over the applicants. We treat the priorities as a non-strategic part of the mechanism (although, our descriptions of mechanisms also treat the priorities as input; see below). SD uses a

doctor). We denote institutions by h (mnemonic: hospital), and matchings by μ.

[9] Applicants may go unmatched, and their preference lists may be partial (indicating that they prefers to remain unmatched over institutions not on her preference list). We also let $h_1 \succ_d h_2$ indicate that applicant d prefers h_1 to h_2; $h_1 \succeq_d h_2$ indicate $h_1 \succ_d h_2$ or $h_1 = h_2$; $\emptyset \succ_d h$ indicate that d does not rank h; $\mu(d)$ denote the match of d in μ; and $\mu(d) = \emptyset$ denote d going unmatched.

[10] As is standard, $\mathcal{T}_{-i}$ denotes the set $\mathcal{T}_1 \times \ldots \times \mathcal{T}_{i-1} \times \mathcal{T}_{i+1} \ldots \mathcal{T}_n$, and for $\succ_i \in \mathcal{T}_i$ and $\succ_{-i} \in \mathcal{T}_{-i}$, we write $(\succ_i, \succ_{-i})$ for the naturally corresponding element of $\mathcal{T}_1 \times \ldots \times \mathcal{T}_n$.

[11] For an applicant i, we let $f_i(\succ_1, \ldots, \succ_n)$ denote i's match $\mu(i)$ in $\mu = f(\succ_1, \ldots, \succ_n)$.

priority order $\succ$ listing all applicants; TTC and SD use a profile of priority orders $\{\succ_h\}_h$, one for each institution h.

Definition 2.2 (SD)**.** For a given priority order $\succ$, Serial Dictatorship (SD) is defined as follows. Applicants are considered in order of highest-to-lowest priority, and each applicant is permanently matched to her favorite not-yet-matched institution.

Definition 2.3 (TTC)**.** For a given profile of institutions' priorities $\{\succ_h\}_h$, Top Trading Cycles (TTC) is defined as follows. Repeat the following until everyone is matched (or has exhausted their preference lists): Every remaining (i.e., currently unmatched) applicant "points" to her favorite remaining institution, and every remaining institution points to its highest priority remaining applicant. There must be some cycle in this directed graph (since there is only a finite number of vertices). Choose any such cycle and "eliminate" that cycle by permanently matching each applicant in the cycle to the institution she is pointing to (and remove all matched agents from consideration for later cycles).

Definition 2.4 (DA)**.** For a given profile of institutions' priorities $\{\succ_h\}_h$, Deferred Acceptance (DA) is defined as follows. Repeat the following until every applicant is matched (or has exhausted her preference list): A currently unmatched applicant is chosen to "propose" to her favorite institution which has not yet "rejected" her. The institution then rejects every proposal except for the *top priority applicant* who has proposed to it thus far. Rejected applicants become (currently) unmatched, while the top priority applicant is tentatively matched to the institution. At the end, the tentative allocation becomes final.

As is well-known, TTC and DA are the two canonical matching mechanisms which are priority-based and strategyproof. TTC is Pareto-efficient for the applicants, and DA is stable. Note that DA refers to the (direct-revelation) mechanism defined by applicant-proposing DA (i.e., outputting the applicant-optimal stable matching); when confusion might arise, we use APDA (and for institution-proposing, we use IPDA). Note also that the outcomes of TTC and of DA are independent of the order in which steps of the above algorithms are chosen (Corollary D.4 and Lemma D.9).

2.2 Descriptions

This paper studies ex ante *descriptions* of matching mechanisms, i.e., descriptions that are given before any concrete inputs are known. When matching markets are

described in detail to participants, this is typically done by specifying a set of explicit, precise, step-by-step instructions for calculating the result, i.e., by specifying an algorithm.[12] Thus, we formally define a description to be any algorithm that uses as input the preferences of the applicants and the priorities of the institutions, and calculates some result (e.g., an outcome matching).[13]

For SD, TTC, and DA, we refer to the description in the above Definitions 2.2 through 2.4 as the *traditional description* of the corresponding mechanism. Formally, for mechanism f, these are algorithm which, using input $\succ_1, \ldots, \succ_n$ (and the priorities of institutions), output $f(\succ_1, \ldots, \succ_n)$. We refer more generally to such algorithms as *outcome descriptions* for f.

Beyond outcome descriptions, we study two other description outlines: menu descriptions and individualized dictatorship descriptions; these are defined in Sections 2.3 and 2.4, respectively.

2.3 Menus and Menu Descriptions

The starting point of our framework for changing mechanism descriptions is the following characterization of strategyproofness in terms of applicants' *menus*.[14]

Definition 2.5 (Menu)**.** For any matching mechanism f, the *menu* $\mathcal{M}_{t_{-i}}$ of applicant i with respect to types $\succ_{-i} \in \mathcal{T}_{-i}$ of other applicants is the set of all institutions h for which there exists some $\succ_i \in \mathcal{T}_i$ such that $f_i(\succ_i, \succ_{-i}) = h$. That is,

$$\mathcal{M}_{\succ_{-i}} = \{\ f_i(\succ_i, \succ_{-i}) \mid \succ_i \in \mathcal{T}_i\ \}.$$

[12] Of course, the way a description/algorithm is actually conveyed to participants can vary. One common real-world approach to relaying matching algorithms is an illustrative video using an example (see, e.g., Figure 3 in Section 5.1 for such a video for DA). In our paper, we abstract over exactly how the algorithm is relayed.

[13] Algorithms can be defined in full detail in various ways. For the reader unfamiliar with mathematical models of algorithms, in Appendix A we give a definition from first-principles that suffices to capture all results in this paper.

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

Theorem 2.6 (Hammond, 1979)**.** *A matching mechanism f is strategyproof if and only if each applicant i always receives her favorite institution from her menu. That is, for every $\succ_{-i} \in \mathcal{T}_{-i}$ and $\succ_i \in \mathcal{T}_i$, it holds that $f_i(\succ_i, \succ_{-i}) \succeq_i h$ for all $h \in \mathcal{M}_{\succ_{-i}}$.*

Proof. Suppose f is strategyproof and fix $\succ_{-i} \in \mathcal{T}_{-i}$. For every $\succ_i \in \mathcal{T}_i$, it holds by definition that for any $h = f_i(\succ'_i, \succ_{-i}) \in \mathcal{M}_{\succ_{-i}}$, we have $f_i(\succ'_i, \succ_{-i}) \succeq_i h$. On the other hand, if applicant i always receives her favorite institution from her menu, then she always prefers reporting $\succ_i$ at least as much as any $\succ'_i$, so f is strategyproof. □

We use menus to describe mechanisms while exposing their strategyproofness:

Definition 2.7 (Menu Description)**.** A *menu description* of mechanism f for applicant i is a description with the following outline:

Step (1) uses only $\succ_{-i} \in \mathcal{T}_{-i}$ to calculate the menu $\mathcal{M}_{\succ_{-i}}$ of applicant i.

Step (2) uses $\succ_i \in \mathcal{T}_i$ to match applicant i to her favorite institution in $\mathcal{M}_{\succ_{-i}}$.

Formally, a menu description for i is thus an algorithm that initially receives only $\succ_{-i}$ as input and calculates $\mathcal{M}_{\succ_{-i}}$ as an intermediate result, then additionally receives $\succ_i$ as input and uses it to calculate i's favorite choice from $\mathcal{M}_{\succ_{-i}}$ as the final result.

The central premise of our paper is that menu descriptions are one way to expose strategyproofness. This is because the strategyproofness of any menu description can immediately be seen via a simple, one-sentence argument: First, applicant i's report cannot affect her menu, and second, straightforward reporting ("truthtelling") gets applicant i her favorite institution from the menu.[15]

2.4 Individualized Dictatorships

Beyond menu descriptions, outcome descriptions that contain menu descriptions also play a key role in our results. We term such descriptions *individualized dictatorships*.

[15] There is also a precise sense in which menu descriptions are the *only* ones for which the above argument for strategyproofness goes through. In particular, suppose a description calculates the match of applicant i in some mechanism f, and has the following outline:

Step (1) uses $\succ_{-i}$ to calculate a set S of institutions.

Step (2) uses $\succ_i$ to match i to her top-ranked institution in S.

Then, it is not hard to show that the set S must be i's menu.

Definition 2.8 (Individualized Dictatorship)**.** An *individualized dictatorship (description)* of mechanism f for applicant i is a description with the following outline:

Step (1) uses only $\succ_{-i} \in \mathcal{T}_{-i}$ to calculate the menu $\mathcal{M}_{\succ_{-i}}$ of applicant i.

Step (2) uses $\succ_i \in \mathcal{T}_i$ to match applicant i to her favorite institution from $\mathcal{M}_{\succ_{-i}}$.

Step (3) uses both $\succ_i$ and $\succ_{-i}$ to calculate the full outcome matching $f(\succ_i, \succ_{-i})$.

Formally, an individualized dictatorship for i is thus an algorithm that initially receives $\succ_{-i}$ as input and calculates $\mathcal{M}_{\succ_{-i}}$, then additionally receives $\succ_i$ as input and calculates i's favorite choice from $\mathcal{M}_{\succ_{-i}}$, and finally proceeds to calculate the entire outcome matching $f(\succ_i, \succ_{-i})$ as the final result.

For example, consider SD (Definition 2.2). This mechanism is easily seen to be strategyproof, directly from its traditional description (and even for many students encountering it for the first time). This is reflected by the fact that applicants are matched in SD via menu descriptions. In particular, when applicants are prioritized $1 \succ 2 \succ \ldots \succ n$, the traditional description of SD can be divided into three steps:

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

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

(3) Each applicant $j > i$ is matched, in order, to her top-ranked remaining institution.

Steps (1) and (2) form menu description, but this menu description is contained within the traditional outcome description, and thus Steps (1) through (3) form an individualized dictatorship.

2.5 Uses of Menu Descriptions

Throughout this paper, we argue that menu descriptions are useful both as an alternative way to describe a mechanism to participants, and as a way to gain insights into the traditional description of the mechanism. Importantly, not all menu descriptions achieve these goals. For example:[16]

[16]This menu description was also identified by Katuščák and Kittsteiner (2020).

Example 2.9 (A "brute force" menu description)**.** Consider any strategyproof matching mechanism f with a traditional description D. For each institution h, let $\{h\}$ denote the preference list that ranks only h as acceptable. Then, consider the following description for applicant i:

(1) Start with $M = \emptyset$. For each institution h separately, evaluate D on $(\{h\}, \succ_{-i})$; if i matches to h, then add h to M.

(2) Match i to her highest-ranked institution in M.

By strategyproofness, h will be included in M in Step (1) if and only if h is on the menu. Thus, the above provides a menu description of f.

While all strategyproof mechanisms have menu descriptions as in Example 2.9, such descriptions do not seem useful. First, we suspect that real participants would find the above description complicated and confusing, precluding its real-world use as an alternative description. Second, without already knowing that the traditional description is strategyproof, there is no clear relation between the outcomes of the above description and those of the traditional description, precluding Example 2.9 as a tool to aid in conveying the traditional description's strategyproofness.

Given the above, we look for simple or appealing new menu descriptions (e.g., our description of DA in Section 3). We also look for menu descriptions that help to illustrate the strategyproofness of traditional description (e.g., our description of TTC in Section 4, which is additionally an individualized dictatorship).

3 A Menu Description of DA

In this section, we present our first main result: A novel menu description of DA. This is Description 1 (rephrased from Table 1 in the introduction).

Description 1 A menu description of (*applicant*-proposing) DA for applicant i

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

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

Description 1 seems comparable in complexity to the traditional description of DA (from Definition 2.4). In fact, it only adds an easy-to-state "menu calculation and matching" step on top of a modified traditional description of DA. We speculate that many real market participants would find such a description understandable. (In fact, our companion paper Gonczarowski et al. (2024) gives empirical evidence that many lab participants can learn this description—see Section 7.)

Crucially, Description 1 uses the *institution*-proposing DA algorithm to describe DA (traditionally described via *applicant*-proposing DA); to give intuition for why the proposing side is flipped, we show via an example that using applicant-proposing DA in this description would not suffice.

Example 3.1. Consider a market with three applicants i, d_1, d_2 and two institutions h_1, h_2. Applicants have preferences $d_1 : h_1 \succ h_2$ and $d_2 : h_2 \succ h_1$, and institutions have priorities $h_1 : d_2 \succ i \succ d_1$ and $h_2 : d_1 \succ i \succ d_2$. Running applicant-proposing DA on these preferences without i gives matching $\{(d_1, h_1), (d_2, h_2)\}$, and both h_1 and h_2 prefer i to their match. However, neither h_1 nor h_2 are on i's menu, since having i propose to any $h_i \in \{h_1, h_2\}$ (after running applicant-proposing DA without i) causes a "rejection cycle" that results in h_i rejecting i. Intuitively, institution-proposing DA fixes this issue by outputting a matching that has no potential "applicant-proposing rejection cycles."[17]

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

Theorem 3.2. *Description 1 is a menu description of DA.*

Proof. Fix institutions' priorities, an applicant i, and preferences $\succ_{-i}$ of applicants other than i. Let $\{h\}$ denote the preference list of i that reports only institution h as acceptable, and let $\emptyset$ denote the preference list of i that reports *no* institution as acceptable. For clarity, denote applicant-proposing DA by $APDA(\cdot) = DA(\cdot)$; denote institution-proposing DA by $IPDA(\cdot)$.

Now, for any institution h, we observe the following chain of equivalences:

h is in the menu of i in $APDA$ (with respect to $\succ_{-i}$)

$\iff$ (By strategyproofness of $APDA$; Theorem D.8)

[17] This intuition regarding "applicant-proposing rejection cycles" is related to the concept of an institution-improving rotation as in Gusfield and Irving (1989).

i is matched to h by $APDA(\{h\}, \succ_{-i})$

$\iff$ (By the Lone Wolf / Rural Hospitals Theorem; Theorem D.6)

i is matched to h by $IPDA(\{h\}, \succ_{-i})$

$\iff$ ($IPDA(\{h\}, \succ_{-i})$ and $IPDA(\emptyset, \succ_{-i})$ coincide until h proposes to i)

h proposes to i in $IPDA(\emptyset, \succ_{-i})$

$\iff$ ($IPDA(\emptyset, \succ_{-i})$ and $IPDA(\succ_{-i})$ produce the same matching;

in $IPDA$, h proposes in favorite-to-least-favorite order)

h prefers i to its match in $IPDA(\succ_{-i})$ (in the market without i). □

In addition to giving a perhaps-appealing alternative description of DA, Theorem 3.2 provides a characterization of the menu in DA which is useful for reasoning about DA's properties. We briefly highlight two applications. First, one can immediately see from Description 1 that, if one applicant's priorities increase at some set of institutions, then (all other things being equal) the match of that applicant in DA can only improve (Balinski and Sönmez, 1999). Second, a short argument using Description 1, which we provide in Remark B.3, shows that in a market with n+1 applicants, n institutions, and uniformly random full length preference lists, applicants receive in DA roughly their $n/\log(n)$th choice in expectation—rather lower than in the case with n applicants, where they receive their $\log(n)$th choice—re-proving results from Ashlagi et al. (2017); Cai and Thomas (2022).

Description 1 generalizes to a broader class of stable matching markets. In fact, in Remark B.2, we observe that the same arguments as in the above proof show that a natural generalization of Description 1 characterizes the menu of DA in many-to-one markets, and even in a general class of markets with contracts, namely, those considered by Hatfield and Milgrom (2005).

Finally, we remark that Description 1 can facilitate a proof from first-principles of the strategyproofness of (traditionally described) DA (without relying on this fact, as in the proof above). We give such a proof in Appendix B. While we view this proof as theoretically appealing, and perhaps useful for classroom instruction, we believe this approach remains far too mathematically involved to convey the strategyproofness of DA's traditional description to real-world participants. In contrast, if a clearinghouse directly adopts Description 1 as a way to describe participants' matches in explicit detail, then strategyproofness follows via a simple, clear proof.

4 An Individualized Dictatorship for TTC

In this section, we present our second main result: A novel individualized dictatorship for TTC that additionally yields a simple, intuitive proof that the traditional description of TTC is strategyproof. This is Description 2.

Description 2 An individualized dictatorship for TTC for applicant i

(1) Using $\succ_{-i}$, iteratively eliminate as many cycles not involving applicant i as possible. Let M denote the set of remaining institutions.

(2) Using $\succ_i$, match i to her highest-ranked institution in M. Call this institution h.

(3) Using $(\succ_i, \succ_{-i})$, eliminate the cycle created when i points to h, then continue to eliminate cycles until all applicants match (or exhaust their preference lists).

Steps (1) and (2) of Description 2 give a menu description of TTC that seems very similar in complexity to the traditional description (from Definition 2.3). Moreover, Description 2 as a whole modifies the traditional one *only* by delaying matching applicant i as long as possible.[18] This accurately describes the full outcome matching since, as is well known, TTC is independent of the order in which cycles are chosen to be eliminated and matched. Formally:

Theorem 4.1. *Description 2 is an individualized dictatorship description of TTC.*

Proof. Fix an applicant i. By construction, Step (1) of Description 2 does not use $\succ_i$. Thus, to prove the theorem, it suffices to show that the set M in Step (1) is i's menu, and that the matching output in Step (3) is the outcome of TTC. We use the fact that TTC is independent of the order in which cycles are eliminated (Lemma D.9).

To see that M is i's menu, first observe that (by Lemma D.9) any institution matched during Step (1) is not on i's menu. Second, observe that after the conclusion of Step (1), once i points to *any* remaining institution, this must complete a cycle. Since this cycle necessarily involves applicant i, she gets the institution she pointed to, and thus this institution is on i's menu. See Figure 2 for an illustration.

[18] This can also be thought of as running TTC, with a twist: During the first stage, applicant i does not point to any institutions. This stage lasts until no cycles exist, after which i points as normal (and immediately gets matched to what she points to).

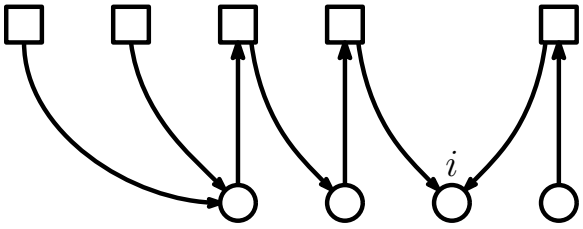


Figure 2: Illustration of the menu calculation in Description 2

Notes: Circles are applicants; squares are institutions; each applicant except i (resp. institution) points to her favorite remaining institution (resp. applicant).

To see that Step (3) calculates the outcome of TTC, observe that given the above (and given Lemma D.9), the entire run of Description 2 constitutes a valid ordering of cycle elimination in the traditional description of TTC. This prove the theorem. □

In addition to constructing a new menu description of TTC, Theorem 4.1 yields a simple proof that the traditional description of TTC is strategyproof. In particular, Theorem 4.1 demonstrates—given (only) the fact that TTC is independent of the order in which cycles are eliminated—that in the traditional description of TTC, any applicant i is matched according to a menu description. Hence, TTC is strategyproof.

The above simple proof is enabled by two crucial facts regarding Description 2. First, it contains a menu description. Second, it only slightly tweaks TTC's traditional (outcome) description. Crucially, a description cannot achieve these two tasks without being an outcome description that contains a menu description, i.e., an individualized dictatorship.

All told, our description of TTC—and the simple argument for the strategyproofness it provides—give new, promising ways to explain TTC's strategyproofness, both in the classroom, and for real-world market participants.[19]

5 Main Impossibility Result for DA

In this section, we present our third main result, which is our main impossibility theorem for DA. We show a formal sense in which—in contrast to what our individual dictatorship for TTC achieved in Section 4—no menu description yields a simple proof of the traditional description of DA's strategyproofness.

Concretely, we show that no slight tweak of the traditional description of DA is an individualized dictatorship. First, we formalize our notion of "no slight tweak."

[19] See footnote 6.

Second, we state and discuss our main impossibility theorem. Third, we present its proof.

5.1 Applicant-Proposing and Linear-Memory Descriptions

We now identify two simplicity desiderata satisfied by the traditional descriptions of SD, TTC, and DA. First, these descriptions only consider the preferences of each applicant once, in a specific, natural order—from favorite to least favorite. We call this property *applicant-proposing*. Second, they require a small amount of bookkeeping as they run—little more than the bookkeeping required to remember a single matching. We formalize this property through *linear-memory*, which stipulates that the bookkeeping used by the description is not much more than that of a single match.

Before formally defining these simplicity conditions, we illustrate how they are used to relay the traditional description of DA in one of its most celebrated practical applications: matching medical doctors to residencies in the US National Residency Matching Program (NRMP). Figure 3 shows a screenshot of a video that describes DA in this market by applying it to a small example. The explanation in the video is aided by two visual elements: crossing off institutions from applicants' lists as the description progresses, and keeping track of a "current tentative matching" illustrated by the yellow-highlighted names. These two simple visual elements are enabled precisely by our two simplicity desiderata.

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

Second, the linear memory requirement—which stipulates that the description use only a small amount of bookkeeping per applicant—is necessary for the yellow highlighting in the video to capture the entire required bookkeeping. This would not have been possible for a description that requires much more bookkeeping.

Without taking a specific stance on what a "slight tweak of DA" is, we take the stance that all slight tweaks of the traditional description of DA (and of SD and TTC) are—as leveraged in Figure 3—applicant-proposing and linear-memory. In particular, slight tweaks of traditional descriptions should read applicants' preferences in a similar

Figure 3: An illustration of the traditional description of DA through an example

Note: Screenshot taken from https://youtu.be/kVTwXNawpbk (NMS, 2020), a video produced by National Matching Services (the company providing matching software to the NRMP).

way, and should not require dramatically more bookkeeping.[20]

We now define applicant-proposing and linear-memory descriptions.[21]

Definition 5.1 (Applicant-proposing and Linear-memory)**.**

- In a matching environment, an description is *applicant-proposing* if it satisfies the following. The description is allowed to use applicants' preferences only by querying a single applicant at a time. (In contrast, the priorities of the institutions can be used by the description in any way.) For every applicant d and every possible input $(\succ_d, \succ_{-d})$, consider each time that d's preference $\succ_d$ is queried during the execution of the description. Then, for each $j = 1, 2, \ldots$, when the jth such query is made to $\succ_d$ in the description, this query depends only on the jth institution on $\succ_d$ (which is considered to be the "empty institution" if $\succ_d$ lists fewer than j institutions).[22]

[20] We emphasize that we do not view *every* applicant-proposing and linear-memory description as a slight tweak of a traditional one. (For example, in Section 6, we present linear-memory descriptions of DA which are inarguably much more complex than—and are not slight tweaks of—the traditional description.)

[21] As discussed in Section 2.2, we formally define descriptions to be algorithms. For a self-contained mathematical model of algorithms, see Appendix A.

[22] While we call this property "applicant-*proposing*," it also applies to descriptions of TTC that one

- The *memory requirement* of description, or any algorithm, is the logarithm in base 2 of the number of possible states of the algorithm. (This is precisely the number of bits required to represent the state; intuitively, it is the amount of extra bookkeeping or "scratch paper" required by the description.) In particular, we ignore the amount of memory required to store the input of the description, and measure the additional memory requirement.

  In a matching environment with n applicants and n institutions, we say a description is *linear-memory* if its memory requirement is at most $\widetilde{\mathcal{O}}(n)$.[23]

Linear memory is the minimal requirement for (either menu, or outcome descriptions of) matching mechanisms. Indeed, $\widetilde{\mathcal{O}}(n)$ is exactly (up to the precise logarithmic factors) the number of bits of memory required to describe a single matching (or a single applicant's menu).[24]

As discussed above, the traditional (outcome) descriptions of each of SD, TTC, and DA are applicant-proposing and linear-memory. Formally:

Observation 5.2. *Each of SD, TTC, and DA has an applicant-proposing and linear-memory outcome description.*

Additionally, the traditional description of SD, as well as our new description of TTC in Section 4, are individualized dictatorships. Since these description coincide with, or only slightly tweak, the corresponding traditional ones, they are applicant-proposing and linear-memory. Formally:

Corollary 5.3. *SD and TTC each have an applicant-proposing and linear-memory individualized-dictatorship description.*

5.2 Main Impossibility Theorem

We now present our main impossibility result. Using the simplicity desiderata of Section 5.1, we prove that no slight tweak of the traditional description of DA is an

might call "applicant-pointing", as well as to any other description that uses applicants' preferences (one time only) in favorite-to-least-favorite order.

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

[24] To see this formally, note that there are $n! = 2^{O(n \log n)}$ distinct matchings involving n applicants and n institutions (and exactly 2^n possible menus). Intuitively, this means that the number of letters it takes to write down a single matching with n applicants and n institutions (or, a subset of the n institutions) is roughly proportional to n.

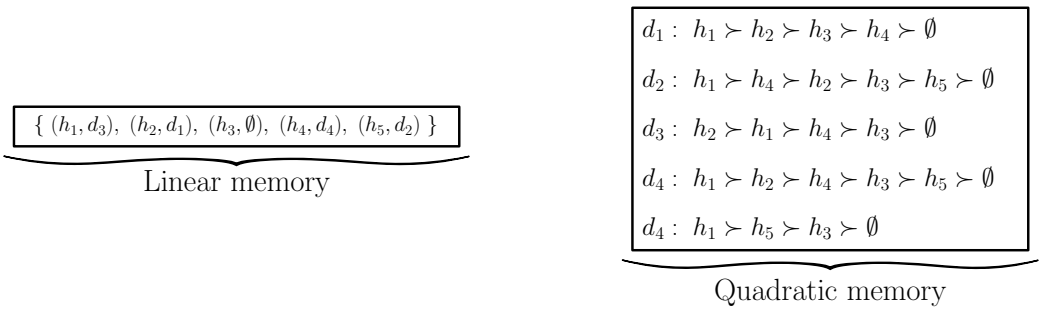
individualized dictatorship. Formally:

Theorem 5.4. *DA has no applicant-proposing, linear-memory individualized dictatorship. In fact, with n applicants and institutions, any applicant-proposing individualized dictatorship for DA requires $\Omega(n^2)$ memory.*[25]

Thus, in a precise sense, slight tweaks of the traditional description of DA cannot expose DA's strategyproofness, in a sharp contrast to (SD and) TTC (Corollary 5.3).

Additionally, our impossibility result is strong. Namely, we show that applicant-proposing individualized dictatorships for DA require *quadratic* memory—$\Omega(n^2)$ bits. This nearly matches the memory requirement of simply memorizing all applicants' preferences—$\widetilde{\mathcal{O}}(n^2)$ bits.[26] If an applicant-proposing description memorizes all applicants' preferences, then it can calculate *any* desired result (formally, by querying each applicant's entire preference list in order, with a separate state of the algorithm's memory for each possible preference profile, and returning a separate desired result for each such state). This shows that quadratic memory is the highest possible amount of memory that an algorithm might require. Thus, where applicant-proposing individualized dictatorships of (SD and) TTC use memory as *low* as possible (linear, see Section 5.1), for DA the memory requirement is as *high* as possible (quadratic). See Figure 4 for an illustration of the qualitative gap between these two memory requirements.

Figure 4: Linear versus quadratic memory



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

[26] To see this formally, observe that there are $(n!)^n = 2^{O(n^2 \log(n))}$ possible preference profiles for all applicants. Intuitively, this means that the number of letters it takes to write down n applicants' preferences over all n institutions is roughly proportional to n^2.

Theorem 5.4 is also tight in the following sense. The theorem shows that descriptions of DA cannot simultaniously satisfy four criteria: being an outcome description, containing a menu description, being applicant-proposing, and using linear-memory. The impossibility only holds when all four of these criteria are assumed. We establish this as follows. First, DA's traditional description is an applicant-proposing, linear-memory outcome description. Second, DA has an applicant-proposing *quadratic memory* individualized dictatorship (i.e., an outcome description containing a menu description), since (as discussed above) quadratic-memory is as high as possible. Third and fourth, we show below in Section 6 that DA has an applicant-proposing linear-memory menu description, and a linear-memory individualized dictatorship which is not applicant-proposing.[27]

All told, there is a stark trichotomy in our framework between SD, TTC, and DA. The traditional description of SD is already a menu description, simultaneously for all applicants, exposing its strategyproofness easily.[28] The traditional description of TTC must be slightly tweaked and specialized to each individual applicant in order to expose strategyproofness.[29] However, once this is done, strategyproofness is easy to see. For DA, in contrast with both other mechanisms, no small tweak of the traditional description suffices to expose strategyproofness through menus, in the robust and strong sense provided by Theorem 5.4. See Figure 1 in the introduction for an illustration.

5.3 Proof of Main Impossibility Theorem

Theorem 5.4 states that applicant-proposing individualized dictatorships for DA require high memory. Recall that such descriptions must—while querying applicants' preferences only once in favorite-to-least-favorite order—calculate i's menu using $\succ_{-i}$, and then proceed to calculate the full matching using $(\succ_i, \succ_{-i})$.

To prove Theorem 5.4, we construct a set of applicant preferences that, intuitively speaking, has two properties: (A) to calculate i's menu given preferences in this set,

[27] We construct an applicant-proposing linear-memory menu description of DA in Section C.2. We construct a linear-memory individualized dictatorship for DA in Section C.3; this description is *institution*-proposing.

[28] SD has an (S)OSP implementation (Li, 2017; Pycia and Troyan, 2023) for a similar reason.

[29] We also observe that, unlike SD, a description of TTC *must* be specialized to a given applicant i in order to contain a menu description for i; we formalize this argument in Remark B.4. The fact that TTC is not OSP-implementable (Li, 2017) is a generalization of this observation.

essentially the full preference list of every applicant other than i must be read in its entirety, and (B) to calculate the final matching, essentially all this information must be remembered in full. Property (B) then establishes the $\Omega(n^2)$ memory requirement.

Proof of Theorem 5.4. Fix an applicant i and let D be any applicant-proposing individualized dictatorship description of DA for i.

We now describe a set $\mathcal{S} \subseteq \mathcal{T}_{-i}$ of possible inputs to DA, illustrated in Figure 5, which allows us to establish property (B) discussed above (intuitively, by allowing i's possible reports to affect the outcome matching in a different way for each different $\succ_{-i} \in \mathcal{S}$). For simplicity, let the number of applicants and institutions n be a multiple of 4. Other than i, there are applicants and institutions d_j, d'_j, h_j, h'_j for each $j \in \{1, \ldots, n/2\}$. There are $n/2$ total "cycles" containing two applicants and two institutions each. Cycle j has applicants d_j and d'_j and institutions h_j and h'_j. The cycles are divided into two classes, "top" cycles (for $j \in \{1, \ldots, n/4\}$) and "bottom" cycles (for $j \in \{n/4+1, \ldots, n/2\}$).

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

For top cycles ($j \leq n/4$):

h_j : $d'_j \succ i \succ d_j$

h'_j : $d_j \succ d'_j$

For bottom 2-cycles ($j > n/4$):

h_j : $d'_j \succ d_1 \succ d_2 \succ \ldots \succ d_{n/4} \succ d_j$

h'_j : $d_j \succ d'_j$.

For the top cycle applicants (d_j with $j \leq n/4$), the preferences vary (in a way we will specify momentarily). Other applicants' preference are fixed, as follows:

For bottom cycles ($j > n/4$): d_j : $h_j \succ h'_j$

For all cycles ($j \in \{1, \ldots, n/2\}$): d'_j : $h'_j \succ h_j$.

Let $\mathcal{S}$ denote the set of preference profiles where we additionally have:

For top cycles ($j \leq n/4$): d_j : $h_j \succ B_j \succ h'_j$,

where B_j is an arbitrary subset of $\{h_k \mid k > n/4\}$, ranked in any fixed order (say, increasing order of j). Any such collection of $\left(T_j\right)_{j=1}^{n/4}$ defines a distinct preference profile in $\mathcal{S}$. Note that $|\mathcal{S}| = 2^{(n/4)^2}$. See Figure 5 for an illustration.

We additionally define a set of inputs $\mathcal{S}' \supseteq \mathcal{S}$, which allow us to establish property

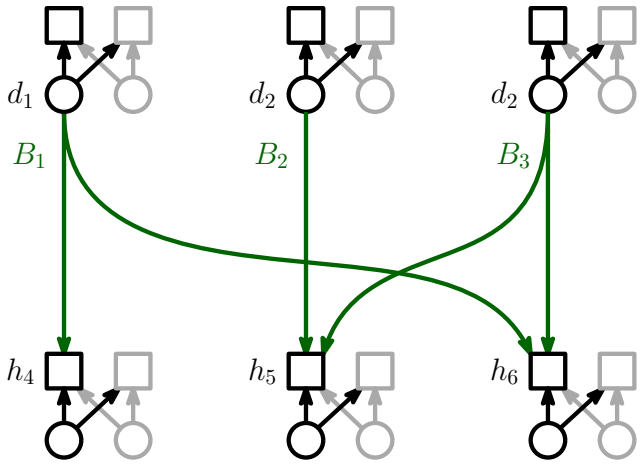


Figure 5: Illustration of the construction used to prove Theorem 5.4.
Notes: Dark nodes represent d_j or h_j for some j, and grey nodes represent d'_j or h'_j. The green arrows directed outwards from a top cycle d_j represent the sets B_j.

(A) discussed above (intuitively, by making i's menu depend on the final institution ranked above $\emptyset$ on other applicants' lists). Specifically, let $\mathcal{S}'$ denote the set containing every element of $\mathcal{S}$, and additionally any top cycle applicant d_j ($j \leq n/4$) may or may not truncate the final institution h'_i off her list, marking it as unacceptable. In other words, in addition to the sets $\left(B_j\right)_{j=1}^{n/4}$, an element of $\mathcal{S}'$ is defined by bits $(c_j)_{j=1}^{n/4}$, such that, for each top cycle j ($j \leq n/4$):[30]

$$\begin{aligned} &\text{When } c_j = 0: && d_j: \quad h_j \succ B_j \succ h'_j \\ &\text{When } c_j = 1: && d_j: \quad h_j \succ B_j. \end{aligned}$$

We now proceed to prove the two crucial properties of DA, and the description D, when run on this family of preference profiles. The following lemmas formalize, respectively, properties (A) and (B) discussed above.

Lemma 5.5. *Consider a preference profile in $\mathcal{S}'$. For each top cycle j (with $j \leq n/4$), we have that h_j is in applicant i's menu if and only if d_j does not rank h'_j (i.e., $c_j = 1$). Hence, to correctly calculate i's menu, description D must read the entire preference list of each such d_j (up to the position of h'_j).*

To prove this lemma, consider the execution of APDA when i submits a list containing only h_j. First, d_j is rejected, then she proposes to every institution $h_k \in B_j$. This "rotates" the bottom cycle containing h_k; in more detail, h_k will accept the proposal from d_j, then d_k will propose to h'_k, then d'_k with propose to h_k, and d_j will

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

be rejected from h_k. This will occur for every $h_k \in B_j$, so d_j will not match to any h_k with $k \in \{n/4+1, \ldots, n/2\}$.

Finally, after getting rejected from each institution in B_j, applicant d_j may or may not propose to h'_j, depending on the bit c_j. If she does not, then d_* remains matched to h_j and in this case h_j is on i's menu. If she does, then h'_j will reject d'_j, who will propose to h_j, which will reject i. So i will go unmatched, and thus in this case h_j is not on i's menu.

The final sentence of the lemma then follows from the fact that D is an applicant-proposing and must calculate i's menu. This proves Lemma 5.5.

Lemma 5.6. *For each distinct preference profile $\succ_{-i} \in \mathcal{S}$ induces a distinct function $APDA(\cdot, \succ_{-i}) : \mathcal{T}_{-i} \to A$ from applicant i's report to outcome matchings. Hence, to correctly calculate the outcome matching, the description D must—across all states where it has calculated i's menu—include (at least) one state for each element of $\mathcal{S}$.*

To prove this lemma, consider two distinct preference profiles in $\mathcal{S}$, one profile $\succ_{-i}$ corresponding to $\left(B_j\right)_{j=1}^{n/4}$, and the other profile $\succ'_{-i}$ corresponding to $\left(B'_j\right)_{j=1}^{n/4}$. Without loss of generality, there must be some j and k such that $h_k \in B_j \setminus B'_j$. Suppose now that i's report $\succ_i$ lists only h_j. Then, consider execution of APDA under $(\succ_i, \succ_{-i})$ and under $(\succ_i, \succ_{-i})$. Under $\succ_{-i}$, the bottom tier cycle containing h_k will be "rotated," i.e. since $h_k \in B_j$, the sequence of rejections will cause h_k to match to d'_k. However, this is not the case under $\succ'_{-i}$, since $h_k \notin B_j$. Thus, $APDA(\cdot, \succ_{-i}) \neq APDA(\cdot, \succ'_{-i})$.

We now prove the final sentence of the lemma. As argued in Lemma 5.5, D must have read all top cycle applicants' preferences in order to calculate i's menu. Moreover, since D is an individualized dictatorship, it must do so before learning $\succ_i$, and hence must remember the entirety of $\left(B_j\right)_{j=1}^{n/4}$. This proves Lemma 5.6.

We now prove Theorem 5.4. Together, Lemma 5.5 and Lemma 5.6 show that when D has just calculated the menu of applicant i, the description must be in a distinct state for each distinct $\succ_{-i} \in \mathcal{S}$. There is one such $\succ_{-i}$ for each possible way of assigning the sets $B_j \subseteq \{h_k \mid k > n/4\}$ for all $j \in \{1, \ldots, n/4\}$. There are $2^{(n/4)^2} = 2^{\Omega(n^2)}$ possible ways to set this collection $\left(B_j\right)_{j=1}^{n/4}$. Thus, the description requires at least this many states, and thus requires memory $\Omega(n^2)$. $\square$

6 Classification of Descriptions of DA

In this section, we present our fourth main result, which is an additional negative finding for DA. We examine a broad classification of mechanism descriptions, and find that—other than the traditional description and Description 1—DA has no additional simple descriptions within this classification.

The section proceeds as follows. First, we introduce our classification of mechanism descriptions, and in fact, uncover additional (unappealing and technical) descriptions of DA within this classification. Second, we introduce a new simplicity desideratum, termed locality, refining those of Section 5.1. Third, we prove our additional impossibility results: none of the additional descriptions can satisfy locality.

6.1 Additional Descriptions of DA

The concepts introduced throughout our paper thus far suggest a classification of matching mechanism descriptions with six classes. First, we can consider different ways the description can read its inputs: applicant-proposing descriptions (like the traditional one), and institution-proposing descriptions (Description 1).[31] Second, we can consider our three description outlines: menu descriptions, outcome descriptions, and individualized dictatorships. See Table 3 for a presentation of all our results for DA, placed into this classification.

Our main impossibility theorem (Theorem 5.4) considers one such class of descriptions (namely, applicant-proposing individualized dictatorships), and shows such descriptions cannot be linear-memory. Perhaps surprisingly, in Appendix C we construct linear-memory descriptions of DA in *every* class not ruled out by Theorem 5.4. That is, we construct the following linear-memory descriptions of DA:

- An institution-proposing outcome description (Section C.1, adapted from an algorithm used by Ashlagi et al. (2017)).
- An applicant-proposing menu description (Section C.2).
- An institution-proposing individualized dictatorship (Section C.3).

[31] While we have not formally defined institution-proposing descriptions, we use this term to mean the analogous definitions to Definition 5.1, in which sides of the market are interchanged.

Table 3: Classification of descriptions of DA

	Menu Description	Outcome Description	Individualized Dictatorship
Applicant proposing	In Section C.2. Necessarily **non-local** by Theorem 6.2.	Traditional DA algorithm	**Completely impossible** by Theorem 5.4.
Institution proposing	Description 1 in Section 3	In Section C.1 / Ashlagi et al. (2017). Necessarily **non-local** by Theorem 6.3.	In Section C.3. Necessarily **non-local** by Theorem 6.3.

Notes: We look for linear-memory descriptions of DA, and ask whether such descriptions can be local. We consider descriptions which either read preferences in an applicant-proposing manner or read priorities in an institution-proposing manner. We consider three description outlines: menu descriptions (exposing strategyproofness), outcome descriptions (conveying the fully matching, hence exposing feasibility), or individualized dictatorship (exposing both).

Unfortunately, in a stark contrast to the traditional description of DA and our Description 1, every new description we construct in Appendix C is a delicate and technical algorithm which requires careful, likely-unintuitive bookkeeping to maintain its linear-memory. Thus, these algorithms seem impractical. However, this does not necessarily imply that there are no other, more attractive, descriptions of these types. We address this (im)possibility in the next section.

6.2 Local One-Side-Proposing Descriptions

We now give precise ways in which all of the additional descriptions discussed in Section 6.1 *must* be convoluted: they cannot satisfy desiderata we call *locality* properties. Like linear-memory (introduced in Section 5.1), these desiderata concern the bookkeeping used by the description. However, where linear-memory restricts the *amount* of bookkeeping used, locality restricts the *manner* in which the the bookkeeping is updated and used. Namely, locality stipulates that the description can update bookkeeping concerning some agent only when making queries directly related to that agent.

More technically, we call an applicant-proposing outcome description (resp., menu description for applicant i) *local* if (a) in addition to any global bookkeeping, it

also maintains local bookkeeping for each institution, and this bookeeping is only updated when that institution is read from any applicant's list, and (b) the calculated match of an institution (resp., whether the institution is on applicant i's menu) only depends on the final state of the local bookkeeping for this institution. The definition of local bookkeeping for institution-proposing descriptions interchanges the roles of applicants and institutions, and locality analogously requires that the calculated part of the output relevant to an applicant depend only on the local bookkeeping for that applicant.

Formally, we define our locality desiderata as follows:

Definition 6.1.

- In a matching environment, *local bookkeeping* for an applicant-proposing description D is a profile of labels $\big(L_h(v)\big)_{v,h}$ for every institution h and every possible state v of the description, with the following property. In any execution of the description on any inputs, the labels $\big(L_h(v)\big)_{v,h}$ change only when the description queries some applicant's preference. Moreover, when the description queries an applicant's preference list, say that of applicant d, and transitions from state v to state v', then $\big(L_h(v)\big)_h$ must differ from $\big(L_h(v')\big)_h$ only at the institution h returned by the preference query. Thus, if the preference query returns institution h_0, then we must have $L_h(v) = L_h(v')$ for all $h \neq h_0$.

- An applicant-proposing outcome description D is *local* if is has local bookkeeping such that for every terminal state v in D (i.e., the state D reaches when it concludes its execution and returns a result) and for every institution h, the match of h depends only on $L_h(v)$.

- An applicant-proposing menu description D for applicant i is *local* if it has local bookkeeping such that for every terminal state v in D and for every institution h, whether h is on i's menu or not depends only on $L_h(v)$.

- The definition of local bookkeeping for an institution-proposing description is completely analogous to that of an applicant-proposing one, interchanging the roles of applicants and institutions. In particular, there is a label $L_d(v)$ at every state v for every applicant d, and and the label of applicant d can only change when d is the result of a query to some institution's priority list.

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

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

The traditional description of DA is a local applicant-proposing outcome description, and Description 1 is a local institution-proposing menu description. In contrast, each of the (convoluted, yet linear-memory) descriptions that we present in Appendix C is non-local.

Like linear-memory, locality captures one feature of the traditional description of DA which is used to explain the mechanism in practice. For instance, in the description depicted in Figure 3 in Section 5.1, locality reflects the fact that the description updates the tentative match of instituion h only when some applicant's next-highest not-yet-read institution equals h. However, while linear-memory is a quite flexible condition, locality seems more restrictive. For example, our menu description of TTC in Description 2 is non-local, and yet seems nearly as simple as, and indeed a slight tweak of, the traditional description of TTC.[32] Thus, in contrast to linear-memory, locality does not seem to be flexible enough to capture all reasonable slight tweaks of the traditional description of matching mechanisms. However, non-locality rules out local versions of each of the (unintuitive, delicate) descriptions we present in Appendix C, strongly suggesting that more-practical descriptions of their types do not exist.

6.3 Additional Impossibility Theorems for DA

We now prove our additional impossibility theorems for DA, which say that except for the traditional description and Description 1, all descriptions of DA in our classification must be non-local. Formally:

[32] To see why Description 2 is non-local, observe that the description can learn that h_1 is on i's menu when some applicant d points to some institution $h_2 \neq h_1$ (for example, this is the case when i has top priority at h_2, and d has top priority at h_1).

Theorem 6.2. *If there are at least three applicants and three institutions, then for every applicant i there exist priorities of the institutions such that any applicant-proposing menu description of DA for applicant i is non-local.*

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

The proofs are deferred to Appendix B. For both of these proofs, we construct an explicit instance of DA such that, given the relevant one-side-proposing constraint, the locality constraint must be violated.

A direct corollary of these results is that no local one-side-proposing individualized dictatorship exists for DA, as such a description contains either an applicant-proposing menu description, or is an institution-proposing outcome description.

Corollary 6.4. *DA has no local individualized dictatorship (application- or institution-proposing).*

All told, our results show that simple one-side-proposing descriptions of DA face a formal tradeoff between conveying the full matching (as in the traditional description) and conveying strategyproofness (as in Description 1).

7 Related work

Our paper is most directly inspired by the contemporary "strategic simplicity" program in mechanism design theory, which largely considers different dynamic implementations of mechanisms. A cornerstone of this literature is Li (2017), which introduces obviously strategyproof (OSP) mechanisms as a way to expose strategyproofness. Unfortunately, TTC (Li, 2017) and DA (Ashlagi and Gonczarowski, 2018) do not have OSP mechanisms (except in rare special cases of institutions' priorities; see Troyan, 2019; Mandal and Roy, 2021; Thomas, 2021).[33]

In contrast to the above literature, we consider different ex ante descriptions of (static, direct-revelation) mechanisms. Breitmoser and Schweighofer-Kodritsch

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

(2022) provide empirical evidence that framing a static auction as an OSP (ascending-clock) auction can be effective towards conveying strategyproofness. Since DA and TTC do not have OSP implementations, they cannot be framed in this way. Nonetheless, by relaying the match of only a single applicant, menu descriptions frame the mechanism in a way that is OSP for that applicant (and in fact *strongly* OSP; Pycia and Troyan 2023).

The experimental paper of Katuščák and Kittsteiner (2020) also suggests describing matching mechanisms to participants via menu descriptions, but does not investigate any menu description beyond that of Example 2.9, which essentially calculates the menu by iterating over all possible reports and running the traditional mechanism description each time.

We are not aware of any prior characterizations of the menu in DA. Our characterization builds on a large literature developing techniques for reasoning about stable matchings.[34] The menu in DA is different than other commonly considered definitions in the theory of stable matching, such as applicant i's set of stable partners (Gale and Shapley, 1962) or her budget set (Segal, 2007; Azevedo and Leshno, 2016; Azevedo and Budish, 2019; Immorlica et al., 2020). In particular, in finite matching markets, these other commonly-considered sets depend on applicant i's report, and hence do not equal i's menu. We provide explicit examples and more discussion in Remark B.5 and Remark B.6.

Proposition 2 in Leshno and Lo (2021) characterizes the menu in TTC in a different way from our Description 2. Their characterization does not give an individualized dictatorship for TTC, and hence cannot be used in the same way as Description 2 to derive a simple proof of the strategyproofness of TTC's traditional description.

Our paper is also loosely inspired by the literature within computer science studying menus. These works largely focus on single-player selling mechanisms (e.g., Hart and Nisan, 2019; Daskalakis et al., 2017; Babaioff et al., 2022; Saxena et al., 2018;

[34] In particular, our proof of Theorem 3.2 in Appendix B analyzes DA by incrementally modifying preference lists. Similar techniques appear in Gale and Sotomayor (1985); Teo et al. (2001); Immorlica and Mahdian (2005); Hatfield and Milgrom (2005); Gonczarowski (2014); Ashlagi et al. (2017); Cai and Thomas (2022), for example. Our proof of Theorem 3.2 in Section 3 uses the strategyproofness of DA; to our knowledge, this is a fairly novel technique.

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

Gonczarowski, 2018).[35] Papers considering menus in multi-player mechanisms include Dobzinski (2016) and Dobzinski et al. (2022), who use menus as a tool for bounding communication complexity. We do not know of any prior algorithmic work on menus of matching mechanisms, nor of any prior work that analyzes different ways to describe multi-player mechanisms in terms of menus.

The present paper is part of our broader research agenda. In the working paper version (Gonczarowski, Heffetz, and Thomas, 2023), we consider more general environments, study a basic extension of our theory for auctions, and conduct an experiment for a second-price auction and median voting. The theoretical computer science paper Gonczarowski and Thomas (2024) investigates a number of complexity questions related to our main theorems (particularly, to Theorems 3.2, 4.1, and 5.4).

Most relevantly, the empirical companion paper Gonczarowski, Heffetz, Ishai, and Thomas (2024) investigates participants' responses to two different descriptions of DA—namely, the traditional one, and Description 1 (our menu description). We find evidence that, while Description 1 is significantly more complex for participants than the traditional one, many participants can understand Description 1 and calculate its outcomes. Interestingly, while levels of strategyproofness-understanding are similar under both descriptions of DA, we see very high levels of strategyproofness-understanding under a less-complex, stripped-down menu description which omits the details of how the menu is calculated. This stripped-down menu description—which relays *only* strategyproofness—yields even higher strategyproofness-understanding levels than a description of strategyproofness inspired by textbook definitions, suggesting it as a promising practical approach for conveying strategyproofness. Furthermore, if Description 1 is included as an optional, more-detailed explanation, then participants interested in these details can understand both the matching process itself, and why this process guarantees strategyproofness.

8 Conclusion

Strategyproofness has long been proposed as a way to make mechanisms fair by leveling the playing field for players who do not strategize well (Pathak and Sönmez, 2008). We warmly embrace this agenda. However, we observe that if participants'

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

do not all *understand* strategyproofness, then disparities may remain. Menu descriptions aim to improve this understanding. From a menu description, participants can directly see why the mechanism is strategyproof, offering an alternative to status-quo tactics such as appeals-to-authority asserting that the mechanism is strategyproof.[36]

While menu descriptions expose strategyproofness, they may obscure other properties of the mechanism. For example, since Description 1—our menu description of DA—relays each applicant's match separately, it is unclear why this description always produces a feasible matching, a fact which is clear in the traditional outcome description.[37] Description 2—our individualized dictatorship for TTC—might be used to simultaneously convey strategyproofness and feasibility. However, for DA, our impossibility theorems suggest a sense in which, for a broad class of simple descriptions, the tradeoff between conveying strategyproofness and conveying feasibility is unavoidable.

In this paper and its experimental companion (Gonczarowski et al., 2024), we suggest that some principled alternative framings of mechanisms (namely, menu descriptions) might better convey their properties (namely, strategyproofness), and we analyze such framings theoretically and empirically. Beyond these specifics, we view our general premise—changing mechanism descriptions and formally evaluating these descriptions' simplicity—as a valuable suggestion in its own right. Many future directions remain in the study of how to best present mechanisms to participants. One natural empirical idea is to present TTC to lab participants using our Description 2, or (as suggested by Morrill and Roth, 2024) to use this description to explain the strategyproofness of TTC's traditional one. Future theoretical work may study other properties one might wish to expose (e.g., fairness or optimality) and study opportunities and tradeoffs for exposing these properties in a variety of different settings.

[36] One common prior approach taken by clearinghouses is to encourage straightforward reporting without explaining strategyproofness. For example, Dreyfuss et al. (2022) notes that an informative video by the National Resident Matching Program (NRMP) was formerly introduced with the text:

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

[37] While traditional mechanism descriptions require participants to trust the description (as noted in, e.g., Akbarpour and Li, 2020), the fact that menu descriptions obscure feasibility may influence some participants' levels of trust. While our work focuses on understanding, trust may be an important direction for future theoretical or empirical work.

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Y. A. Gonczarowski and C. Thomas. Structural complexities of matching mechanisms. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 455–466, 2024.

Y. A. Gonczarowski, O. Heffetz, and C. Thomas. Strategyproofness-exposing mechanism descriptions. Earlier working paper version of the current article, 2023.

Y. A. Gonczarowski, O. Heffetz, G. Ishai, and C. Thomas. Describing deferred acceptance and strategyproofness to participants: Experimental analysis. Working Paper, 2024.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

T. Morrill and A. E. Roth. Top trading cycles. *Journal of Mathematical Economics*, 112: 102984, 2024.

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

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

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

A. Rees-Jones and R. Shorrer. Behavioral economics in education market design: A forward-looking review. *Journal of Political Economy Microeconomics*, 1(3):557–613, 2023.

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

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

C.-P. Teo, J. Sethuraman, and W.-P. Tan. Gale-shapley stable marriage problem revisited: Strategic issues and applications. *Management Science*, 47(9):1252–1267, 2001.

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

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

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

Appendix

A Detailed Formal Model of Algorithms

In this appendix, we define from first-principles a mathematical model of descriptions of mechanisms which can express all our results.

We introduce the notion of an *extensive-form description*. For generality, we state this definition in terms of a general mechanism design environment with players $1, \ldots, n$, type spaces $\mathcal{T}_1, \ldots, \mathcal{T}_n$, and outcome space A. At a technical level, an extensive-form description is similar to an extensive-form mechanism, except that different branches may "merge," i.e., the underlying game tree is actually a directed acyclic graph (DAG).[1] Note, however, that the interpretation is different from that of an extensive-form mechanism: Rather than modeling an interactive process where the players may act multiple times, an extensive-form description spells out the steps used to calculate some result by iteratively querying the directly-reported types of the players.

We are interested in three types of extensive-form descriptions, corresponding to our three description outlines: outcome descriptions, menu descriptions, and individualized dictatorships.

Definition A.1 (Extensive-Form Descriptions)**.**

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

[1] Alternatively, extensive-form descriptions can be viewed as finite automata where state transitions are given by querying the types of players.

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

$(t_1, \ldots, t_n) \in \mathcal{T}_1 \times \ldots \times \mathcal{T}_n$ is defined as follows: Start in the root vertex s, and whenever reaching any non-terminal vertex v that queries a player i and has transition function ℓ_v, follow the edge $\ell_v(t_i)$.

- An *extensive-form outcome description* of a mechanism f is an extensive-form description in which each terminal vertex is labeled by an outcome, such that for each type profile $(t_1, \ldots, t_n) \in \mathcal{T}_1 \times \mathcal{T}_n$, the terminal vertex reached by following the evaluation path on $t \in T$ is labeled by the outcome $f(t_1, \ldots, t_n)$.

- An *extensive-form menu description* of a social choice function f for player i is an extensive-form description with $k+1$ layers, such that (a) each vertex preceding layer k queries some player other than i, (b) each vertex v in layer k queries player i and is labeled by some set $M(v) \subseteq A_i$, such that if v is on the evaluation path on a type profile $(t_1, \ldots, t_n) \in \mathcal{T}_1 \times \mathcal{T}_n$, then $M(v) = \mathcal{M}_{t_{-i}}$ is the menu of player i with respect to t_{-i} in f, and (c) each (terminal) vertex v in the final layer $k+1$ is labeled by an outcome for player i,[3] such that if v is reached by following the evaluation path on a type profile $(t_1, \ldots, t_n)$, then v is labeled by i's outcome in $f(t_1, \ldots, t_n)$.

- An *extensive-form individualized dictatorship (description)* of f for player i is an extensive-form outcome description such that, for some k, the first $k+1$ layers are an extensive-form menu description.

For a concrete example of an extensive-form description, we consider a menu description of a second price auction.[4] In this mechanism, a bidder's menu consists of two options: winning the item and playing the highest bid placed by any other bidder, or winning nothing and paying nothing. Thus, a menu description can be given as follows:

(1) Your "price to win" the item will be set to the highest bid placed by any other player.

[3] Formally, in a general mechanism design environment, an *outcome of player i* (or, an *i-outcome*) is a maximal set E of outcomes such that all possible types of player i in $\mathcal{T}_i$ view each outcome in E as equally desirable.

[4] While we have not formally defined menus or menu descriptions in non-matching environments, they naturally generalize by considering the menu of i induced by reports t_{-i} to be the set of i's outcomes consistent with t_{-i}.

(2) If your bid is higher than this "price to win," then you will win the item and pay this price. Otherwise, you will win nothing and pay nothing.

An extensive-form description can formalize this menu description by querying the other bidders one-by-one, while keeping track of only the highest bid placed by any of them. Figure A.1 provides an illustration.

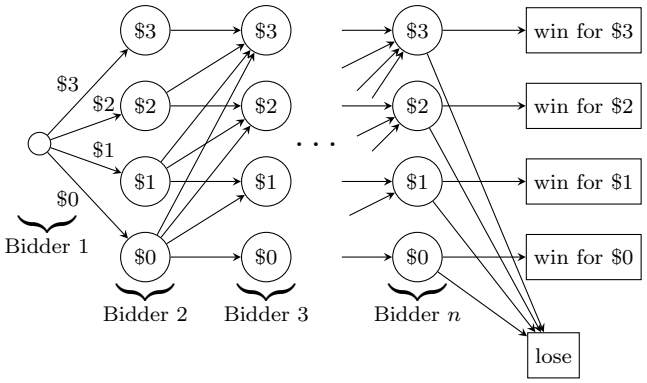


Figure A.1: An extensive-form menu description for bidder n in a second-price auction

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

More broadly, any precise algorithm taking players types as inputs induces an extensive-form description in a natural way: the vertices in layer j are the possible states of the algorithm after querying the types of different players altogether j times. In particular, our positive results (Description 1 and Description 2) correspond to extensive-form descriptions. The definitions of all our simplicity desiderata (one-side-proposing, linear-memory, and local) also extend naturally to extensive-form descriptions. Moreover, the proofs of all our impossibility theorems (Theorem 5.4, Theorem 6.2, and Theorem 6.3) hold, mutatis mutandis, for the relevant class of extensive form descriptions.

In addition to providing a self-contained mathematical language for expressing our results, the definition of an extensive-form description allows us to clarify some ways in which our impossibility results are strong. Namely, while algorithms are often required to work for any number of players, our impossibility results hold even if one can use a separate extensive-form description for each number of players n (in computer-science terminology, an extensive-form description can be nonuniform),[5] and regardless of the computational complexity of such a description. Relatedly, our

[5] In Theorem 5.4 and Theorem 6.2, our proof use a fixed profile of institutions' priorities. Thus, these impossibilities hold even if one can use a separate extensive-form description for every fixed profile of priorities. (A similar remark holds for Theorem 6.3 regarding a fixed profile of applicants' preferences.)

impossibility result follow from direct combinatorial arguments and do not depend on any complexity-theoretic conjectures such as $P \neq NP$.

B Omitted Proofs and Remarks

In this appendix, we provide proofs and remarks omitted from the main text. For the reader's convenience, we restate results before giving proofs.

We start by proving Theorem 3.2 without assuming the strategyproofness of DA. This providing an alternative, instructive approach for proving DA's strategyproofness. For additional completeness, known results used in the proof are stated with full proofs in Appendix D.

Theorem 3.2. *Description 1 is a menu description of DA.*

Direct Proof. For this proof, we denote applicant i—to whom the menu is described—by d_*. We also let P be the preference profile, let P_{d_*} denote d_*'s preference in P. Let $h_* = APDA_{d_*}(P)$ denote the match of d_* according to applicant-proposing DA. We wish to show that h_* is the P_{d_*}-favorite institution in the set containing (1) the "outside option" of going unmatched, and (2) all institutions h such that h prefers d_* to $IPDA_h(P_{-d_*})$ (the match of h according to institution-proposing DA in the market without d_*).

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

1. If $h_* \neq \emptyset$, then then h_* proposes to d_* during $IPDA(P|_{d_*:\emptyset})$.

2. d_* gets no proposal in $IPDA(P|_{d_*:\emptyset})$ that is P_{d_*}-preferred to h_*.

We start with the first claim. Assume that $h_* \neq \emptyset$. Let $P|_{d_*:\{h_*\}}$ denote the preference profile obtained by altering P so that d_* reports a preference list consisting only of h_* (i.e., marking all other institutions as unacceptable). Observe that $APDA(P)$, the applicant-proposing DA outcome for preferences P, is stable under preferences $P|_{d_*:\{h_*\}}$. Thus, by the Lone Wolf / Rural Hospitals Theorem (Roth,

1986, see Theorem D.6), since d_* is matched in $APDA(P)$, she must be matched in $IPDA(P|_{d_*:\{h_*\}})$ as well. Thus, $IPDA(P|_{d_*:\{h_*\}}) = h_*$. Since regardless of the order in which we choose to make proposals in DA, the same proposals are made and the same outcome is reached (Dubins and Freedman, 1981, see Corollary D.3), the following is a valid run of $IPDA(P|_{d_*:\emptyset})$: first run $IPDA(P|_{d_*:\{h_*\}})$, then have d_* reject h_*, then continue running (according to $P|_{d_*:\emptyset}$) until IPDA concludes. Thus, h_* proposes to d_* during $IPDA(P|_{d_*:\emptyset})$, proving the first claim.

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

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

Remark B.1. As noted in Section 3, Theorem 3.2 extends to many-to-one markets with substitutable priorities. To quickly see why this extension holds in the special case in which institutions have responsive preferences (i.e., the special case in which each institution has a master preference order and a capacity), fix a many-to-one market, and following a standard approach, consider a one-to-one market where each institution from the original market is split into "independent copies." That is,

the number of copies of each institution equals the capacity of the institution, each "copied" institution has the same preference list as the original institution, and each applicant ranks all the copies of the institution (in any order) in the same way she ranked the original institution. Ignoring the artificial difference between copies of the same institution, the run of applicant-proposing DA is equivalent under these two markets. Thus, an applicant's menu is equivalent under both markets, and so by Theorem 3.2, a menu description for the many-to-one market can be given through institution-proposing DA under the corresponding one-to-one market, which in turn is equivalent to institution-proposing DA under the original market (where at each step, each institution proposes to a number of applicants up to its capacity). The only change in Description 1 in this case would be replacing the condition $d \succ_h \mu_{-d}(h)$ with $\exists d' \in \mu_{-d}(h) : d \succ_h d'$.

Remark B.2. As additionally noted in Section 3, Theorem 3.2 also extends to many-to-one markets with contracts in which the institutions have substitutable preferences that satisfy the law of aggregate demand (the conditions under which Hatfield and Milgrom (2005) prove that the strategyproofness of applicant-proposing DA and the rural hospitals theorem hold), as shown in Description A.1. This new description generalizes Description 1 as follows: (1) Description A.1 uses the generalized Gale–Shapley algorithm of Hatfield and Milgrom (2005) starting from $(\emptyset, X)$ (where X is the set of all possible contracts) to calculate the institution-optimal stable outcome without d_* to get a matching μ_{-d_*}. (2) A given contract $c = (d_*, h, c)$ (i.e., an $(\text{applicant}, \text{institution}, \text{term})$ tuple) is on d_*'s menu if and only if h would choose (d_*, h, c) if given a choice from the set containing (d_*, c) and its matches in μ_{-d_*} (in the notation of Hatfield and Milgrom (2005), $c \in C_h(\mu_{-d_*}(h) \cup \{c\})$). Under this modification, each step of the proof of Theorem 3.2 in Section 3 holds by a completely analogous argument for this market.

Remark B.3. In this remark, we show how Theorem 3.2, which characterizes the menu in DA in terms of Description 1, can be used to prove results from Ashlagi et al. (2017) via arguments similar to Cai and Thomas (2022). Consider a randomized market with $n + 1$ applicants and n institutions, where such that each applicant/institution draws a full-length preference list uniformly at random, and let μ be the result of (applicant-optimal) DA with these preferences. We prove that the expected rank each applicant receives on their preference list (formally, the expectation

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

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

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

of $|\{h : h \succeq_d \mu(d)\}|$ for any d) is at least $(1-\epsilon)n/\log(n)$ for any $\epsilon > 0$ and large enough n.

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

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

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

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

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

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

Remark B.4. One now formally show that, unlike SD, a description of TTC *must* be specialized to individual applicants in order to contain a menu description for them.

To do this, it suffices to construct an instance containing two applicants d_1 and d_2 such that each of their menus depends on the other. For example, consider an instance where $h_i : d_i \succ d_{3-i}$ and $d_i : h_{3-i} \succ h_i$ for $i \in \{1, 2\}$. Under this instance, for each $i \in \{1, 2\}$, institution h_{3-i} is on d_i's menu, but if applicant d_{3-i} changed her preference list, this would no longer be true. Hence, a description cannot calculate either applicant's menu before the description queries the other applicant's type.

Theorem 6.2. *If there are at least three applicants and three institutions, then for every applicant i there exist priorities of the institutions such that any applicant-proposing menu description of DA for applicant i is non-local.*

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

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

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

$$\begin{aligned}
&\succ_1 : h_1 \succ h_2 \succ h_3 \qquad &&\succ'_1 : h_1 \succ h_3 \succ h_2\\
&\succ_2 : h_2 \succ h_1 \succ h_3 \qquad &&\succ'_2 : h_2 \succ h_3 \succ h_1
\end{aligned}$$

Consider executing D when preferences are $P = (\succ_1, \succ_2)$ (i.e. where d_1 has preference

$\succ_1$ and d_2 has preference $\succ_2$). Consider the final time that the description learns the difference between $\succ_j$ and $\succ'_j$ for some $j \in \{1, 2\}$, that is, the last state v along the execution path of P where the execution diverges from that of some $P' \in \{(\succ'_1, \succ_2), (\succ_1, \succ'_2)\}$. By the symmetry in the defined set of preferences, it is without loss of generality to assume that this state queries applicant d_1, and thus v has one successor state consistent with preferences $P = (\succ_1, \succ_2)$, and a different successor state consistent with preferences $P' = (\succ'_1, \succ_2)$.

Now, when preferences are P, applicant d_*'s menu is $\{h_3\}$. But when preferences are P', applicant d_*'s menu is $\{h_1, h_3\}$. Recalling that D is applicant-proposing, we can see that h_1 must have already been queried from both d_1 and d_2's lists in (or before) state v (for d_1, this is because both $\succ_1$ and $\succ'_1$ rank h_1 first; for d_2, this is because $\succ_2$ ranks h_1 before h_3, but $\succ'_2$ does not, and at v we already know d_2's preference is not $\succ'_2$). Thus, because D is local, the label L_{h_1} (which determines whether h_1 is or is not on the menu) must be equal in v and in all states following v. This is a contradiction, because there are some terminal states following v where h_1 is on the menu, and some where h_1 is not on the menu. Thus, no local applicant-proposing menu description of DA exists. □

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

Proof. Assume for contradiction that D is a local institution-proposing outcome description in a market with institutions h_1, h_2 and applicants d_1, d_2, d_3. We first define preferences of three applicants as follows:

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

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

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

Analogously to the previous proof, consider the last state v along the execution path

with priorities $Q = (\succ_1, \succ_2)$ where the execution diverges from that of some priority profile in $\{(\succ'_1, \succ_2), (\succ_1, \succ'_2)\}$, and without loss of generality suppose that this state v has one successor state consistent with Q and another consistent with $Q' = (\succ'_1, \succ_2)$.

Now, recalling that D is intitution-proposing, we can see that in predecessors of v, the description has queried d_1 from both institutions' priority lists. Thus, the label L_{d_1} (which determines the match of d_1) must be equal in v and all states following v. Under Q, the matching is $\{(h_1, d_2), (h_2, d_1)\}$, and under Q', the matching is $\{(h_1, d_1), (h_2, d_2)\}$. But by locality, D must assign d_1 to the same match in all states following v, which is a contradiction. Thus, no local institution-proposing outcome description of DA exists. □

Remark B.5. We now show that in finite-market DA, budget sets and menus are different sets, and moreover, neither set includes the other. For a fixed profile of preferences and priorities, denote an applicant i's budget set $B(i) = \{h | i \succeq_h \mu(i)\}$, where μ is the outcome of DA. Let $M(i) = \mathcal{M}_{\succ_{-i}}$ denote i's menu.

Now, consider the market with institutions h_1, h_2, h_3, and h_4, and applicants d_1, d_2, d_3, and d_4. Let the preferences and priorities be as follows:

$$
\begin{array}{ll}
h_1 : d_1 \succ d_2 \succ \dots & d_1 : h_1 \succ \dots \\
h_2 : d_4 \succ d_3 \succ d_2 \succ d_1 \succ \dots & d_2 : h_1 \succ h_2 \succ h_4 \succ \dots \\
h_3 : d_3 \succ \dots & d_3 : h_3 \succ \dots \\
h_4 : d_2 \succ d_4 \succ \dots & d_4 : h_4 \succ h_2 \succ \dots
\end{array}
$$

Then, one can check that DA pairs h_i to d_i for each $i = 1, \dots, 4$, and that $h_2 \in B(d_3) \setminus M(d_3)$, and also $h_2 \in M(d_1) \setminus B(d_1)$. Thus, neither the menu nor the budget set contain each other in general. Moreover, the relationship between the two sets does not seem to be restricted in a straightforward way based on priorities: despite the fact that $d_3 \succ_{h_2} d_2$, we have $h_2 \notin M(d_3)$; despite $d_1 \prec_{h_2} d_2$ at h_2, we have $h_2 \in M(d_1)$.

Remark B.6. We now show that in DA, an applicant's set of stable partners is a (possibly strict) subset of her menu. For a given profile of preferences and priorities, let $S(i)$ denote the set of stable partners of applicant i, and let $M(i)$ denote her menu. To see that $M(i) \neq S(i)$, consider any instance with two institutions h_1, h_2 which both rank i above all other applicants. If i ranks h_1 above all other institutions,

then $h_2 \in M(i) \setminus S(i)$.

We now show that $S(i) \subseteq M(i)$. Suppose the profile of preferences and priorities is P. Consider any $h \in S(i)$, and let μ be a stable matching with $\mu(i) = h$. Then, let $\widetilde{P}$ denote modifying P by having i submit a list which ranks only h. Then, observe that μ is also stable under $\widetilde{P}$. Thus, by the Rural Hospital Theorem (Theorem D.6), i and h must be matched in every stable matching under $\widetilde{P}$, in particular, in $DA(\widetilde{P})$. Thus, $h \in M(i)$, and $S(i) \subseteq M(i)$.

C Delicate Descriptions of Deferred Acceptance

In this appendix, we present additional descriptions of DA. While these are interesting on a technical level, we believe these descriptions are vastly more complicated than traditional descriptions of DA, and quite impractical. For notational convenience, in this appendix, we refer to the priorities of institutions as "preferences." We also denote the set of applicants by $\mathcal{D}$, the set of institutions by $\mathcal{H}$, and (when relevant) we describe the menu to applicant d_*.

C.1 Institution-proposing outcome description of DA

First, we construct an *institution*-proposing linear-memory outcome description of DA. Interestingly, essentially this same algorithm was used as a lemma by Ashlagi et al. (2017) (henceforth, AKL).[6]

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

Proof. AKL refer to the sides of the market as "men" and "women", and define "Algorithm 2 (MOSM to WOSM)", a men-proposing algorithm for the women-optimal stable matching. Description A.2 follows the exact same order of proposals as this algorithm from AKL. The only difference apart from rewriting the algorithm in a more

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

"pseudocode" fashion is that Description A.2 performs bookkeeping in a slightly different way—Algorithm 2 from AKL maintains *two* matchings, and their list V keeps track of only women along a rejection chain; our list V keeps track of both applicants and institutions along the rejection chain (and can thus keep track of the "difference between" the two matchings which AKL tracks).

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

C.2 Applicant-proposing menu description of DA

In this section, we construct an applicant-proposing linear-memory menu description of DA. On an intuitive level, the algorithm works as per Example 2.9, but avoiding the need to "restart many times" by using the various properties of DA and by careful bookkeeping (to intuitively "simulate all of the separate runs of the brute-force description on top of each other").

On a formal level, we describe the algorithm as a variant of Description A.2. The proof constructing this algorithm uses a bijection between one applicant's menu in DA under some preferences, and some data concerning the *institution*-optimal stable matching under a related set of preferences. Our applicant-proposing menu description is then phrased as a variation of Description A.2, which (reversing the roles of applicants and institutions from the presentation in Description A.2) is able to compute the institution-optimal matching using an applicant-proposing algorithm.

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

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

Description A.2 An institution-proposing outcome description of DA

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

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

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

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

We need to modify the preference lists of the pre-existing institutions as well. But this modification is simple: for each $h_i \in \mathcal{H}$, replace d_* with d_i^{try}. The institution-optimal matching for this augmented set of preferences P' will encode the menu, as we need.[7]

Proposition C.2. *An institution $h_i \in \mathcal{H}$ is on d_*'s menu in APDA with preferences P if and only if in the institution-optimal stable matching with the augmented preferences P', we have h_i^{try} matched to d_i^{try}.*

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

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

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

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

[7] For the reader familiar with the rotation poset of stable matchings (Gusfield and Irving, 1989b), the intuition for this construction is the following: having h_i^{try} reject applicant d_i^{try} corresponds to d_* "trying" to get $h_i \in \mathcal{H}$, i.e., "trying to see if h_i is on their menu." If d_* would be rejected by h_i after proposing, either immediately or after some "rejection rotation," then so will d_i^{try} (because they serve the same role as d_* at h_i). So if a rotation swapping h_i^{try} and h_i^{fail} exists (e.g., in the institution optimal matching) then h_i is *not* on d_*'s menu. On the other hand, if d_* could actually permanently match to h_i, then d_i^{try} proposing to h_i will result in a rejection chain that ends at some other applicant (either exhausting their preference list or proposing to an institution in $\mathcal{H}_{\text{term}}$), which does not result in finding a rotation (or writing a new set of matches as we "work towards the institution-optimal match"). Thus, if h_i^{try} and h_i^{fail} do not swap their matches in the institution-optimal stable outcome, then h_i *is* on d_*'s menu.

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

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

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

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

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

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

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

C.3 Institution-proposing individualized dictatorship description of DA

In this section, we construct an institution-linear individualized dictatorship description of (applicant-optimal) DA.[8] Throughout this section, let $P|_{d_i:L}$ denote altering preferences P by having d_i submit list L.

Unlike our applicant-proposing menu description of DA from Section C.2, our institution-proposing individualized-dictatorship description cannot be "reduced to" another algorithm such as Description A.2. However, the algorithm is indeed a modified version of Description A.2 that "embeds" our simple institution-proposing menu algorithm Description 1 (i.e., IPDA where an applicant d_* submits an empty preference list) as the "first phase." The key difficulty the algorithm must overcome is being able to "undo one of the rejections" made in the embedded run of Description 1. Namely, the algorithm must match d_* to her top choice from her menu, and "undo" all the rejections caused by d_* rejecting her choice.[9] To facilitate this, the description has d_* reject institutions that propose to d_*"as slowly as possible," and maintains a delicate $\widetilde{O}(n)$-bit data structure that allows it to undo one of d_*'s rejections.[10] The

[8]For some technical intuition on why such a description might exist, consider the construction used in Theorem 5.4, and consider an individualized dictatorship for applicant i executed on these preferences. To find the menu in this construction with an applicant-proposing algorithm, all of the "top tier rotations" must be "rotated", but to find the correct final matching after learning t_i, some arbitrary subset of the rotations must be "unrolled" (leaving only the subset of rotations which t_i actually proposes to). Theorem 5.4 shows that all of this information must thus be remembered in full. Now consider a run of Description A.2 on these preferences (or on a modified form of these preferences where institutions' preference lists determine which top tier rotations propose to bottom tier rotations). Some subset of top-tier institutions will propose to applicant i. To continue on with a run of Description A.2, it suffices to undo *exactly one* of these proposals. So, if two or more top-tier rotations trigger a bottom-tier rotation, then we can be certain that the bottom-tier rotation will be rotated, and we only have to remember which bottom-tier rotations are triggered by exactly one top-tier rotation (which takes $\widetilde{O}(n)$ bits).

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

[10]Interestingly, this "rolled back state" is *not* the result of institution-proposing DA on preferences $(P, d_i : \{h_j\})$, where h_j is d_i's favorite institution on her menu. Instead, it is a "partial state" of Description A.2 (when run on these preferences), which (informally) may perform ad-

way this data structure works is involved, but one simple feature that illustrates how and why it works is the following: *exactly one* rejection from d_* will be undone, so if some event is caused by *more than one* (independent) rejection from d_*, then this event will be caused regardless of what d_* picks from the menu.

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

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

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

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

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

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

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

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

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

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

ditional "applicant-improving rotations" on top of the result, and thus we can continue running Description A.2 until we find the applicant-optimal outcome.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Now let $\overline{Q}'$ be the revealed preferences after adding d to h's list, and let μ' and Δ' be the updated version of these values after Description A.4 processes this proposal

(formally, if NextInterestedApplicant *returns d, fix μ' and Δ' to the values of μ and Δ after the algorithm finishes running* AdjustUnrollDag*; if* NextInterestedApplicant *does not return d, set $\mu' = \mu$ and $\Delta' = \Delta$). Then (μ', Δ') are unroll-correct for Q at $\overline{Q}'$.*

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

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

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

Now suppose $d \notin \{\emptyset, d_*\}$ is returned from AdjustUnrollDAG, and d does not already appear in Δ. In this case, $h \succ_d \mu(d)$, and for every $h_* \in \mathcal{H}_{\text{menu}}$, the unrolled state when preferences $(\overline{Q}, d : \{h_*\})$ will pair d to $\mu(d)$. Under preferences $(\overline{Q}', d : \emptyset)$,

a single additional proposal will be made on top of the proposals of $(\overline{Q}, d : \emptyset)$, namely, h will propose to d and d will reject $\mu(d)$. However, *if h_* is such that h is "unrolled"* (formally, if h_* is such that UnrollOneChain(μ, Δ, h_*) changes the partner of h) then h cannot propose to d in $(\overline{Q}, d : \emptyset)$ (because all pairs in Δ can only "unroll" h to partners before $\mu(h)$ on h's list), nor in $(\overline{Q}', d : \emptyset)$ (because $\overline{Q}'$ only adds a partner to h's list after $\mu(h)$). Thus, for all h_* such that h is unrolled, the pair $(d, \mu(d))$ should be unrolled as well. On the other hand, for all h_* such that h is not unrolled, h will propose to d (matched to d'), so d will match to h in the unrolled-to state. This is exactly how μ' and Δ' specify unrolling should go, as needed.

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

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

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

$\overline{Q}'$. Formally, under preferences $(\overline{Q}', d_* : \{h_*\})$, if d_1 rejected h_1, the rejections would follow exactly as in the path in Δ between p_1 and p, and finally h would propose to d_1. Description A.2 would then call WriteRotation, and the value of μ would be updated for each d on this path. So deleting these nodes is correct in this subcase.[11]

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

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

Second, suppose h_* is such that (d_*, h_*) is an ancestor of some node in P_1 in Δ. There are two subcases. If $h_{\min} = h$, then we have $\mu(d_1) = \mu'(d_1)$, but when UnrollOneChain(μ', Δ', h_*) is run, we unroll d_1 to h. Correspondingly, in IPDA with preferences $(\overline{Q}', d_* : \{h_*\})$, we know d_1 will not receive a proposal from $\mu(d_1)$ (as this match is unrolled in $\overline{Q}$) but d_1 will receive a proposal from h (as this additional proposal happens in $\overline{Q}'$ but not in $\overline{Q}$, regardless of whether this happens due to a "re-

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

jection rotation" of AKL, or simply due to two rejection chains causing this proposal, as discussed above), which d_1 prefers to the unrolled-to match under preferences $\overline{Q}$. Thus, under preferences $(\overline{Q}', d_* : \{h_*\})$, we know d_1 will match to $h_{\min} = h$ in a valid partial AKL-state. So (μ', Δ') is correct for $\overline{Q}'$ in this subcase. If, on the other hand, $h_{\min} = \mu(d_1)$, then in Δ', UnrollOneChain(μ', Δ', h_*) will not contain the new node p_{new}. However, $\mu'(d_1) = h$, and we know d would receive a proposal from h $(\overline{Q}', d_* : \{h_*\})$, and would accept this proposal. So (μ', Δ') is correct for $\overline{Q}'$ in this subcase.

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

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

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

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

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

We can now prove our main result:

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

Proof. We know Description A.4 correctly computes the menu, and that it is institution-proposing and $\widetilde{O}(n)$ memory. So we just need to show that it correctly computes the

final matching. To do this, it suffices to show that at the end of Phase 1 of Description A.4, (μ, Δ) is unroll-correct for Q at the truncated revealed preferences $\overline{Q}$ (for then, by definition, running Description A.2 after UnrollOneChain will correctly compute the final matching).

To see this, first note that an empty graph is unroll-correct for the truncated revealed preference after running $IPDA(P_{\text{hold}})$, as no further proposals beyond d_* can be made in these truncated preferences. Second, each time we pick an $h \in \mathcal{H}_*$ on Description 8, a single (d_*, h) added to Δ (with no edges) is unroll-correct for Q at $\overline{Q}'$, by construction. Finally, by Lemma C.8, every other query to any institution's preference list keeps (μ, Δ) unroll-correct after the new query. So by induction, (μ, Δ) is unroll-correct at the end of Phase 1, as desired. □

D Proofs of Known Results

This appendix is dedicated to reproducing complete, from scratch proofs of all lemmas we need to reason about DA and TTC. The results in Section D.1 for DA, along with our direct proof of the correctness of our menu description of DA in Appendix B, demonstrate the strategyproofness of (the traditional description of) DA from first-principles. If desired, one can then compare this proof with one classical, direct proof presented in Section D.2. In Section D.3, we present a classical lemma for TTC: that the result is independent of the order in which cycles are eliminated (in the traditional description).

D.1 Direct Proof of Stable Matching Lemmas

Here, we supply all lemmas needed for the direct proof of Theorem 3.2 given in Appendix B. Let D denote the set of applicants, and H the set of institutions. Recall that a matching μ is *stable* if $\mu(a) \succ_a \emptyset$ for all $a \in D \cup H$, and moreover there is no pair $d \in D, h \in H$ such that $h \succ_d \mu(h)$ and $d \succ_h \mu(d)$.

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

Proof. Consider running the traditional description of DA on some profile of preferences (and priorities), and let the output matching be μ. Consider a pair $d \in D$, $h \in H$ which is unmatched in μ. Suppose for contradiction $h \succ_d \mu(d)$ and $d \succ_h \mu(h)$.

In the DA algorithm, d would propose to h before $\mu(d)$. However, it's easy to observe from the traditional description of DA that once an institution is proposed to, they remain matched and can only increase their priority for their match. This contradicts the fact that h was eventually matched to $\mu(h)$. □

Note that Lemma D.1 also proves that at least one stable matching always exists. Next, we show that DA (i.e., the matching output by the APDA algorithm) is (simultaniously) the best stable matching for all applicants.

Lemma D.2 (Gale and Shapley, 1962)**.** *If an applicant $d \in D$ is ever rejected by an institution $h \in H$ during some run of the APDA algorithm, 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 APDA where such a rejection occurred. In particular, let h reject $d \stackrel{\text{def}}{=} \mu(h)$ in favor of $\widetilde{d} \neq d$ (either because $\widetilde{d}$ proposed to h, or because $\widetilde{d}$ was already matched to h and d proposed). We have $\widetilde{d} \succ_h d$. We have $\mu(\widetilde{d}) \neq h$, simply because μ is a matching. Because this is the *first* time an applicant has been rejected by her match in μ, $\widetilde{d}$ has not yet proposed to $\mu(\widetilde{d})$. This means $h \succ_{\widetilde{d}} \mu(\widetilde{d})$, and μ is not stable.

Thus, no institution can ever reject a stable partner in APDA. □

The following corollaries are immediate:

Corollary D.3 (Gale and Shapley, 1962)**.** *In DA, every applicant is matched to her favorite stable partner.*

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

A phenomenon dual to Corollary D.3 occurs for the institutions:

Lemma D.5 (McVitie and Wilson, 1971)**.** *In the match returned by APDA, every $h \in H$ is paired to her least-favorite stable partner.*

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, we must in fact have $\mu(h) \succ_h d$. □

Finally, we show that the set of matched agents must be the same in each stable matching.

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

Proof. Consider any stable matching μ in which applicants D^μ and institutions H^μ are matched, and let D^0 and H^0 be matched in DA. By Corollary D.3, we know that for all $d \in D^\mu$, the match of d can only improve in DA; in particular, d is still matched in DA, and thus $D^\mu \subseteq D^0$. Similarly, Lemma D.5 implies that each agent in H^0 is matched in every stable outcome, so $H^0 \subseteq H^\mu$. But then, since the matching is one-to-one, 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 and in DA. Thus, $D^0 = D^\mu$ and $H^0 = H^\mu$. □

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

To contrast between Section D.1 and Theorem 3.2, we also include a direct proof of the strategyproofness of DA, adapted from Gale and Sotomayor (1985). Note, however, that the following proof also shows that DA is *weakly group* strategyproof, whereas Theorem 3.2 does not.

Lemma D.7 (Attributed to J.S. Hwang by Gale and Sotomayor, 1985)**.** *Let $\mu = APDA(P)$ and μ' be any other matching. Let T denote the set of all applicants who* strictly *prefer their match in μ' to their match in μ, and suppose $T \neq \emptyset$. Then there exists a blocking pair (d, h) in μ'* with *$d \notin T$.*

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

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

Case 2: $\mu(T) = \mu'(T)$. This case is a bit harder. Consider the run of $APDA(P)$. First, for any $d \in T$, note that d must have proposed to $\mu'(d)$ before proposing to

$\mu(D)$, so each institution in $\mu(T)$ receives at least two proposals from applicants in T.

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

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

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

Theorem D.8 (Roth, 1982; Dubins and Freedman, 1981)**.** *APDA is (weakly group-)strategyproof for the applicants.*

Proof. Suppose a set L of applicants change their preferences, and each of them improve their match. In particular, if $\mu' = APDA(P')$, where P' is the altered list of preferences, then $L \subseteq T$ as in Lemma D.7. 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. □

D.3 Direct Proof that TTC is Independent of the Order Of Cycle Elimination

We now prove that TTC is independent of the order in which the steps are chosen in the traditional description (analagous to Corollary D.4 for DA). Intuitively, this follows mostly from the observation that cycles in the traditional description of TTC must always be disjoint (since the pointing graph has out-degree 1), so choosing different orderings to eliminate cycles is actually immaterial; we formalize this idea

below. See also Carroll (2014); Morrill and Roth (2024) for similar contemporary proofs.

Lemma D.9 (Follows from Shapley and Scarf, 1974; Roth and Postlewaite, 1977)**.** *The TTC algorithm is independent of the order in which cycles are chosen and eliminated.*

Proof. Fix a profile of priorities and preferences. Define the *elimination graph* G as follows. The vertices of G are the set of all partial matchings between applicants and institutions. There is an edge $\mu_1 \to \mu_2$ in G whenever μ_2 differs from μ_1 by the elimination of exactly one cycle, as defined in Definition 2.3, under the given preferences and priorities. Formally, this is defined as follows. Fix μ_1, and consider the *pointing graph* $B = B_{\mu_1}$ given μ_1 to be the bipartite graph formed by applicants and institutions who are unmatched in μ_1, where each agent points to her top-ranked agent on the other side who is unmatched in μ_1 (if any such agents on the other side remain). Then, we have an edge $\mu_1 \to \mu_2$ whenever there exists a cycle in B such that, if μ_1 is modified such that every applicant in the cycle is matched to the institution she points to, then the resulting matching is μ_2. When $\mu_1 \to \mu_2$ in G, and the cycle C in B_{μ_1} represents the difference between μ_2 and μ_1, we say that C is *available* in μ_1.

Now, define a *elimination sequence* T to be any sequence $T = \mu_1 \to \mu_2 \to \ldots \to \mu_k$ of adjacent edges in G, such that μ_1 is the empty matching which pairs no agents, and T is of maximal possible length. Observe that the outcome of TTC is defined to be the final matching μ_k of an elimination sequence.

We make the following observations regarding any elimination sequence $T = \mu_1 \to \ldots \to \mu_k$:

- For any fixed pointing graph B_{μ_i}, all of the cycles C in B_{μ_i} are disjoint. This follows because the pointing graph has out-degree 1.

- If C is available in some μ_x, then there exists a $z > x$ such that C is available in every subsequent μ_y for $x \leq y < z$. This follows from the previous observation, since for each $\mu_y \to \mu_{y+1}$ with $x \leq y < z$ with y increasing inductively, the vertices in the cycle C are not changed as we switch from μ_y to μ_{y+1}, unless the cycle C itself is eliminated. Thus, in particular, μ_z differs from μ_{z-1} by the elimination of C.

- Suppose that in T, cycle C_1 is available in some μ_x, but $C_2 \neq C_1$ eliminated in μ_x to get μ_{x+1}. Then, there exists another elimination sequence $T' = \mu_1 \to \mu_x \to \mu'_{x+1} \to \ldots \to \mu'_k$ which agrees with T up until μ_x, but C_1 is eliminated at μ_x to get μ'_{x+1}, and which ends in the same final matching $\mu'_k = \mu_k$. To show this, we construct T' as follows. After eliminating C_1 at μ_x to get μ'_{x+1}, follow the same order of eliminating cycles as in T until cycle C_1 is eliminated in T—i.e., go from μ'_{y+1} to μ'_{y+2} via the same cycle used to go from μ_y to μ_{y+1}, for each $y \geq x$ such that C_1 is not eliminated in $\mu_y \to \mu_{y+1}$ in T. (All such cycles must be available as needed in T', since before C_1 was eliminated in T, none of these cycles could have involved agents in C_1 in any way.) At some point, C_1 must be eliminated in T, say in $\mu_z \to \mu_{z+1}$. After this point, the elimination sequence T' will from that point onward agree with T, i.e., $\mu_w = \mu'_w$ for $w \geq z + 1$.

Now, suppose for contradiction that there are two elimination orderings T_1 and T_2 which produce different final matchings, and additionally suppose among all such pairs, the index $j > 1$ where T_1 and T_2 first disagree is *as large as possible*. Then, at index j, two cycles C_1 and C_2 are eliminated in T_1 and T_2, respectively. Then, by final observation listed above, we can consider the elimination sequence T'_2 that disagrees with T_1 at least one step later than j (by eliminated C_1), but has the same final matching as T_2. This contradicts the assumption that j was as large as possible.

This proves that all elimination sequences must produce the same final matching, which is the outcome of TTC. This proves the result. □

References for Appendices

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

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

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

G. Carroll. A general equivalence theorem for allocation of indivisible objects. *Journal of Mathematical Economics*, 51:163–177, 2014.

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

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

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

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

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

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

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

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

T. Morrill and A. E. Roth. Top trading cycles. *Journal of Mathematical Economics*, 112: 102984, 2024.

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

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

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

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

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