A General Equivalence Theorem for Allocation of Indivisible Objects

Gabriel Carroll, Stanford University

gdc@stanford.edu

December 13, 2013

Abstract

We consider situations in which n indivisible objects are to be allocated to n agents. A number of recent papers studying such allocation problems have shown various interesting equivalences between randomized mechanisms based on trading and randomized mechanisms based on serial dictatorship. We prove a very general equivalence theorem from which many previous equivalence results immediately follow, and we give several new applications. Our general result sheds some light on why these equivalences hold by presenting the existing serial-dictatorship-based mechanisms as randomizations of a general mechanism which we call *serial dictatorship in groups*. Our proof technique also streamlines the bijective methods used in previous proofs, showing that it is enough to assemble a bijection from smaller pieces, without needing to construct the pieces explicitly.

Keywords: Equivalence; indivisible goods; random assignment; random serial dictatorship; serial dictatorship in groups; top trading cycles

JEL Classifications: C78, D79

Thanks to (in random order) Özgün Ekici, Jay Sethuraman, Daron Acemoglu, Parag Pathak, Glenn Ellison, and Tayfun Sönmez for discussions and advice. One anonymous referee also offered exceptionally helpful suggestions. Most of the research for this paper was performed while the author was a student at MIT and was supported by an NSF Graduate Research Fellowship.

1 Introduction

We consider situations in which there are n agents and n indivisible objects to be allocated to them. Each agent is to receive one of the objects, and each agent has strict preferences over the objects. Monetary transfers are not possible. The task is to allocate the objects based on the agents' preferences. This model has numerous real-life applications; the objects may, for example, be housing units, jobs, seats at public schools, time slots for the use of a machine, or organs for transplant [3, 4, 6, 23].

Many of the allocation mechanisms often proposed for such situations are versions of one of two basic mechanisms: one (*serial dictatorship*) in which agents are lined up and choose objects one by one, and another (*top trading cycles*, or TTC for short) in which the agents are initially endowed with objects and then trade them. In either case, fairness can be introduced into the mechanism by some randomization over the agents. In recent years, a number of results have appeared showing unexpected equivalences between randomized versions of serial dictatorship on one hand, and randomized versions of TTC on the other. The appearance of several similar results, and often with similar styles of proof, suggests that there should be a single master equivalence theorem incorporating all of them as special cases. The main mission of the present paper is to give such a master theorem.

To introduce this contribution in more detail, we first expand on the two basic mechanisms. The first, serial dictatorship, applies naturally when there is no initial ownership of the objects, and works as follows. The agents are arranged in some order. The first agent gets his favorite object; the second agent then gets his favorite object from the ones remaining; the third agent then gets his favorite object from the ones remaining, and so forth.

The second mechanism, TTC (first described in [24]), applies when each object is initially owned by a different agent. The model then becomes a special case of an exchange economy. In this setting, TTC allocates the objects by the following algorithm: Initially, each agent points to the owner of his favorite object. In the resulting graph, there is at least one cycle, i.e. a sequence of agents $i^1, \ldots, i^k$ such that i^1 points to i^2, i^2 points to i^3, …, i^k points to i^1. (An agent pointing to himself forms a cycle of length 1.) For each such cycle, give each agent the object owned by the agent he points to, and remove these agents and objects. Next, among the remaining agents, have each point to the owner of his favorite among the remaining objects. Again apply and remove any cycles. Continue in this way until every agent has been allocated an object.

A number of characteristics help explain the popularity of these two mechanisms: Both of them lead to Pareto-efficient allocations [2]. In the second case, TTC computes the allocation in the core of the exchange economy (which turns out to be unique) [22]. Moreover, both mechanisms are strategyproof, i.e. for each agent, it is a dominant strategy to report his preferences truthfully. (See [21] for TTC.) Serial dictatorship is intuitive and easy to execute in practice; and TTC is actually the unique mechanism for exchange economies that is Pareto-efficient, strategyproof, and individually rational [13].

The connection between randomized versions of the two mechanisms was pointed out by Abdulkadiroğlu and Sönmez [2], who considered an allocation problem with no initial ownership. They considered on the one hand *random serial dictatorship* (RSD), given by ordering the agents uniformly at random and then executing serial dictatorship; and on the other hand *core from random endowments*, given by initially allocating each object to a different agent uniformly at random and then executing TTC. They showed that these two mechanisms produce the same probability distribution over allocations.[1] They presented this result as an extra justification for the use of RSD in real-world settings. Subsequently, Pathak and Sethuraman [19] and Sönmez and Ünver [25] gave several other equivalence theorems in the same vein. We will describe their results more detail later, in Subsections 2.1 and 2.2. Aside from their mathematical interest, these equivalences also help shed light on the practical problem of choosing an allocation mechanism, since they lead to new interpretations of certain mechanisms.

To formulate our result, we will consider here a version of TTC that is more general than the one sketched above, drawing on [19] among others [4, 15]. Instead of having a single owner, each object has a *priority list* that gives an ordering of the n agents. Different objects may have different priority lists. Each agent points to his favorite object, and each object points to the top agent in its priority list. At least one cycle forms. In each cycle, every agent is assigned the object he points to, and these agents and objects are removed. Next, each remaining agent points to his favorite of the remaining objects, and each remaining object points to the highest-priority of the remaining agents; any cycles are again applied and removed, and so forth until every agent has received an object.

We will express our results using the concept of *priority frameworks*, which make such TTC mechanisms anonymous by instead giving each object an ordered list of n abstract *priority roles*. Given a priority framework, if we choose an assignment of the agents to the roles, we get a priority list for each object and can then apply TTC as above.

For expository ease, our unifying theorem is presented in two steps. We first give

[1]This result was also discovered by Knuth [10].

a special case (Theorem 1) and then the full result (Theorem 3). Theorem 1 says that for any fixed priority framework, if we randomly assign the agents to the roles and then run TTC, the resulting distribution is the same as RSD. Theorem 3 then extends this by assuming that the set of agents is partitioned into groups, and the priority roles are partitioned into groups of corresponding sizes. The agents are randomly assigned to roles within each group, and then TTC is run. Theorem 3 states that this mechanism is equivalent to a randomization of a new mechanism we call *serial dictatorship in groups*, a kind of generalization of serial dictatorship where several dictators from different groups choose simultaneously.

Our paper makes substantive, conceptual, and methodological contributions to the indivisible goods literature. On the substantive front, we unify various previous equivalence results by showing that they are all special cases of Theorem 3, and also illustrate the gain in generality with a few new applications.[2]

On the conceptual front, we introduce the ideas of anonymous priority frameworks and priority roles. We also introduce the mechanism of serial dictatorship in groups (SDIG), which is properly a special case of TTC, but can also be thought of as intermediate between serial dictatorship and TTC. Bridging this gap helps build insight into why the existing equivalence results hold: each such equivalence compares two mechanisms in which, at each step, some collection of agents get to choose their favorite objects; the mechanisms superficially differ in the order in which agents choose, and our proof shows that these differences have no effect on the allocations.

On the methodological front, the proof of the main equivalence result, Theorem 3, exhibits a new technique that draws on the bijective method used in most previous proofs [2, 18, 25, 8]. We show that each way of assigning the agents to roles for TTC can be mapped to a different ordering of the agents for SDIG, such that TTC leads to the same allocation as the corresponding SDIG. But unlike previous bijective proofs, we do not fully construct this mapping. Instead, we show how to build it up from smaller bijections, each tying a set of several role assignment functions to a set of several orderings that all produce the same allocation. Rather than construct each smaller bijection explicitly, we simply show that the two sets involved have the same size, and so the bijection between them can be chosen arbitrarily. This approach avoids some of the messy details of previous arguments. On the other hand, the full proof of Theorem 3 is not short: with the

[2]As of this writing (December 2013), there is only one previously-published result in this literature that is *not* a special case of our main theorem, namely an equivalence between single and multiple lotteries for house allocation with existing tenants, mentioned in [19]. However, there is also more recent work [12] that is still more general — discussed ahead.

new generality that comes from partitioning agents, the process of verifying that the construction works becomes fairly involved.

After the present paper was first circulated, Lee and Sethuraman [12] gave a general technique for proving equivalence of random allocation mechanisms. Their technique, extending ideas from Pathak and Sethuraman [19], involves an inductive argument, using inclusion-exclusion to express the outcomes of two mechanisms in terms of outcomes for smaller sets of agents and objects, then showing that the terms that differ between the two mechanisms cancel out. This allows them to give a quick proof of a new equivalence that further generalizes our Theorem 3, and also of a related result in a working paper by Ekici [8] that does not follow from ours. Although their proofs are shorter than ours, the bijective approach here arguably offers more understanding as to why the equivalence holds.

In the next section, we present the model and the necessary notation in detail. We first define priority roles and priority frameworks, which allows us to state Theorem 1. Then, we describe how to extend these concepts to partitions of agents into groups, leading to Theorem 3. In the process, we describe the previous equivalence results and show how they are special cases, and also give new applications. The following section gives the proof: first a quick sketch of the ideas in the special case represented by Theorem 1, then the full details for the more general Theorem 3. After this, we give some examples of the necessity of a technical condition in Theorem 3. Finally, we give a brief conclusion.

2 The results

2.1 Full anonymity

We first develop the machinery necessary to state our basic theorem, Theorem 1, and discuss the existing results that are special cases, as well as some new applications. Again, the proof is ahead, in Section 3.

We consider sets of n *agents*, $I = \{i_1, \ldots, i_n\}$, and n *objects*, $O = \{o_1, \ldots, o_n\}$. Each agent i has a strict preference ordering $\succ_i$ over the objects; $o \succ_i o'$ means that i prefers object o over o'. Preferences will be held fixed. An *allocation* is an assignment of one object to each agent, with different agents getting different objects.

Ordinarily, a *mechanism* is a function mapping profiles of preferences for the n agents to allocations, or probability distributions over allocations. However, our purpose here is to prove that two (random) mechanisms are equivalent — that is, they lead to the same distribution over allocations for any given preference profile. We are not interested in any properties of the mechanism that require comparing different preference profiles (e.g. strategyproofness), except in passing, as motivation for focusing attention on particular mechanisms. For our purposes, then, it is more convenient to hold preferences fixed, and we define a *mechanism* to be a (possibly randomized) algorithm which produces an allocation as output. Two mechanisms are *equivalent* if they lead to the same probability distribution over allocations. Since our equivalence results will hold for arbitrary preferences, they can also be interpreted as equivalences of mechanisms under the more usual definition.

One commonly used mechanism is as follows. Given an ordering of the agents — that is, a bijection $f : \{1, \ldots, n\} \to I$ — we can do the following:

- At step 1, agent $f(1)$ is assigned his favorite object. This agent and object are removed.
- At each step t, for $t > 1$, agent $f(t)$ is assigned his favorite object from the ones currently remaining. This agent and object are removed.

After n steps, all agents and objects have been removed; each agent has been assigned a distinct object, so we have an allocation. This mechanism is the *serial dictatorship* corresponding to the ordering f.

The *random serial dictatorship* (RSD) is the mechanism given as follows: first choose an ordering f uniformly at random from the $n!$ possible orderings, then apply the serial dictatorship corresponding to f.

The other basic mechanism of interest to us is *top trading cycles* (TTC). We use the priority-lists form of TTC as described by Abdulkadiroğlu and Sönmez [4] (see also [15]). Suppose that each object has associated with it a *priority list*, an ordering of the agents. A specification of n priority lists, one for each object, will be called a *priority structure.* (The terminology in previous literature varies; see e.g. [15, 7, 26, 18, 19].) Given a priority structure π, the following is the *top trading cycles* mechanism corresponding to π:

- At step 1, a directed graph is formed having the objects and agents as vertices, where each vertex has one outgoing edge. Specifically, each agent points to his favorite object, and each object points to the agent at the top of its priority list. Any cycle in this graph must alternate between agents and objects. For each such cycle, we assign each agent the object to which he points. (We can do this, because all cycles are mutually disjoint.) All the agents and objects involved in any cycle are removed.

- At step t, for $t > 1$, we again form a directed graph on the remaining agents and objects. Each agent points to his favorite among the remaining objects, and each object points to whichever of the remaining agents is highest on its priority list. Again, for any cycle, we assign each agent the object to which he points. All the agents and objects involved in any cycle are removed.

At each step, at least one cycle must form, since we have a finite graph in which each vertex has outdegree 1. Thus the algorithm must terminate in at most n steps.

If f is an ordering of the agents, and every object has priority list $f(1), \dots, f(n)$, then the resulting TTC mechanism is equivalent to the serial dictatorship for f. Indeed, at each step t, all objects point to $f(t)$, and so the only cycle consists simply of agent $f(t)$ and his favorite object among those available.

If each object has a different agent at the top of its priority list, then our TTC mechanism coincides with the simpler TTC mechanism for an exchange economy, described in the introduction, where each object is initially owned by the agent at the top of its list. (The rest of each priority list is irrelevant.) Indeed, the cycles formed at each step are the same as in that algorithm, except that instead of agent i pointing to the owner of his favorite object among those available, i points to his favorite object, which in turn points to its owner. To check this, we just need to verify that at each step t each remaining object is pointing to its owner. This holds by induction on t: If it fails, some remaining o must have had its owner i removed at time $t-1$, but only o was pointing to i at $t-1$ (by the induction hypothesis) so they should have been removed together, a contradiction.

As discussed in the introduction, both serial dictatorship and TTC for endowment economies are very natural mechanisms for a variety of reasons. The normative case for the more general version of TTC with priority lists is perhaps less clear-cut; but if one grants that it is natural in some applications for objects to have exogenous priority lists, then TTC can be thought of as a way to respect priorities for each object while allowing agents to achieve Pareto efficiency by "trading" their priorities [4]. Abdulkadiroğlu and Che [1] give an axiomatization of TTC with priority lists that formalizes this idea; see also [14, 26, 17] for related axiomatization results.

We have written out the TTC algorithm so that all cycles are removed at each step. However, the same allocation would result if we removed only a subset of cycles at each step — regardless of the order in which the cycles were removed. Previous literature has recognized that this "slow TTC" produces the same allocation as TTC (see [3, Remark 1]). We will take this fact for granted in what follows; but for completeness, we provide a formal statement and its (routine) proof in the appendix, as Lemma 6.

Next, we introduce the formal vocabulary needed to make priority structures anonymous. Define a set of n *priority roles* $R = \{r_1, \ldots, r_n\}$. A *priority role list* is an ordering of the priority roles, and a *priority framework* is a specification of n priority role lists, one for each object. A *role assignment function* g is a bijection from the priority roles to the agents. Suppose we are given a priority framework ϕ and a role assignment function g. We can then construct a priority structure from ϕ by replacing each role r_j by $g(r_j)$ in every object's priority role list; denote this structure by $g(\phi)$.

Thus, a priority framework describes the logical relationships between different objects' priority lists without identifying individual agents. For example, if ϕ is the priority framework in which every object has list $r_1, \ldots, r_n$, then $g(\phi)$ is the priority structure in which every object has list $g(r_1), \ldots, g(r_n)$. TTC with this priority structure is then equivalent to serial dictatorship with the ordering $f(j) = g(r_j)$.

For any priority framework ϕ, we can define the mechanism *random TTC from* ϕ as follows: pick a role assignment function g, uniformly at random from the set of all $n!$ possible such functions; then execute TTC with priority structure $g(\phi)$. Intuitively, this mechanism respects the priority relationships specified by ϕ while treating the agents entirely symmetrically.

At this point we provide a concrete example to illustrate the concepts presented thus far. Suppose $n = 3$, and the preferences and the priority role structure ϕ are as follows:

$$\begin{array}{llcll}
i_1: & o_1 \succ o_2 \succ o_3 & \qquad & o_1: & r_3, r_1, r_2 \\
i_2: & o_3 \succ o_2 \succ o_1 & & o_2: & r_3, r_2, r_1 \\
i_3: & o_3 \succ o_1 \succ o_2 & & o_3: & r_1, r_3, r_2.
\end{array}$$

Let us calculate the result of applying random TTC from ϕ. First consider the role assignment function given by $g(r_j) = i_j$ for each $j = 1, 2, 3$. This gives us the following priority structure $g(\phi)$:

$$\begin{array}{ll}
o_1: & i_3, i_1, i_2 \\
o_2: & i_3, i_2, i_1 \\
o_3: & i_1, i_3, i_2.
\end{array}$$

At the first step of the TTC algorithm, we have the cycle (i_1, o_1, i_3, o_3); whereas i_2 points to o_3 and o_2 points to i_3. So i_1 is assigned o_1 and i_3 is assigned o_3, and they are removed. At the second step, only i_2 and o_2 remain, and they point to each other. Thus, the allocation produced by TTC from $g(\phi)$ is (o_1, o_2, o_3) (this should be read to mean that agents i_1, i_2, i_3 receive o_1, o_2, o_3 respectively).

For each choice of g, we calculate in this way the priority structure $g(\phi)$ and the resulting allocation. Writing g as the triple $(g(r_1), g(r_2), g(r_3))$, we have

$$\begin{array}{ccccccccccc}
 & & i_3, i_1, i_2 & & & \qquad & & & i_1, i_2, i_3 & & \\
(i_1, i_2, i_3) & \to & i_3, i_2, i_1 & \to & (o_1, o_2, o_3) & \qquad & (i_2, i_3, i_1) & \to & i_1, i_3, i_2 & \to & (o_1, o_3, o_2) \\
 & & i_1, i_3, i_2 & & & \qquad & & & i_2, i_1, i_3 & & \\
 & & & & & & & & & & \\
 & & i_2, i_1, i_3 & & & \qquad & & & i_2, i_3, i_1 & & \\
(i_1, i_3, i_2) & \to & i_2, i_3, i_1 & \to & (o_1, o_3, o_2) & \qquad & (i_3, i_1, i_2) & \to & i_2, i_1, i_3 & \to & (o_1, o_2, o_3) \\
 & & i_1, i_2, i_3 & & & \qquad & & & i_3, i_2, i_1 & & \\
 & & & & & & & & & & \\
 & & i_3, i_2, i_1 & & & \qquad & & & i_1, i_3, i_2 & & \\
(i_2, i_1, i_3) & \to & i_3, i_1, i_2 & \to & (o_2, o_3, o_1) & \qquad & (i_3, i_2, i_1) & \to & i_1, i_2, i_3 & \to & (o_1, o_2, o_3) \\
 & & i_2, i_3, i_1 & & & \qquad & & & i_3, i_1, i_2 & &
\end{array}$$

Thus, random TTC from ϕ produces the allocations (o_1, o_2, o_3), (o_1, o_3, o_2), and (o_2, o_3, o_1) with probabilities $1/2, 1/3, 1/6$, respectively.

We can also easily calculate the outcome of random serial dictatorship. For each of the six possible orderings of the agents, we compute the allocation produced by serial dictatorship:

$$\begin{array}{ccccccc}
(i_1, i_2, i_3) & \to & (o_1, o_3, o_2) & \qquad & (i_2, i_3, i_1) & \to & (o_2, o_3, o_1) \\
(i_1, i_3, i_2) & \to & (o_1, o_2, o_3) & \qquad & (i_3, i_1, i_2) & \to & (o_1, o_2, o_3) \\
(i_2, i_1, i_3) & \to & (o_1, o_3, o_2) & \qquad & (i_3, i_2, i_1) & \to & (o_1, o_2, o_3)
\end{array}$$

Thus random serial dictatorship also gives (o_1, o_2, o_3), (o_1, o_3, o_2), (o_2, o_3, o_1) with probabilities $1/2, 1/3, 1/6$ — the same as random TTC.

Our first main result says that this identity is completely general.

Theorem 1 *Let ϕ be any priority framework. Then random TTC from ϕ is equivalent to random serial dictatorship.*

Now, in order to illustrate how previous equivalence results follow as special cases — and to offer new applications — we first give a simple corollary. Say that two priority structures π, π' are *equivalent* if there exists a permutation τ of the agents such that π' is obtained from π by replacing each agent i by $\tau(i)$ in every priority list. Clearly this is an equivalence relation. Moreover, for any given priority framework ϕ, the set of priority structures $g(\phi)$, as g varies over role assignment functions, exactly forms an equivalence class; let us say that ϕ *represents* this class.

Corollary 2 *Let Π be a nonempty set of priority structures such that, if $\pi \in \Pi$, every priority structure equivalent to π is also in Π. Consider the following mechanism: choose $\pi \in \Pi$ uniformly at random, and execute TTC with priority structure π. This mechanism is equivalent to random serial dictatorship.*

Proof: Partition Π into equivalence classes. Each equivalence class contains $n!$ priority structures. For each equivalence class Σ, fix a priority framework ϕ_Σ that represents it. Then, choosing a structure $\pi \in \Pi$ uniformly at random is equivalent to choosing Σ uniformly at random, choosing a role assignment function g uniformly at random, and taking $\pi = g(\phi_\Sigma)$.

Conditional on the choice of Σ, choosing g at random and applying TTC with priorities $\pi = g(\phi_\Sigma)$ produces the same distribution over allocations as RSD, by Theorem 1. Since this is true for every choice of Σ, the unconditional distribution over allocations is also the same as RSD. $\square$

We now describe the applications, old and new.

Core from random endowments. Consider any priority framework ϕ in which each object o_j has r_j first in its priority role list. Assigning each role r_j to a different agent $g(r_j)$ and then running TTC gives us the core of the economy where each object o_j is initially owned by $g(r_j)$. Therefore, random TTC from ϕ is just the core from random endowments. Theorem 1 tells us this mechanism is equivalent to RSD; thus we recover the result of Abdulkadiroğlu and Sönmez [2].

Partitioned random priority. Pathak and Sethuraman [19] proved an equivalence that addressed a practical design question in the secondary round of the New York City high school match: should priorities be determined by a single lottery at the citywide level, or a separate lottery for each school? The formal statement of their result concerns the *partitioned random priority* mechanism, defined in terms of an exogenously given partition of the objects into disjoint classes $C_1, \ldots, C_l$ (so $|C_1| + \cdots + |C_l| = n$). In partitioned random priority, a priority structure π is drawn uniformly at random subject to the constraint that objects in the same class have identical priority lists, and then TTC is applied. The result states that this mechanism is equivalent to RSD. That is, single-lottery and multiple-lottery systems yield the same outcome.

To see how this equivalence arises as a special case of our result, just notice that the set of priority structures π satisfying the constraint meets the conditions of Corollary 2.

Partitioned random endowment. Pathak and Sethuraman also considered *partitioned random endowment*, which can be described as follows. We run TTC with a priority structure π drawn uniformly at random, subject to the constraint that objects in the same class have identical priority lists, and the following additional constraint: if we consider the $|C_1|$ highest-priority agents for any object in C_1, the $|C_2|$ highest-priority agents for any object in C_2, …, the $|C_l|$ highest-priority agents for any object in C_l, then these n agents are equal to $i_1, \ldots, i_n$ in some order. (Thus, the mechanism is a kind of generalization of core from random endowments, where each class of objects initially is jointly "owned" by a correspondingly-sized set of agents, and each agent is an owner for exactly one class.) Pathak and Sethuraman showed that partitioned random endowment is also equivalent to RSD.

Again, the set of possible π satisfies the conditions of Corollary 2, so this equivalence result is obtained as another special case.

More general partitioned lotteries. For a new application, we can generalize Pathak and Sethuraman's two mechanisms as follows. Assume again that the objects are partitioned into classes $C_1, \ldots, C_l$, and let $b_1, \ldots, b_l$ be given nonnegative integers such that $b_1 + \cdots + b_l \leq n$. Draw a priority structure π uniformly at random, subject to the constraints that

- any two objects in the same class have the same priority list, and
- if agent i is one of the b_j highest-priority agents for class C_j, and agent i' is one of the $b_{j'}$ highest-priority agents for class $C_{j'}$, where $j \neq j'$, then $i \neq i'$.

Then execute TTC. Corollary 2 tells us that this mechanism is always equivalent to random serial dictatorship. Partitioned random priority is the special case $b_j = 0$ for all j; partitioned random endowment has $b_j = |C_j|$ for all j.

Inheritance relations. One more application is inspired by the work of Ekici [8], in which inheritance relations between agents are randomized (this work is discussed further in the conclusion). Suppose we assign each object an initial owner, uniformly at random; multiple objects may have the same owner. We also choose for each agent i an *heir* $h(i) \in I$. Heirs are chosen uniformly at random subject to the constraint that the sequence $i, h(i), h(h(i)), \ldots$ should consist of one cycle containing all n agents. Then run TTC, with each object pointing to its owner at each stage. When the initial owner i of object o is removed, o becomes owned by $h(i)$; if $h(i)$ is also removed, o is owned by $h(h(i))$, and so forth. (The one-long-cycle constraint ensures that inheritance is always well-defined — an object never gets stuck cycling among agents who have all been removed.)

This random mechanism is again equivalent to RSD. Indeed, running TTC as described is equivalent to using the following priority lists: for each object o, its list consists of its initial owner i, followed by $h(i)$, $h(h(i))$, and so forth. In view of the uniform randomizations, Corollary 2 applies.

In summary, Theorem 1 and its corollary apply to many TTC-based mechanisms one might imagine in which all agents are to be treated equitably.

There are, however, situations where it is natural to make some distinctions among agents — most prominently, the model known as "house allocation with existing tenants," where some agents initially own an object and others do not [3, 25]. This motivates us to formulate the more general equivalence result of the next subsection.

2.2 Anonymity within groups

Suppose henceforth that we are given an exogenous partition of the set of agents into *groups* $I_1, \ldots, I_m$. We would like to write down an equivalence result that looks like Theorem 1, but instead of randomly permuting all the agents, the serial-dictatorship-based mechanism only permutes the agents within each group separately.

Suppose that we are given, for each group, an ordering of the agents within that group; call these orderings $f_1, \ldots, f_m$. Suppose we are also given a priority structure π. Consider the following mechanism:

- At each step $t \geq 1$, for each group whose agents have not all been already removed, we define its *head* to be the earliest-ordered currently remaining agent in the group. Each remaining agent points to his favorite among the remaining objects. Each remaining object points not to its highest-priority remaining agent, but rather to the head of the group containing that agent. As in TTC, any cycles in this graph must alternate between agents and objects, and distinct cycles are disjoint. Each agent participating in a cycle is assigned the object he points to; these agents and objects are removed.

- This is iterated until all agents and objects are removed.

Just as in TTC, at least one cycle forms at every step, so the algorithm is sure to terminate. This mechanism will be referred to as *serial dictatorship in groups* (SDIG) with priority structure π and orderings $f_1, \ldots, f_m$.

To build some intuition about this mechanism, notice that if all the agents are in one big group, with ordering f, then at step t all the objects point to agent $f(t)$, and this agent and his favorite remaining object are removed. Thus, we recover serial dictatorship. On the other hand, if there are n groups of one agent each, then we simply recover TTC with priority structure π. SDIG can be thought of as a generalization of serial dictatorship with several potential dictators at each step (namely, the heads of the various groups), using priorities to help determine which of the dictators get to choose.

We can randomize SDIG easily enough, given a priority structure π. For each group I_k, choose one of the $|I_k|!$ possible orderings of its members uniformly at random (and independently across groups); then perform SDIG with these orderings. This mechanism will be called *random SDIG* with priority structure π. When all agents are in one big group, random SDIG reduces to random serial dictatorship, regardless of the priority structure.

Now suppose that, in addition to the partition of the set of agents, the priority roles are also partitioned into groups $R_1, \ldots, R_m$, whose sizes correspond to those of the groups of agents. For convenience we may as well assume $R_k = \{r_j \mid i_j \in I_k\}$ for each k. We will speak of a priority role in R_k being *associated with* group I_k and vice versa. A *role assignment function* g is now a bijection from priority roles to agents such that, for each r_j, $g(r_j)$ belongs to the group associated with r_j. We will say that a priority framework ϕ *respects the partition* of priority roles if it satisfies the following condition: for each object, roles in the same group must appear consecutively in its priority role list. That is, we cannot have roles r_j, r_k, r_l appear in that order in some object's list, where r_j, r_l belong to the same group but r_k belongs to a different group. In this case, each object's priority role list gives an unambiguous ordering of the groups relative to each other.

Suppose ϕ respects the partition, g is a role assignment function, and $f_1, \ldots, f_m$ are given orderings of the agents within each group. Consider the priority structure π obtained by *sorting* ϕ using the orderings $f_1, \ldots, f_m$, as follows: for each object o, we convert its priority role list into a priority list by replacing the roles in each group R_k with the agents in I_k, ordering these agents according to f_k. Then, it is not hard to see that SDIG with priority structure $g(\phi)$ and orderings $f_1, \ldots, f_m$ is exactly the same as TTC with priority structure π. More specifically, both mechanisms involve the same graph forming, and the same agents and objects being removed (with the same assignments), at each step.

The construction of this priority structure π does not make any use of g. So we can unambiguously talk about the SDIG mechanism with priority framework ϕ and orderings $f_1, \ldots, f_m$, without reference to g.[3]

[3]This observation requires the assumption that ϕ respects the partition of priority roles; see Section 4.

In particular, given a ϕ that respects the partition, we can define *random SDIG from* ϕ as follows: For each group of agents I_k, randomly choose an ordering f_k, uniformly and independently across groups; then run SDIG with priority framework ϕ and these orderings.

On the other hand, we can also define *random-within-groups TTC from* ϕ as follows: Choose a role assignment function g uniformly at random. (This is equivalent to separately choosing a random bijection from R_k to I_k for each k.) Then run TTC with priority structure $g(\phi)$.

Random SDIG is the analogue of random serial dictatorship in this partitioned context, where we randomly order agents within each group, rather than ordering all n agents. Random-within-groups TTC is the analogue of random TTC, where we constrain each priority role to be given to an agent in the corresponding group.

To illustrate briefly we give another example. Consider again $n = 3$, and suppose there are two groups, $I_1 = \{i_1, i_2\}$ and $I_2 = \{i_3\}$. Let the preferences and the priority framework ϕ now be

$$\begin{array}{llcll} i_1: & o_1 \succ o_2 \succ o_3 & \qquad & o_1: & r_1, r_2, r_3 \\ i_2: & o_3 \succ o_1 \succ o_2 & & o_2: & r_3, r_2, r_1 \\ i_3: & o_1 \succ o_2 \succ o_3 & & o_3: & r_2, r_1, r_3. \end{array}$$

This ϕ respects the partition of roles.

For random-within-groups TTC from ϕ, there are two possible role assignment functions: $(g(r_1), g(r_2), g(r_3)) = (i_1, i_2, i_3)$ or (i_2, i_1, i_3). Generating the corresponding priority structures and then running TTC, we have:

$$(i_1, i_2, i_3) \;\to\; \begin{array}{c} i_1, i_2, i_3 \\ i_3, i_2, i_1 \\ i_2, i_1, i_3 \end{array} \;\to\; (o_1, o_3, o_2) \qquad (i_2, i_1, i_3) \;\to\; \begin{array}{c} i_2, i_1, i_3 \\ i_3, i_1, i_2 \\ i_1, i_2, i_3 \end{array} \;\to\; (o_1, o_3, o_2)$$

Thus random-within-groups TTC produces allocation (o_1, o_3, o_2) with probability 1.

For random SDIG, there are two possible orderings (I_1 may be ordered (i_1, i_2) or (i_2, i_1), and I_2 is always ordered (i_3)). In each case, as described above, we can speedily calculate the result of SDIG by sorting the priority framework using the orderings and then running TTC. This gives

$$(i_1,i_2),(i_3) \;\to\; \begin{array}{c} i_1,i_2,i_3 \\ i_3,i_1,i_2 \\ i_1,i_2,i_3 \end{array} \;\to\; (o_1,o_3,o_2) \qquad (i_2,i_1),(i_3) \;\to\; \begin{array}{c} i_2,i_1,i_3 \\ i_3,i_2,i_1 \\ i_2,i_1,i_3 \end{array} \;\to\; (o_1,o_3,o_2)$$

So in this example, random SDIG also produces (o_1, o_3, o_2) with probability 1.

By now the reader can probably guess what the theorem is going to say.

Theorem 3 *Given corresponding partitions of the set of agents and the set of priority roles, let ϕ be any priority framework that respects the partition. Then random-within-groups TTC from ϕ is equivalent to random SDIG from ϕ.*

Notice that even though we have shown how to compute the outcome of both mechanisms as instances of TTC, and even though our theorem asserts that the two mechanisms lead to the same random allocation, the executions — i.e. the sets of cycles forming at each TTC step — are not the same. In the above example, when applying random-within-groups TTC, with either role assignment function, two agents receive objects at the first step (under role assignment function (i_1, i_2, i_3), the two cycles (i_1, o_1) and (i_2, o_3) form; under (i_2, i_1, i_3), the one cycle (i_1, o_1, i_2, o_3) forms). When applying random SDIG, only one agent gets an object at the first step (under $(i_1, i_2), (i_3)$, the only cycle is (i_1, o_1); under $(i_2, i_1), (i_3)$, the only cycle is (i_2, o_3)).

One special case of Theorem 3 occurs when there is simply one group consisting of all n agents. In this case, every priority framework respects the partition. Random SDIG reduces to random serial dictatorship, and random-within-groups TTC is just random TTC as defined in Subsection 2.1. Thus, we immediately recover Theorem 1.

We now discuss new applications not covered by Theorem 1.

Existing tenants. In the model of house allocation with existing tenants [3], there are $m \leq n$ objects that have (distinct) initial owners, and the rest are unowned. Running serial dictatorship in this model may be individually irrational, meaning that an owner may end up with a worse object than he started with. To remedy this, Abdulkadiroğlu and Sönmez proposed the following variant, *you request my house, I get your turn* (YRMH-IGYT): The agents are arranged in an order. Each agent successively names his favorite from the available objects. If it is unowned, he gets it, and relinquishes any other object he owns. If it is owned, then the owner gets moved up from his position in the order and placed ahead of the current agent. If this process ever fails to terminate, it is because a cycle forms, consisting of existing owners, each requesting the object owned by the next one. Give each such owner his desired object, remove them from the list, and continue.

Sönmez and Ünver [25] considered the following version of the core mechanism: initially, randomly allocate the unowned objects to the unendowed agents, and then run TTC for this exchange economy. They showed that this is equivalent to a randomized version of YRMH-IGYT, where the order is chosen by randomizing uniformly over all orders of the non-owners, and then listing all the owners *at the end* in a fixed order.

We indicate how to recover their equivalence result from Theorem 3. Without loss of generality, the initial owners are agents $i_1, \ldots, i_m$, endowed with objects $o_1, \ldots, o_m$ respectively. Consider the following partition of the set of agents: each agent $i_1, \ldots, i_m$ is in a group by himself, and then the last group consists of all other agents. Partition the set of priority roles correspondingly. Consider the priority framework ϕ given as follows:

- For each owned object o_j, the role r_j has highest priority, then the non-owning roles $r_{m+1}, \ldots, r_n$, and the other owning roles ($r_1, \ldots, r_m$ with r_j omitted) in increasing numerical order.

- For each unowned object o_j, the role r_j has highest priority, then the other non-owning roles in arbitrary order, then $r_1, \ldots, r_m$ in increasing order.

Such a ϕ respects the partition of roles.

Consider what happens when we run random-within-groups TTC. A role assignment function g is just an assignment of the unowned objects to different non-owning agents. So the random-within-groups TTC mechanism just assigns these objects randomly among these agents and then applies ordinary TTC.

On the other hand, running random SDIG is equivalent to doing the following: form an ordering f of all agents, by randomly arranging all the non-owning agents and then putting all the owners in numerical order behind them; set each unowned object's priority list to be this f, and each owned object's list to be the same except that its initial owner is moved to the top; and then run TTC.

Abdulkadiroğlu and Sönmez make the straightforward observation that this latter TTC is equivalent to YRMH-IGYT with the specified ordering f of agents [3, Theorem 3]. Combining this observation with our Theorem 3, we recover exactly the equivalence of Sönmez and Ünver [25].

Partitioned randomization with group priorities. Our next application extends the single versus multiple lottery result of Pathak and Sethuraman [19] to allow schools to prioritize one group of students over another. For motivation, schools might, say, wish to prioritize students who live nearby, or those who bring socioeconomic or gender balance to the student body.

To model this, suppose that the objects are partitioned into classes $C_1, \dots, C_l$, in addition to the partition of agents into groups $I_1, \dots, I_m$. For each class, we are given a priority ordering over the groups. We would like to run TTC, which requires breaking ties for priority within each group so as to form strict orderings of the set of all agents. There are two natural ways to do this.

- Single lotteries: For each group I_k, select an ordering f_k of its members, uniformly at random. For each object o in any class C_j, form a priority list by ordering the groups of agents relative to each other as specified for C_j; within each group I_k, members are ordered consecutively according to f_k. This gives us a priority structure π, and we run TTC with this priority structure.

- Multiple lotteries: For each class C_j and each group I_k, select an ordering f_{jk} of the members of the group, uniformly at random. The f_{jk} are chosen independently across agent groups *and* object classes. For each object $o \in C_j$, form its priority list by ordering the groups relative to each other as specified for C_j, and ordering the agents within each group I_k according to order f_{jk}. This gives us a priority structure π, and we run TTC with this priority structure.

The result is that the single-lottery and multiple-lottery mechanisms are equivalent.

The proof follows the same lines as the original equivalence for partitioned random endowment from Subsection 2.1. It uses an analogue of Corollary 2, in which we consider the set Π of all priority structures that can be obtained via multiple lotteries and partition this Π into equivalence classes (of cardinality $|I_1|! \cdot |I_2|! \cdots |I_m|!$ each). Conditional on each equivalence class, the multiple-lotteries mechanism is equivalent to the single-lotteries mechanism, so the same holds unconditionally as well. We omit further details.

Limited preference conflicts. One last application of Theorem 3 will require further assumptions on agents' preferences. Specifically, suppose the objects are partitioned into classes $C_1, \dots, C_m$ with $|C_k| = |I_k|$ for each k, and it is known that each agent in any I_k prefers objects in C_k above all other objects. If objects in C_k are randomly allocated to distinct agents in I_k and then TTC is applied, the result is equivalent to running random serial dictatorship in each group separately. (This follows directly from [2].) In fact, we can now say more. Let $C'_1, \dots, C'_m$ be any *other* partition of the objects into classes with sizes $|C'_k| = |I_k|$. Suppose we randomly allocate the objects in each C'_k to distinct agents of I_k, and then run TTC. This is still equivalent to RSD within each group. Indeed, it is just random-within-groups TTC from any ϕ that respects the partitions and has each object in C'_k give top priority to a distinct role in R_k. And the corresponding SDIG is equivalent to serial dictatorship within each group, because in the course of the SDIG algorithm, dictators from different groups never make conflicting demands.

We close this section by commenting on an assumption we have made throughout the paper, namely, that the numbers of agents and objects are equal. We can now see that this is not really a substantive restriction. If there are n_I agents and $n_O > n_I$ objects, one can modify the definitions of TTC and SDIG by stipulating that the algorithms terminate when every agent is assigned, and any remaining objects are left unallocated. Random-within-groups TTC then remains equivalent to random SDIG. To see this, just posit an additional group of $n_O - n_I$ "dummy" agents with arbitrary preferences, append the corresponding roles at the end of every object's priority role list, and apply Theorem 3. Similarly, if $n_O < n_I$, we can define the mechanisms to stop when the objects run out, and remaining agents are left unassigned; equivalence again follows from Theorem 3 by defining $n_I - n_O$ dummy objects and adding them to the bottom of every agent's preference ordering.

3 The proof

3.1 Outline

We are ready to commence the proof. The basic approach is bijective, inspired by the constructions in [2, 18]. We will begin by building intuition with an outline of the main ideas involved in proving Theorem 1. Then we will briefly discuss the additional complexities involved in generalizing to the setting of Theorem 3. In the ensuing subsections, we proceed to the full proof of Theorem 3. We do not provide a separate detailed proof for Theorem 1 since it is a special case of Theorem 3.

Suppose we are in the setting of Theorem 1; fix a priority framework ϕ. Let Raf denote the set of role assignment functions, and Ord the set of orderings of all agents. Both of these sets have cardinality $n!$. We would like to construct a bijection $F : Raf \to Ord$ such that, for each $g \in Raf$, TTC with priority structure $g(\phi)$ produces the same allocation as serial dictatorship with order $F(g)$. This will imply the desired equivalence.

Take $g \in Raf$, and consider running the TTC algorithm from $g(\phi)$. Let I^t denote the set of agents removed at step t of the algorithm. (The I^t are separate from the sets

I_k introduced in Subsection 2.2.) We will also need the important distinction between *satisfied* and *unsatisfied* agents (first introduced in [2]). For each step $t > 1$, let an agent $i \in I^t$ be *satisfied* if

- i is pointing to the same object at step t as he was at step $t-1$, and moreover
- this object is pointing to the same agent at step t as at step $t-1$.

Otherwise, we say that i is *unsatisfied.* Notice that every cycle that forms at step t must contain at least one unsatisfied agent: otherwise the cycle would have formed by step $t-1$ and so would have been removed. It will also be convenient to specify that every agent in I^1 is unsatisfied. Denote by S^t and U^t the sets of satisfied and unsatisfied agents in I^t, respectively.

The sets I^t, S^t, U^t partially describe the execution of the TTC algorithm. We use them to construct the ordering $F(g)$ as follows: first, we have the agents in I^1, in some order; then we have the agents in I^2, in some order, beginning with one of the unsatisfied agents; then I^3 in some order, again beginning with an unsatisfied agent, and so forth. (We can do this, since there is some unsatisfied agent at each step — in fact, at least one in each cycle.) This will have the desired property that serial dictatorship with order $F(g)$ produces the same allocation as TTC from $g(\phi)$, since TTC gives each agent in I^t his favorite object from among those not removed at any earlier step (and this object cannot have been assigned to another agent in the same I^t). We do need to check that we can order the agents within each step so that F is a bijection.

Given such an ordering $F(g)$, we can uniquely recover the set I^t of agents removed at each step t of the TTC procedure, as well as the set O^t of objects they receive and the corresponding set of roles $R^t = g^{-1}(I^t)$. For example, I^1 is the largest initial sequence of agents in the order such that their favorite objects are all distinct and these objects' first priority roles in ϕ are also all distinct. (This follows from the fact that the next agent listed after I^1 is unsatisfied.) So we can reconstruct I^1, and also O^1 since each agent in I^1 is assigned his favorite object. We also know the set R^1 of priority roles assigned to agents in I^1: it must consist of the top priority role for each object $o \in O^1$. From this, we know which agents, objects, and priority roles remain at step 2. By iterating on these remaining agents, objects, and priority roles, we can identify I^2, O^2, and R^2, and so forth. Thus, for each step t, we can uniquely reconstruct the sets I^t, O^t, R^t. Notice that this also gives us enough information to reconstruct the sets of satisfied and unsatisfied agents at each step t.

In order for F to be a bijection, we need to be able to uniquely reconstruct which priority role in R^t corresponds to each agent in I^t, given $F(g)$. We use the ordering of agents in I^t among themselves to encode this information. Consider the following two sets:

- $URaf(I^t, R^t, S^t)$, the set of bijections between R^t and I^t ("step-t role assignment functions") for which no cycle in the resulting graph at step t of TTC consists entirely of satisfied agents;
- $UOrd(I^t, S^t)$, the set of orderings of I^t such that the first agent is not satisfied.

Both of these sets have cardinality $(|I^t| - |S^t|) \cdot (|I^t| - 1)!$.[4] So we can choose an arbitrary "step-t" bijection between them, independent of g — for example, list members of both sets in lexicographical order and specify that members in the same position correspond to each other.

We can thus use the ordering of the agents in I^t to encode the necessary information about which agent has which role. Using the step-t bijection to determine the order of agents within the set I^t, for each step t, we can complete the construction of $F(g)$. Conversely, given $F(g)$, the function g can be uniquely reconstructed by inverting these step-t bijections. This ensures that the big map F is a bijection, which is what we wanted.

This outline shows the overall approach. We construct a bijection F between the role assignment functions g for TTC and the orderings for serial dictatorship. But to avoid unnecessary mess, we do not fully specify F explicitly; we only specify how F is built up from its pieces (the step-t bijections), and the pieces need not be actually constructed.

In order to apply this approach in the setting where agents are partitioned into groups, more work is necessary. For one thing, it is no longer so obvious how to reconstruct each successive step I^t of the TTC algorithm from the orderings of agents. With a single ordering of all agents, we simply peel off as many agents as possible from the beginning of the ordering; with orderings for each group separately, we have to show that there is a unique maximal way to peel off agents from each group simultaneously. (This will follow from Lemma 4 below.) In addition, the analogue of the fact that $|URaf(I^t, R^t, S^t)| = |UOrd(I^t, S^t)|$, which is required to ensure existence of the step-t bijection, becomes substantially trickier to prove (Lemma 5); we use an inclusion-exclusion

[4]For $UOrd(I^t, S^t)$ this is easy to see. For $URaf(I^t, R^t, S^t)$, the problem is equivalent to counting the permutations of I^t that have no cycles of satisfied agents. If we try to construct such a permutation h by first assigning a value $h(i)$ for each satisfied agent i in turn, then assigning $h(i)$ for each unsatisfied agent, we find we have $|I^t| - j$ choices for the jth assignment when $j \leq |S^t|$ and $|I^t| - j + 1$ choices when $j > |S^t|$. Multiplying gives the formula.

argument somewhat like the counting method in [19]. The argument that TTC from g gives the same allocation as SDIG from $F(g)$ also becomes more complicated, because the cycles formed in the course of SDIG are no longer trivial. And, inevitably, there is a certain amount of notational baggage to carry throughout the proof; we hope Table 1 will ease the pain.

We now embark on the project of filling in these details.

F	master bijection $Raf \to Tab$
$F_{((\widehat{I}_k);(\widehat{R}_k);s;\widehat{S})}$	bijection from $URaf((\widehat{I}_k);(\widehat{R}_k);s;\widehat{S})$ to $UTab((\widehat{I}_k);(\widehat{R}_k);s;\widehat{S})$
g	a role assignment function
I	set of all agents
I_k	kth group of agents
I^t	set of agents removed at step t
I^{t+}	set of agents remaining at step t
I_k^t	set of agents in group k removed at step t
I_k^{t+}	set of agents in group k remaining at step t
$\widehat{I}$	set of all agents (Subsection 3.2)
$\widehat{I}_k$	set of agents in group k (Subsection 3.2)
L_k	kth list in a tableau
m	number of groups
O	set of all objects
O^t	set of objects removed at step t
O^{t+}	set of objects remaining at step t
O_k^t	set of objects removed at step t that pointed to an agent in group k
R	set of all priority roles
R_k	kth group of priority roles
R^t	set of priority roles removed at step t
R^{t+}	set of priority roles remaining at step t
R_k^t	set of priority roles in group k removed at step t
R_k^{t+}	set of priority roles in group k remaining at step t

Table 1: Frequently used notation

$\widehat{R}$	set of all priority roles (Subsection 3.2)
$\widehat{R}_k$	set of priority roles in group k (Subsection 3.2)
r	a role assignment function (Subsection 3.2)
Raf	set of all role assignment functions
$Raf_{\mathcal{S}}$	set of role assignment functions for which every set in $\mathcal{S}$ is mapped to itself (Subsection 3.2)
S^t	set of satisfied agents removed at step t
S^{t+}	set of potentially satisfied agents remaining at step t
$\widehat{S}$	set of satisfied agents (Subsection 3.2)
s	successor role function
s^t	successor role function for agents removed at step t
s^{t+}	successor role function for agents remaining at step t
T	a tableau (or $F(g)$ in main proof)
T^t	tableau formed from step t of TTC algorithm
T^{t+}	tableau formed by agents remaining at step t
Tab	set of all full tableaus
$Tab_{\mathcal{S}}$	set of tableaus for which every set in $\mathcal{S}$ forms a balanced initial subtableau (Subsection 3.2)
$\bar{t}$	number of steps of TTC algorithm
U^t	set of unsatisfied agents removed at step t
U^{t+}	set of potentially unsatisfied agents remaining at step t
$URaf((\widehat{I}_k); (\widehat{R}_k); s; \widehat{S})$	set of role assignment functions leading to a permutation of agents having no cycle consisting of satisfied agents
$UTab((\widehat{I}_k); (\widehat{R}_k); s; \widehat{S})$	set of full tableaus with no nonempty, balanced initial subtableau consisting of satisfied agents
π	a priority structure
ϕ	a priority framework

Table 1: Frequently used notation (continued)

3.2 Tableaus and permutations

We will begin by developing the conceptual tools that are needed to prove the existence of the step-t bijections (Lemma 5). Along the way, we will also prove a lemma that will subsequently be crucial in reconstructing a role assignment function g from the corresponding orderings (Lemma 4). The proofs of these two lemmas are logically separate from the rest of the main proof, and the proof for Lemma 5 in particular is somewhat involved, so the reader only interested in the main bijective argument can skip over them.

In this subsection, we take as given m disjoint finite sets of agents $\widehat{I}_1, \ldots, \widehat{I}_m$, and m disjoint sets of priority roles $\widehat{R}_1, \ldots, \widehat{R}_m$, with $|\widehat{I}_k| = |\widehat{R}_k|$ for each k. Let $\widehat{I} = \cup_k \widehat{I}_k$ and $\widehat{R} = \cup_k \widehat{R}_k$. We also assume given a *successor role* function $s : \widehat{I} \to \widehat{R}$. We are especially, but not exclusively, interested in the case where s is a bijection. (When we apply the results of this subsection, $\widehat{I}_k$ and $\widehat{R}_k$ will be subsets of the original sets I_k of agents and R_k of priority roles; $s(i)$ will denote the highest-priority role for the object to which agent i points, at some step of the TTC algorithm. But for now we can just think of $\widehat{I}_k, \widehat{R}_k, s$ abstractly.)

We are interested in bijections $r : \widehat{R} \to \widehat{I}$ that assign the agents to the priority roles. For purposes of this subsection, such a bijection is called a *role assignment function* if $r(\widehat{R}_k) = \widehat{I}_k$ for each k. If $s : \widehat{I} \to \widehat{R}$ is a bijection, then for any bijection $r : \widehat{R} \to \widehat{I}$, the composition $r \circ s$ is a permutation of $\widehat{I}$.

A *tableau* will denote a sequence of m ordered lists $(L_1, \ldots, L_m)$, where each L_k is a (possibly empty) list of distinct elements of $\widehat{I}_k$. (Note that to define a tableau we need only to have specified the sets $\widehat{I}_k$, not the $\widehat{R}_k$ or s.) The tableau is *full* if each L_k contains every element of $\widehat{I}_k$. Thus, a full tableau specifies an ordering of all the elements of $\widehat{I}_k$, for each k. The tableau is *empty* if every L_k is empty. The tableau $(L_1, \ldots, L_m)$ is an *initial subtableau* of $(L'_1, \ldots, L'_m)$ if, for each k, the list L_k consists of the first l_k elements of L'_k, for some l_k (possibly zero).

We will be a little notationally sloppy and write $i \in T$ to indicate that agent i appears in tableau T; likewise $|T|$ will denote the number of agents appearing in T, and if L is a list (or a set) of agents, $T \cap L$ will denote the set of agents appearing in both T and L.

We will say the tableau $(L_1, \ldots, L_m)$ is *balanced* if, for each k, the number of agents $i \in \widehat{I}$ appearing in the tableau with $s(i) \in \widehat{R}_k$ is equal to the length of L_k. It is *consistent* if it does not contain two different agents i, i' with $s(i) = s(i')$. Notice that the tableau is balanced and consistent if and only if there exists some role assignment function r such that $r \circ s$ permutes the set of agents appearing in the tableau. Also, if s is a bijection, then every tableau is automatically consistent, and a full tableau is always balanced, as is the empty tableau.

These definitions allow us to state the following important lemma.

Lemma 4 *Let $T = (L_1, \ldots, L_m)$ be a tableau and T_1, T_2 two balanced initial subtableaus. Their union — defined by taking, for each k, the list of elements of L_k appearing in either T_1 or T_2, in order according to L_k — is again a balanced initial subtableau of T. Moreover, if T_1 and T_2 are both consistent, then so is their union.*

Likewise, the intersection of two balanced initial subtableaus T_1, T_2 — defined by taking, for each k, the list of elements of L_k appearing in both T_1 and T_2 — is a balanced initial subtableau of T, as well as of T_1 and of T_2.

Proof: Let $T_\cup$ denote the union. It is clear that it is an initial subtableau: if it contained some agent $i \in L_k$ such that an earlier agent $j \in L_k$ did not appear in $T_\cup$, then either T_1 or T_2 would again contain i and omit j, contrary to the assumption that T_1 and T_2 are initial subtableaus of T.

Now to show that $T_\cup$ is balanced. For each k, we want to show that the total number of agents $i \in T_\cup$ satisfying $s(i) \in \widehat{R}_k$ equals the number of agents in L_k that appear in $T_\cup$. Let l_{1k} be the number of agents in L_k that appear in T_1, and l_{2k} the number of agents in L_k that appear in T_2; without loss of generality assume $l_{1k} \geq l_{2k}$. Then

$$|\{i \in T_\cup \mid s(i) \in \widehat{R}_k\}| \geq |\{i \in T_1 \mid s(i) \in \widehat{R}_k\}| = l_{1k} = \max\{l_{1k}, l_{2k}\} = |T_\cup \cap L_k| \qquad (1)$$

where the first equality holds because T_1 is balanced, and the third equality holds because T_1, T_2 are initial subtableaus, so every agent in L_k appearing in T_2 also appears in T_1.

Thus for each k we get $|\{i \in T_\cup \mid s(i) \in \widehat{R}_k\}| \geq |T_\cup \cap L_k|$. Adding over all k gives

$$|T_\cup| \geq |T_\cup|.$$

So the equality must hold for each k individually; that is, $|\{i \in T_\cup \mid s(i) \in \widehat{R}_k\}| = |T_\cup \cap L_k|$, which says that $T_\cup$ is balanced.

If T_1, T_2 are both consistent, we wish to show that $T_\cup$ is as well. Suppose that there are two different agents $j, j' \in T_\cup$ with $s(j) = s(j') \in \widehat{R}_k$. Let l_{1k}, l_{2k} be as above, and again assume without loss of generality that $l_{1k} \geq l_{2k}$. The above argument shows that equality holds in (1), so that $\{i \in T_\cup \mid s(i) \in \widehat{R}_k\} = \{i \in T_1 \mid s(i) \in \widehat{R}_k\}$; thus, j, j' both appear in T_1. But then this violates the consistency of T_1. This contradiction shows that $T_\cup$ is consistent.

Finally, let $T_\cap$ denote the intersection of T_1 and T_2. It is again an initial subtableau (of T, T_1, and T_2): if it contains some agent $i \in L_k$, then every earlier agent in L_k must also appear in both T_1 and T_2, hence in $T_\cap$. The proof that $T_\cap$ is balanced is essentially identical to the proof for $T_\cup$, with all the inequalities reversed and with max replaced by min in (1). $\square$

Next, we proceed to the counting lemma which will be used to produce the step-t bijections. For this we will assume that the successor role function s is a bijection. We also need to assume that, in addition to the sets of agents and priority roles and the successor role function, we are given a designated subset $\widehat{S} \subseteq \widehat{I}$. We will refer to agents in $\widehat{S}$ as *satisfied*.

We define two important sets as follows:

- $URaf(\widehat{I}_1, \ldots, \widehat{I}_m; \widehat{R}_1, \ldots, \widehat{R}_m; s; \widehat{S})$ is the set of role assignment functions r with the following property: the permutation $r \circ s$ of $\widehat{I}$ has no cycles consisting entirely of satisfied agents.

- $UTab(\widehat{I}_1, \ldots, \widehat{I}_m; \widehat{R}_1, \ldots, \widehat{R}_m; s; \widehat{S})$ is the set of full tableaus with the following property: there is no nonempty, balanced initial subtableau consisting entirely of satisfied agents.

For brevity, we will denote these sets by $URaf$ and $UTab$ respectively as long as there is no confusion about their arguments. They are the generalizations to the partitioned setting of the sets $URaf$ and $UOrd$ described in Subsection 3.1.

Lemma 5 *Suppose s is a bijection. Then $|URaf| = |UTab|$.*

Proof: Let $\mathcal{P}$ denote the power set of $\widehat{S}$, minus the empty set; that is, $\mathcal{P}$ is the set of all nonempty sets of satisfied agents. For any $\mathcal{S} \subseteq \mathcal{P}$, define the following two sets:

- $Raf_{\mathcal{S}}$ is the set of all role assignment functions r such that for each $S \in \mathcal{S}$, the permutation $r \circ s$ of the agents maps S to itself.

- $Tab_{\mathcal{S}}$ is the set of full tableaus such that for each $S \in \mathcal{S}$, there exists a balanced initial subtableau whose set of agents is exactly S.

In particular, $Tab_\emptyset$ is the set of all full tableaus, and $Raf_\emptyset$ is the set of all role assignment functions.

The inclusion-exclusion principle implies that

$$|URaf| = \sum_{\mathcal{S}\subseteq\mathcal{P}} (-1)^{|\mathcal{S}|}|Raf_{\mathcal{S}}|; \tag{2}$$

$$|UTab| = \sum_{\mathcal{S}\subseteq\mathcal{P}} (-1)^{|\mathcal{S}|}|Tab_{\mathcal{S}}|. \tag{3}$$

We would like to show that these two sums are equal. Our strategy will be to show that for some choices of $\mathcal{S}$, the term in (2) is equal to the corresponding term in (3); and for the remaining $\mathcal{S}$'s, the corresponding terms in (2) cancel in pairs, and the terms in (3) also cancel in pairs.

To this end, let $\mathcal{Q}$ be the power set of $\mathcal{P}$, and let

$$\mathcal{Q}^* = \{\mathcal{S} \in \mathcal{Q} \mid \text{there exist } S, S' \in \mathcal{S} \text{ with } S \not\subseteq S' \text{ and } S' \not\subseteq S\}.$$

Thus, $\mathcal{Q}^*$ consists of all the collections of nonempty sets of satisfied agents that contain at least two mutually non-nested sets.

We will define an involution $\psi : \mathcal{Q}^* \to \mathcal{Q}^*$ as follows. First, let $\mathcal{Q}^*_2 = \{\mathcal{S} \in \mathcal{Q}^* \mid |\mathcal{S}| = 2\}$ be the collection of all pairs of non-nested sets of satisfied agents. Order these pairs according to the size of the larger set in each pair — that is, if $\{S_1, S_1'\}, \{S_2, S_2'\}$ are pairs with $\max\{|S_1|, |S_1'|\} < \max\{|S_2|, |S_2'|\}$, then $\{S_1, S_1'\}$ appears before $\{S_2, S_2'\}$. (Ties may be broken arbitrarily.) Now, for any $\mathcal{S} \in \mathcal{Q}^*$, let S, S' be the pair of non-nested sets in $\mathcal{S}$ that appears earliest in this ordering. Then, if $S \cup S' \in \mathcal{S}$, let $\psi(\mathcal{S}) = \mathcal{S} \setminus \{S \cup S'\}$; otherwise let $\psi(\mathcal{S}) = \mathcal{S} \cup \{S \cup S'\}$. In words, to apply ψ, we find the earliest pair of sets $S, S' \in \mathcal{S}$ that are mutually non-nested, and we change $\mathcal{S}$ by either adding or deleting their union $S \cup S'$.

Because S, S' are mutually non-nested and the pairs have been ordered by size of the larger set, any pair containing $S \cup S'$ appears later in the ordering of $\mathcal{Q}^*_2$ than $\{S, S'\}$. It follows that S and S' are also the earliest mutually non-nested sets in $\psi(\mathcal{S})$, and therefore $\psi(\psi(\mathcal{S})) = \mathcal{S}$. That is, ψ is indeed an involution. It is also clear that $|\psi(\mathcal{S})| = |\mathcal{S}| \pm 1$.

Now let S, S' be mutually non-nested sets of satisfied agents, and r a role assignment function. Notice that if $r \circ s$ maps S to itself and also maps S' to itself, then $r \circ s$ maps $S \cup S'$ to itself. This means that for $\mathcal{S} \in \mathcal{Q}^*$, $r \in Raf_{\mathcal{S}}$ if and only if $r \in Raf_{\psi(\mathcal{S})}$. That is, $Raf_{\mathcal{S}} = Raf_{\psi(\mathcal{S})}$. Looking back at (2), for any $\mathcal{S} \in \mathcal{Q}^*$, the terms corresponding to $\mathcal{S}$ and $\psi(\mathcal{S})$ are of equal magnitude and opposite sign. So we can cancel them out. This leaves us with

$$|URaf| = \sum_{\mathcal{S}\in\mathcal{Q}\setminus\mathcal{Q}^*} (-1)^{|\mathcal{S}|}|Raf_{\mathcal{S}}|. \tag{4}$$

Similarly, let $T = (L_1, \ldots, L_m)$ be a full tableau such that there is a balanced initial subtableau T_S whose set of agents is exactly S, and a balanced initial subtableau $T_{S'}$ whose set of agents is exactly S'. Lemma 4 tells us that there is a balanced initial subtableau whose set of agents is $S \cup S'$. This means that $T \in Tab_{\mathcal{S}}$ if and only if $T \in Tab_{\psi(\mathcal{S})}$. That is, $Tab_{\mathcal{S}} = Tab_{\psi(\mathcal{S})}$. So again, we can cancel out these two terms from (3) for each $\mathcal{S} \in \mathcal{Q}^*$, and we are left with

$$|UTab| = \sum_{\mathcal{S}\in\mathcal{Q}\setminus\mathcal{Q}^*} (-1)^{|\mathcal{S}|}|Tab_{\mathcal{S}}|. \tag{5}$$

Now consider any $\mathcal{S} \in \mathcal{Q} \setminus \mathcal{Q}^*$. That is, $\mathcal{S}$ is a collection of nonempty subsets of $\widehat{S}$ that are nested. We will show that $|Raf_{\mathcal{S}}| = |Tab_{\mathcal{S}}|$. Write

$$\mathcal{S} = \{S_1, \ldots, S_q\} \qquad \text{with} \qquad S_1 \subset S_2 \subset \cdots \subset S_q.$$

Let $S_0 = \emptyset$ and $S_{q+1} = \widehat{I}$ for convenience. Write $S_{kl} = \widehat{I}_k \cap (S_l \setminus S_{l-1})$ for each $k = 1, \ldots, m$ and $l = 1, \ldots, q+1$. The $m(q+1)$ sets S_{kl} form a partition of $\widehat{I}$.

Suppose that there exist k and l such that

$$|\widehat{I}_k \cap S_l| \neq |\{i \in S_l \mid s(i) \in \widehat{R}_k\}|.$$

(We must have $1 \leq l \leq q$.) Then there are no tableaus having a balanced initial subtableau with elements S_l. There is also no role assignment function r such that $r \circ s$ maps S_l into itself, since for such an r to exist we would have to have

$$|\widehat{I}_k \cap S_l| = |(r \circ s)^{-1}(\widehat{I}_k) \cap S_l| = |\{i \in S_l \mid s(i) \in r^{-1}(\widehat{I}_k)\}| = |\{i \in S_l \mid s(i) \in \widehat{R}_k\}|.$$

Hence, $|Raf_{\mathcal{S}}| = |Tab_{\mathcal{S}}| = 0$ in this case.

Suppose, on the other hand, that for all k and l we have

$$|\widehat{I}_k \cap S_l| = |\{i \in S_l \mid s(i) \in \widehat{R}_k\}|. \tag{6}$$

Denote $S'_{kl} = (s^{-1}(\widehat{R}_k)) \cap (S_l \setminus S_{l-1})$ for each $k = 1, \ldots, m$ and $l = 1, \ldots, q+1$. These sets again form a partition of $\widehat{I}$. Notice that

$$|S'_{kl}| = |s^{-1}(\widehat{R}_k) \cap S_l| - |s^{-1}(\widehat{R}_k) \cap S_{l-1}| = |\widehat{I}_k \cap S_l| - |\widehat{I}_k \cap S_{l-1}| = |S_{kl}|.$$

Now, consider a bijection $r : \widehat{R} \to \widehat{I}$. r is a role assignment function if and only if $r \circ s$ maps $s^{-1}(\widehat{R}_k)$ to $\widehat{I}_k$ for each k. Therefore, the joint requirement that r should be a role assignment function and $r \circ s$ should map each S_l to itself is equivalent to the requirement that $r \circ s$ should map S'_{kl} to S_{kl}, for all k and l. The number of such bijections r is then

$$|Raf_{\mathcal{S}}| = \prod_{k=1}^{m} \prod_{l=1}^{q+1} |S_{kl}|!. \tag{7}$$

Consider a full tableau $T = (L_1, \ldots, L_m)$. It is evident that the following two conditions on T are equivalent:

- for each $l = 1, \ldots, q$, there exists an initial subtableau consisting of agents S_l;
- for each $k = 1, \ldots, m$, the list L_k consists of the agents in S_{k1} in some order, followed by the agents in S_{k2} in some order, …, the agents in $S_{k(q+1)}$ in some order.

Moreover, if these initial subtableaus exist, they are automatically balanced, by (6). Therefore, the number of full tableaus for which there exists a balanced initial subtableau with agent set S_l, for every $l = 1, \ldots, q$, is just

$$|Tab_{\mathcal{S}}| = \prod_{k=1}^{m} \prod_{l=1}^{q+1} |S_{kl}|!. \tag{8}$$

By combining (7) and (8), we see that $|Raf_{\mathcal{S}}| = |Tab_{\mathcal{S}}|$ for every $\mathcal{S} \in \mathcal{Q} \backslash \mathcal{Q}^*$. Therefore, the sums in (4) and (5) are equal term-by-term, giving $|URaf| = |UTab|$. □

3.3 The main argument

We now return to the setting and notation of Subsection 2.2, where overall sets of agents $I_1, \ldots, I_m$ and priority roles $R_1, \ldots, R_m$ are exogenously given.

As one more preliminary, we prepare the application of Lemma 5. Consider any sets $\widehat{I}_1, \ldots, \widehat{I}_m$ and $\widehat{R}_1, \ldots, \widehat{R}_m$ of agents and priority roles, respectively, such that $\widehat{I}_k \subseteq I_k$, $\widehat{R}_k \subseteq R_k$, $|\widehat{I}_k| = |\widehat{R}_k|$ for each k; consider further any bijection $s : \cup_k \widehat{I}_k \to \cup_k \widehat{R}_k$ and any subset $\widehat{S} \subseteq \cup_k \widehat{I}_k$. For each such choice of the sets $\widehat{I}_k, \widehat{R}_k, \widehat{S}$ and choice of s, we fix an arbitrary bijection

$$F_{((\widehat{I}_k);(\widehat{R}_k);s;\widehat{S})} : URaf((\widehat{I}_k);(\widehat{R}_k);s;\widehat{S}) \to UTab((\widehat{I}_k);(\widehat{R}_k);s;\widehat{S}).$$

We know that some such bijection exists because Lemma 5 tells us that the two sets involved have the same cardinality. We will take all these bijections to be fixed in what follows, and we will not need to know anything about their exact content.

Proof of Theorem 3: Let Raf denote the set of all possible role assignment functions g. Let Tab denote the set of all m-tuples $(f_1, \ldots, f_m)$, where each f_k is an ordering of I_k (that is, the set of full tableaus for the agent groups I_k).

We are given the priority framework ϕ. We wish to construct a bijection $F : Raf \to Tab$ with the following property: for each $g \in Raf$, TTC from $g(\phi)$ produces the same allocation as SDIG from ϕ and orderings given by $F(g)$. Once we have such a bijection, Theorem 3 will be immediate.

Moreover, because $|Raf| = |I_1|! \cdot |I_2|! \cdots |I_m|! = |Tab|$, if we can construct F and show that it is one-to-one, then it will immediately follow that it is bijective.

Our proof will be broken down into four parts: (1) constructing F; (2) verifying that TTC from $g(\phi)$ is equivalent to SDIG from ϕ and $F(g)$; (3) showing how the agents, objects, and roles removed at each step of TTC can be uniquely recovered from $F(g)$; (4) using Part 3 to uniquely reconstruct all of g from $F(g)$, showing that F is one-to-one.

Some statements within the parts will be highlighted as claims and subclaims. Their proofs will be delineated by the ending symbols $\diamond$ and $\triangle$ respectively.

Part 1: Constructing the map F. Let a role assignment function g be given. Run the TTC algorithm from $g(\phi)$. Following the terminology established in Subsection 3.1, we let I^t be the set of agents removed at step t, O^t the set of objects removed, and $R^t = g^{-1}(I^t)$ the set of roles corresponding to I^t. Also let $S^t, U^t \subseteq I^t$ be the sets of satisfied and unsatisfied agents, respectively. Finally, let $\bar{t}$ be the total number of steps required. We will construct a tableau T^t based on what happens at step t of the algorithm, then combine these tableaus $T^1, \ldots, T^{\bar{t}}$ to form one big tableau, and this big tableau will be $F(g)$.

For each t and each $k = 1, \ldots, m$, let $I_k^t = I^t \cap I_k$, and $R_k^t = R^t \cap R_k$. Define the successor role function $s^t : I^t \to R^t$ as follows: For each $i \in I^t$, there is a unique agent j such that i points to an object that points to j (at step t of the algorithm); let $s^t(i) = g^{-1}(j)$. Since the agents in I^t belong to cycles formed at step t, $(g|_{R^t}) \circ s^t$ permutes the agents in I^t by advancing two steps along each cycle, hence s^t is a bijection from I^t

to R^t. Also, $g|_{R^t}$ is clearly a role assignment function (in the language of Subsection 3.2) with respect to the agent sets I^t_k and priority role sets R^t_k.

Claim. The permutation $(g|_{R^t}) \circ s^t$ has no cycles consisting entirely of satisfied agents.

Proof. Suppose there is such a cycle. By the definition of satisfied agents, $t > 1$, and each agent i in the cycle is pointing to the same object at step t as at step $t-1$; moreover this object is pointing to the same agent at step t as at step $t-1$, namely $((g|_{R^t}) \circ s^t)(i)$. Hence, the agents in the cycle, together with the objects they point to, form a cycle in the graph at step t of the execution of the TTC algorithm, and this cycle was already present at step $t-1$. But this means these agents and objects should have been removed before step t — a contradiction. $\diamond$

Thus we have

$$g|_{R^t} \in URaf(I^t_1, \dots, I^t_m; R^t_1, \dots, R^t_m; s^t; S^t).$$

So, applying the bijection $F_{(I^t_1,\dots,I^t_m;R^t_1,\dots,R^t_m;s^t;S^t)}$ defined at the beginning of this subsection, we get a tableau

$$T^t \in UTab(I^t_1, \dots, I^t_m; R^t_1, \dots, R^t_m; s^t; S^t).$$

Do this for each step t, obtaining tableaus $T^1, \dots, T^{\bar{t}}$. Each tableau T^t contains exactly those agents that are in I^t, so each agent appears in one tableau.

Finally, concatenate these tableaus to form a full tableau T. That is, for each $k = 1, \dots, m$, the kth list of T is formed by writing the kth list of T^1, followed by the kth list of T^2, and so on in succession. We have $T \in Tab$. Define $F(g) = T$.

Part 2: Showing equivalence between g and $F(g)$. We need to show that TTC from $g(\phi)$ produces the same allocation as SDIG from ϕ and $F(g)$.

Let π be the priority structure produced by sorting ϕ using the orderings given by the tableau $F(g)$. From the discussion in Subsection 2.2, we know that SDIG from ϕ and $F(g)$ is equivalent to TTC from π. So we just need to show that TTC from $g(\phi)$ and TTC from π produce the same allocation.

One might hope to prove this by showing that $g(\phi)$ and π lead to the same cycles at each step of the TTC algorithm. Unfortunately, things are not quite so simple. The two priority structures may not lead to the same cycles, and even if they do, these cycles may not appear in the same order. (Indeed, we saw this in the example in Subsection 2.2.) Hence the need for the more complex argument that follows.

Continue to let I^t, O^t denote the agents and objects removed at step t of TTC from $g(\phi)$. Let $I^t_k = I^t \cap I_k$ as before. Let O^t_k denote the set of objects in O^t that, at step t of TTC from $g(\phi)$, were pointing to an agent in I^t_k. Since I^t and O^t formed cycles, we have

$|O^t_k| = |I^t_k|$.

We now define the following modified version of TTC from π.

- Step 1:
  - Substep 1: Each agent initially points to his favorite object, and each object points to the highest-priority agent according to π. For any cycle that forms *such that all of the agents in the cycle belong to* I^1, assign each agent in the cycle the object he points to, and remove these agents and objects. Any other cycles are left untouched.
  - Substep $t' > 1$: Each agent points to his favorite remaining object, and each object points to the highest-priority remaining agent according to π. For any cycle forms all of whose agents belong to I^1, assign each agent the object he points to, and remove these agents and objects.

    Repeat this process until there are no cycles remaining that can be removed. Then proceed to step 2.
- Step t, $1 < t \leq \bar{t}$: This is analogous to step 1.
  - At each substep t', each agent points to his favorite object among those currently remaining, and each object points to the highest-priority agent according to π among those currently remaining. If any cycle forms, all of whose agents belong to I^t, then assign each agent in the cycle the object he points to, and remove these agents and objects. Repeat this process until it is no longer possible to remove cycles. Then proceed to step $t+1$, if $t < \bar{t}$. If $t = \bar{t}$, end.

It is not obvious that this algorithm will successfully assign every agent an object. However, we will show:

Claim. At step t of the modified TTC algorithm, all the agents in I^t are in fact removed, and each agent in I^t gets the same object in this algorithm as he does in TTC from $g(\phi)$ (which in particular lies in O^t).

Proof. We use induction on t. So suppose we find ourselves at the beginning of step t of modified TTC from π. By the induction hypothesis, the agents in $I^1, \ldots, I^{t-1}$ and the objects in $O^1, \ldots, O^{t-1}$ have been removed (and clearly no others have).[5]

[5]This same argument applies for the base case, in which case no agents or objects have been removed yet.

Each agent is pointing to his favorite of the remaining objects, which is the same as in step t of TTC from $g(\phi)$. In particular, each agent in I^t is pointing to a distinct object in O^t.

Subclaim. At every substep t' of step t (of modified TTC), each agent in I^t is still pointing to the same object as at the beginning of step t.

Proof. By induction on t': If at substep $t'-1$, agent $i \in I^t$ was pointing to an object that got removed, then by the induction hypothesis he was the only agent in I^t pointing to that object, so he must have also been removed. So for every remaining agent in I^t at substep t', the object he was previously pointing to is still available, hence is still his favorite. $\triangle$

In particular, this subclaim ensures that only objects in O^t can be removed at step t.

Subclaim. At the start of step t, for each k and each object $o \in O^t_k$, the top $|O^t_k|$ remaining agents in o's priority list in π are simply the agents of I^t_k in some order.

Proof. First look at o's priority list in $g(\phi)$. After the agents in $I^1, \ldots, I^{t-1}$ have been removed, the highest-priority of the remaining agents belongs to I_k (because $o \in O^t_k$). Moreover, the agents in I_k occur consecutively in o's priority list (because the priority structure respects the partition), and there are at least $|I^t_k| = |O^t_k|$ of these agents remaining at the start of step t.

Now, π is obtained from $g(\phi)$ be reordering the agents within groups. Hence, in π also, after the agents in $I^1, \ldots, I^{t-1}$ have been removed, the top $|O^t_k|$ remaining agents in o's priority list are all in I_k. Moreover, within $F(g)$, the agents within I_k have been sorted so that those in I^1_k occur before those in I^2_k, which in turn occur before those in I^3_k, and so forth. So o's priority list in π also has I_k arranged in this order. Hence, once agents in $I^1, \ldots, I^{t-1}$ have been removed, o's remaining priority list must start with the agents of I^t_k, in some order. $\triangle$

Subclaim. At each substep t' of any step t, for each k, each object $o \in O^t_k$ is pointing to an agent in I^t_k.

Proof. By contradiction: Consider the smallest t' such that, for some k, some remaining object $o \in O^t_k$ is not pointing to an agent in I^t_k. By the previous subclaim, I^t_k was at the top of o's priority list at the start of step t, so it must be the all the agents in I^t_k have been removed at some substeps $1, \ldots, t'-1$. But each such agent can only have been removed together with a (distinct) object in O^t pointing to him. Moreover, at each of these earlier substeps, for each $k' \neq k$, each object in $O^t_{k'}$ was pointing to an agent in $I^t_{k'}$ (by the minimality of t'), not to an agent in I^t_k. That is, whenever an agent in I^t_k was removed, the object in his cycle pointing to him (which was also removed) must have been in O_k^t. Thus, at substep t', at least $|I_k^t|$ objects from O_k^t have already been removed. Since $|O_k^t| = |I_k^t|$, this means that all of O_k^t has been removed, contrary to the assumption that $o \in O_k^t$ is still surviving. $\triangle$

At this point we have shown that throughout step t, every remaining agent in I^t is pointing to an object in O^t, and every remaining object in O^t is pointing to an agent in I^t.

It follows that, as long as the agents in I^t and objects in O^t have not all been removed, these agents and objects form an induced subgraph in which every vertex still has outdegree 1, so there must be a cycle — and we can remove it. This shows that we do succeed in removing all of I^t and O^t at step t.

We also showed that, through step t, each agent in I^t continues pointing to the same object (until he gets removed). This is his favorite object in O^t, which is the same object he gets assigned in TTC from $g(\phi)$. Hence, each agent in I^t gets the same object in modified TTC from π as in TTC from $g(\phi)$. This completes the proof of the claim. $\diamond$

Now, our modified TTC algorithm is just a version of TTC from π where we do not necessarily remove every cycle at each step. Since we now know that all agents and objects do eventually get removed, it is an instance of slow TTC, and so it produces the same allocation as TTC from π (Lemma 6 in the appendix). Thus, the claim ensures that TTC from $g(\phi)$ and TTC from π produce the same allocation, wrapping up Part 2.

Part 3: Showing that $F(g)$ uniquely determines the agents, objects, and roles at each step. We continue to denote by I^t, O^t, R^t, S^t, U^t the agents, objects, roles, satisfied and unsatisfied agents at step t of TTC from $g(\phi)$. We want to show that $F(g)$ uniquely determines I^t, O^t, R^t. That is, if $g, \widetilde{g} \in Raf$ with $F(g) = F(\widetilde{g})$, and $\widetilde{I}^t, \widetilde{O}^t, \widetilde{R}^t$ are the corresponding sets generated from $\widetilde{g}(\phi)$, then we want to show

$$\widetilde{I}^t = I^t, \quad \widetilde{O}^t = O^t, \quad \widetilde{R}^t = R^t$$

for each t. We will avoid explicitly mentioning all the tilde-adorned sets and simply use the phrase "uniquely determined" throughout.

Up until now, we have avoided formal notation for the sets of agents, objects, and so forth remaining at the beginning of each step of the TTC algorithm from $g(\phi)$, but these will now be necessary. So, for each step t, define the following sets:

- $I^{t+} = I \setminus \cup_{t'<t} I^{t'}$ is the set of agents remaining at (the beginning of) step t. Also let $I_k^{t+} = I^{t+} \cap I_k$.

- $O^{t+} = O \setminus \cup_{t'<t} O^{t'}$ is the set of objects remaining at step t.

- $R^{t+} = R \setminus \cup_{t'<t} R^{t'}$ is the set of priority roles remaining at step t. Also let $R_k^{t+} = R^{t+} \cap R_k$.

- $s^{t+} : I^{t+} \to R^{t+}$ is defined as follows: for each agent $i \in I^{t+}$, if at step t, i points to object o which in turn points to agent j, then $s^{t+}(i) = g^{-1}(j)$. Notice that s^{t+} need not be a bijection, but its restriction to I^t coincides with the bijection $s^t : I^t \to R^t$.

- T^{t+} is the tableau formed by concatenating tableaus $T^t, T^{t+1}, \dots, T^{\bar{t}}$ defined in Part 1 above. Equivalently, it is obtained by taking the tableau $T = F(g)$ and removing the agents not in I^{t+}.

- If $t > 1$, define an agent in I^{t+} to be *potentially satisfied at step t* if he is pointing to the same object at step t as at step $t-1$, and moreover this object is pointing to the same agent at t as at $t-1$. Then let S^{t+} be the set of all potentially satisfied agents in I^{t+} at step t. If $t = 1$, let $S^{t+} = \emptyset$. Notice that $S^t = S^{t+} \cap I^t$.

- $U^{t+} = I^{t+} \setminus S^{t+}$, the set of *potentially unsatisfied agents at step t*. Notice that $U^t = U^{t+} \cap I^t$.

These data describe the state of the TTC algorithm at the beginning of step t. (Actually T^{t+} depends on future steps of the algorithm as well, but given $F(g)$, it can be reconstructed in terms of the state of the algorithm at step t.) We will show how they can be used to uniquely characterize T^t, and this will lead to an iterative procedure for reconstructing the sets I^t, O^t, R^t from the tableau $T = F(g)$.

In the course of this proof we will refer to several tableaus and several successor role functions. For convenience, the latter will sometimes be indicated in square brackets; for example, "$T^t[s^t]$ is balanced" will mean that T^t is balanced with respect to s^t. When there is no need to identify the successor role function, we will leave it out. There will be no ambiguity about the partitions of the agents and roles, which we can take to be $I_1, \dots, I_m$ and $R_1, \dots, R_m$ throughout.

Claim. $T^{t+}[s^{t+}]$, has no nonempty, balanced initial subtableau consisting entirely of agents who are potentially satisfied at step t.

We already know that $T^t[s^t]$ has no such subtableau, since by construction T^t lies in the set $UTab((I_k^t); (R_k^t); s^t; S^t)$. Showing this fact for T^{t+} takes a little more work, since there might conceivably be such a subtableau that is not contained in T^t.

Proof. Suppose such a subtableau T' exists. Consider the first step $t' \geq t$ at which one of the agents in T' is removed.

Subclaim. Let i be any agent in T'. Suppose that at step t, i points to object o, which in turn points to agent j. Then i continues to point to o and o continues to point to j throughout steps $t, t+1, \ldots, t'$.

Proof. Suppose $j \in I_k$. Because T' is balanced with respect to s^{t+}, it contains some agent in I_k. Because none of the agents in T' is removed before step t', we see that no agent in I_k is removed at steps $t, t+1, \ldots, t'-1$. (If any such agent were removed, then the first agent in the kth list of T^{t+} would be among the first to be removed — but this agent must be in T', because T' is an initial subtableau.) Thus, j cannot be removed at any of these steps, and then o (which points to j) cannot be removed at any such step either. So i continues to point to o, and o continues to point to j. $\triangle$

In particular, this shows that each $i \in T'$ remains potentially satisfied at each step $t, t+1, \ldots, t'$. This also shows that $s^{t'+}(i) = s^{t+}(i)$ for all agents $i \in T'$. This in turn means that T' forms a balanced, initial subtableau of $T^{t'+}[s^{t'+}]$.

Now consider the agents in T' that are removed at step t': these agents constitute the intersection of $T^{t'}$ and T'. By Lemma 4, this intersection is a balanced initial subtableau of $T^{t'}[s^{t'+}]$ and so of $T^{t'}[s^{t'}]$. Thus $T^{t'}[s^{t'}]$ has a nonempty, balanced initial subtableau consisting entirely of agents that are potentially satisfied at step t', and so satisfied. But this means that

$$T^{t'} \notin UTab(I_1^{t'}, \ldots, I_m^{t'}; R_1^{t'}, \ldots, R_m^{t'}; s^{t'}; S^{t'}),$$

which is a contradiction to the way that $F(g)$ was constructed. This proves the claim. $\diamond$

Next: We know that T^t is a balanced, consistent initial subtableau of $T^{t+}[s^{t+}]$. We can now show more:

Claim. T^t is a *maximal* balanced, consistent initial subtableau of $T^{t+}[s^{t+}]$.

Proof. Again, by contradiction. Suppose that T' is a balanced, consistent initial subtableau of T^{t+} that strictly contains T^t. Let T^- be the tableau obtained from T' by deleting the agents in T^t. Notice that T^- is an initial subtableau of $T^{(t+1)+}$, and it is again balanced with respect to s^{t+}.

Consistency implies that the agents in T' all point to different objects at step t. In particular, since the agents in T^t (namely I^t) point to the objects in O^t, the agents in T^- point to objects outside of O^t. So these objects remain at step $t+1$. Let O^- be the set of these objects. Consistency of T' also implies that the objects in $O^t \cup O^-$ all point to different agents at step t; so objects in O^- all point to agents outside of I^t. Hence these agents are also still remaining at step $t+1$. Thus, every agent in T^- is still pointing to the same object at step $t+1$ as at step t, and this object is still pointing to the same agent at step $t+1$ as at step t. Thus, all the agents in T^- are potentially satisfied at step $t+1$. Moreover, the successor role function $s^{(t+1)+}$ agrees with s^{t+} on each of these agents.

So T^- is a nonempty, balanced initial subtableau of $T^{(t+1)+}[s^{(t+1)+}]$, and it consists entirely of agents who are potentially satisfied at step $t+1$. But the previous claim showed that such a subtableau cannot exist. This is a contradiction, proving the current claim. $\diamond$

Claim. T^t is the *unique* maximal balanced consistent initial subtableau of $T^{t+}[s^{t+}]$.

Proof. Suppose that there are two distinct balanced, consistent initial subtableaus of $T^{t+}[s^{t+}]$. By Lemma 4, their union is again a balanced, consistent initial subtableau. Hence, there can only be one maximal such subtableau. $\diamond$

This last claim implies the following: Once we know T^{t+}, the groups of agents I_k^{t+} and roles R_k^{t+}, and the successor role function s^{t+}, these data uniquely determine the subtableau T^t.

Now we can take a deep breath and then finish the proof of Part 3.

We treat $F(g)$ as known and g as unknown. We want to show that the sets I^t, O^t, R^t, for each t, are uniquely determined by $F(g)$. We do this by induction on t. So suppose that $I^{t'}, O^{t'}, R^{t'}$ have been determined for each $t' < t$. We want to show that there is a unique choice of I^t, O^t, R^t that is consistent with the tableau $F(g)$.

Since we know $I^{t'}, O^{t'}, R^{t'}$ for $t' < t$, these uniquely determine what the sets of remaining agents, objects, and priority roles (I^{t+}, O^{t+}, R^{t+}) are. Thus the sets of remaining agents and priority roles in each group (I_k^{t+}, R_k^{t+}) are also uniquely determined. So is the tableau of remaining agents, T^{t+}, since it is formed by simply taking $F(g)$ and deleting all the agents not in I^{t+}. And the successor role function s^{t+} is also uniquely determined: for each agent $i \in I^{t+}$, look at his favorite object in O^{t+}, and then that object's highest-priority role in R^{t+}, according to ϕ, must be $s^{t+}(i)$.

Since our last claim showed how the tableau T^t can be uniquely characterized in terms of these data, T^t is also uniquely determined.

This means that I^t is uniquely determined — it consists of the agents in T^t. And then O^t is uniquely determined: since each agent in I^t is assigned his favorite object in O^{t+}, O^t is exactly the set of these favorite objects. Then R^t is also determined in turn: each object in O^t was pointing to the agent assigned the highest-priority role in R^{t+}; since the roles thus picked out by O^t must be assigned to the agents removed at step t, they form the set R^t.

Thus, by induction on t, the sets I^t, O^t, R^t are uniquely determined. This completes the proof of Part 3.

Part 4: Showing that F is one-to-one. All that remains now is to put together the pieces. We want to show that the tableau $F(g) \in Tab$ uniquely determines $g \in Raf$. We saw in Part 3 that it uniquely determines the sets I^t, O^t, R^t; and these in turn uniquely determine the sets I^t_k, R^t_k. We also saw in the process that the successor role function $s^{t+} : I^{t+} \to R^{t+}$ is uniquely determined (hence so is its restriction $s^t = s^{t+}|_{I^t}$), and T^t is as well. In addition, after knowing all the I^t, O^t, R^t, we can reconstruct the object to which each remaining agent is pointing at step t, and whether or not that object's highest-priority role at step $t-1$ is still available at step t (if $t > 1$). Therefore, the sets S^t and U^t of satisfied and unsatisfied agents at each step are also uniquely determined.

But from the construction in Part 1, we have

$$g|_{R^t} = F^{-1}_{(I^t_1,\ldots,I^t_m;R^t_1,\ldots,R^t_m;s^t;S^t)}(T^t).$$

This is uniquely defined, since $F_{(I^t_1,\ldots,I^t_m;R^t_1,\ldots,R^t_m;s^t;S^t)}$ is a bijection.

So from $F(g)$, we can uniquely reconstruct R^t and $g|_{R^t}$ for each t, and thus we can reconstruct the entire function g. This shows that F is one-to-one, completing the proof.

□

4 On respecting partitions

The statement of Theorem 3 imposes that the priority framework ϕ should respect the partition of priority roles into groups. We now give an example to show that this condition in Theorem 3 is necessary.

In fact, when ϕ does not respect the partition of priority roles, random SDIG from ϕ is not even clearly defined. Recall that we first defined (random) SDIG from the priority *structure* $g(\phi)$. When ϕ respects the partition, we saw that SDIG from $g(\phi)$ (and given orders) does not depend on the choice of g, so we can talk unambiguously about random SDIG from ϕ. The following example with $n = 3$ agents and 3 objects shows that this is not the case in general. Let the partition of agents be $I_1 = \{i_1\}$, $I_2 = \{i_2, i_3\}$ (and correspondingly $R_1 = \{r_1\}$, $R_2 = \{r_2, r_3\}$). Let the agents' preferences over objects and the priority framework ϕ be

$$\begin{array}{llll} i_1: & o_1 \succ o_2 \succ o_3 & o_1: & r_3, r_1, r_2 \\ i_2: & o_1 \succ o_2 \succ o_3 & o_2: & r_2, r_3, r_1 \\ i_3: & o_2 \succ o_1 \succ o_3 & o_3: & r_2, r_3, r_1. \end{array}$$

This priority framework does not respect the partition of priority roles.

There are two possible choices of the role asssignment function g. Suppose we choose $g(r_j) = i_j$ for each j. Then straightforward calculations show that random SDIG from $g(\phi)$ results in allocations (o_3, o_1, o_2) and (o_1, o_3, o_2), each with probability 1/2. (Here the allocations are notated as in the examples in Subsections 2.1 and 2.2.)

On the other hand, what if we choose $g(r_1) = i_1$, $g(r_2) = i_3$, $g(r_3) = i_2$? Then one can check that random SDIG from $g(\phi)$ leads to (o_3, o_1, o_2) with probability 1.

This shows that random SDIG from ϕ is not even well-defined when ϕ does not respect the partition, since it depends on which g we use to map ϕ into a priority structure. Thus, Theorem 3 is not properly stated in this case.

One might hope to patch up the statement by redefining random SDIG as follows: Pick the orderings $f_1, \ldots, f_m$ uniformly at random, then form a priority structure π by sorting ϕ using the orderings, as in Subsection 2.2. That is, within the priority list of each object o, we assign the agents of each group I_k to the roles of group R_k so that these agents appear in the order given by f_k, thereby obtaining a priority list. (This can be done regardless of whether the roles of R_k appear consecutively in o's list.) Then perform TTC from this π. When ϕ respects the partition, we know this is equivalent to the original definition of random SDIG, so Theorem 3 holds. Does it still hold in general?

The answer is no. Once again, we can use the same counterexample ϕ. First consider calculating random TTC from ϕ. We have two choices for the role assignment function g: we can take $g(r_j) = i_j$ for each j, or take $g(r_1) = i_1$, $g(r_2) = i_3$, $g(r_3) = i_2$. Either way, TTC from $g(\phi)$ gives the allocation (o_3, o_1, o_2).

On the other hand, consider random SDIG under the new definition proposed above. If I_2 is ordered (i_3, i_2), then sorting leads to the priority structure

$$\begin{array}{ll} o_1: & i_3, i_1, i_2 \\ o_2: & i_3, i_2, i_1 \\ o_3: & i_3, i_2, i_1 \end{array}$$

and the resulting allocation is (o_1, o_3, o_2). Hence, random TTC from ϕ cannot give the same distribution over allocations as random SDIG from ϕ with the new sorting definition.

What goes wrong in the proof of Theorem 3 when ϕ does not respect the partition? This assumption on ϕ was needed in just one place, in Part 2 of the proof, during the second subclaim. We needed to show that throughout each step t of the modified TTC algorithm, each remaining object $o \in O_k^t$ was always pointing to an agent in I_k^t. This depended on knowing that the agents in I_k occurred consecutively in o's list in the sorted priority structure π: otherwise, if some of these agents had been removed at an early substep, o could be left pointing to an agent outside of I_k at a later substep.

5 Conclusion

Having gone through a fair amount of technical detail, we close this paper by recapitulating the main points of what we have accomplished.

We have considered mechanisms for allocating indivisible goods. We have presented a general equivalence theorem connecting randomized mechanisms based on the top trading cycles procedure and randomized mechanisms based on serial dictatorship. The basic version of our result, Theorem 1, says that for any priority framework ϕ, random TTC from ϕ is equivalent to random serial dictatorship. More verbally, this essentially says that if we take any TTC-based allocation mechanism and make it anonymous by permuting the agents uniformly at random, the result is equivalent to random serial dictatorship. This theorem directly implies all of the RSD equivalence results in the previously published literature [2, 19]. It can be interpreted as giving extra support to the use of RSD in applications, since it shows that many possible allocation mechanisms, once made anonymous, all become equivalent to RSD.

The full version of our result, Theorem 3, achieves greater generality by allowing the agents to be partitioned into groups and permuting uniformly at random within each group. For the partitioned setting, we have introduced the mechanism of serial dictatorship in groups, which can be seen as a generalization of serial dictatorship in which several dictators (the heads of different groups) may choose their favorite objects simultaneously. Theorem 3 states that any TTC mechanism (that respects the partition), with the agents randomly permuted within each group, is equivalent to SDIG with the agents randomly ordered in each group. This includes Theorem 1 as a special case, as well as the previous equivalence result [25] for house allocation with existing tenants.

In addition to unifying many existing theorems, our equivalence result is general enough to be potentially useful for future applications. We have presented several new examples to emphasize this point.

The paper also makes a couple of conceptual contributions to the study of mechanisms for allocating indivisible goods. One is the notion of priority frameworks, a device for making TTC allocation mechanisms anonymous. Another contribution is the notion of serial dictatorship in groups, which helps close the conceptual gap between traditional TTC and SD, and so helps us understand the equivalence results as invariance statements about the order in which agents get to choose favorite objects.

Finally, the method of proof we have used for Theorem 3 simplifies the bijective approach by injecting a dose of the enumerative approach. It arguably casts some new light on the equivalence, relative to existing bijective proofs for special cases, by showing that what matters is the decomposition of the set of all role assignments and the set of all tableaus into subsets that can be mapped to each other — without having to worry about exactly how the bijections between these subsets are constructed.

In spite of the relative generality of Theorem 3, this paper is not the final word on equivalences in allocation of indivisible goods. As mentioned earlier, the recent paper by Lee and Sethuraman [12] gives a further generalization. In particular, they cover a version of TTC in which priority frameworks take the form of trees (as in Pápai [15]) instead of lists, so that the highest-priority remaining agent for any given object may depend on the objects that were assigned to earlier agents at previous steps. It may well be possible to go even further. A natural target would be the full class of Pareto-efficient and group-strategyproof mechanisms recently characterized by Pycia and Ünver [20], which are similar to TTC but with a couple of possible variations. Preliminary work by Bade [5] looks to an equivalence result for this full class of mechanisms.

It also would be interesting to explore whether the techniques can say anything about more general kinds of allocation problems. For example, it is well understood that much of the elegant mathematical structure of the indivisible objects model breaks down once we venture beyond the simple case of one object per agent (see e.g. [16, 9, 11] for negative results in this area). If our methods can be partially extended to this broader class of models, they might then offer new insights that could help inform the choice of mechanisms for such problems.

A Appendix

Here we give the detailed discussion of slow TTC promised in Subsection 2.1.

Given the usual sets of agents and objects, and a priority structure π, define the *slow TTC* algorithm, whose execution is nondeterministic, as follows:

- At each step $t \geq 1$, each remaining agent points to his favorite remaining object, and each remaining object points to the highest-priority of the remaining agents. Choose at least one (and possibly more) of the cycles in the resulting graph. Assign each agent in any chosen cycle the object to which he points, and remove these agents and objects.

- Continue until all agents and objects are removed.

This definition makes sense: as in the original TTC algorithm, at least one cycle must form at each step.

Lemma 6 *The allocation produced by slow TTC is independent of the choice of which cycles to remove at each step.*

Proof: First note that for any execution of slow TTC in which several cycles are removed at once, we can separate this step into removing these same cycles one by one, without any other cycles intervening; this change in the execution certainly has no effect on the resulting allocation. So it suffices to prove the lemma under the added restriction on slow TTC that only one cycle is removed at each step.

Suppose this version of the lemma is false. Consider a counterexample for which the number n of agents is minimal. Let $C^1, C^2, \ldots, C^{\bar{t}}$ describe one execution of the (restricted) slow TTC algorithm, with C^t denoting the cycle of agents and objects removed at step t; let $\widetilde{C}^1, \widetilde{C}^2, \ldots, \widetilde{C}^{\tilde{\bar{t}}}$ describe a different execution that leads to a different allocation of objects to agents.

If $C^1 = \widetilde{C}^1$, then the remaining sequences, $C^2, \ldots, C^{\bar{t}}$ and $\widetilde{C}^2, \ldots, \widetilde{C}^{\tilde{\bar{t}}}$, must lead to different allocations of the remaining objects to the remaining agents. But this contradicts the initial choice of a minimal counterexample. Hence $C^1 \neq \widetilde{C}^1$. Now consider the execution $\widetilde{C}^1, \ldots, \widetilde{C}^{\tilde{\bar{t}}}$ of slow TTC. The cycle C^1 is present at the beginning of the algorithm. But a cycle, once formed, continues to exist until some agent or object in the cycle is removed, at which point the entire cycle must be removed. Therefore, C^1 must be equal to one of the cycles $\widetilde{C}^1, \ldots, \widetilde{C}^{\tilde{\bar{t}}}$. Let $\tilde{t}$ be such that $C^1 = \widetilde{C}^{\tilde{t}}$. Then, removing $\widetilde{C}^{\tilde{t}}$ at the beginning does not interfere with the successive formation of the cycles $\widetilde{C}^1, \ldots, \widetilde{C}^{\tilde{t}-1}$, since none of them uses any agent or object in $\widetilde{C}^{\tilde{t}}$. So there exists an execution of slow TTC in which the cycles removed at successive steps are

$$\widetilde{C}^{\tilde{t}}, \widetilde{C}^1, \widetilde{C}^2, \ldots, \widetilde{C}^{\tilde{t}-1}, \widetilde{C}^{\tilde{t}+1}, \ldots, \widetilde{C}^{\tilde{\bar{t}}}.$$

This sequence leads to the same allocation as $\widetilde{C}^1, \dots, \widetilde{C}^{\widetilde{\bar{t}}}$, which is a different allocation than that produced by $C^1, \dots, C^{\bar{t}}$. But now we have two executions of slow TTC, starting with the same cycle $C^1 = \widetilde{C}^{\widetilde{\bar{t}}}$, and leading to different allocations. We already showed that this contradicts the minimality of n. This contradiction proves the lemma. □

Lemma 6 ensures, in particular, that slow TTC always produces the same allocation as TTC (since the latter is a special case of the former).

References

[1] Atila Abdulkadiroğlu and Yeon-Koo Che (2010), "The Role of Priorities in Assigning Indivisible Objects: A Characterization of Top Trading Cycles," unpublished, Columbia University.

[2] Atila Abdulkadiroğlu and Tayfun Sönmez (1998), "Random Serial Dictatorship and the Core from Random Endowments in House Allocation Problems," *Econometrica* 66 (3), 689–701.

[3] Atila Abdulkadiroğlu and Tayfun Sönmez (1999), "House Allocation with Existing Tenants," *Journal of Economic Theory* 88 (2), 233–260.

[4] Atila Abdulkadiroğlu and Tayfun Sönmez (2003), "School Choice: A Mechanism Design Approach," *American Economic Review* 93 (3), 729–747.

[5] Sophie Bade (2013), "Random Serial Dictatorship: The One and Only," unpublished, Max Planck Institute for Research on Collective Goods.

[6] Anna Bogomolnaia and Hervé Moulin (2001), "A New Solution to the Random Assignment Problem," *Journal of Economic Theory* 100 (2), 295–328.

[7] Lars Ehlers and Bettina Klaus (2004), "Resource-Monotonicity for House Allocation Problems," *International Journal of Game Theory* 32 (4), 545–560.

[8] Özgün Ekici (2011), "Fair and Efficient Discrete Resource Allocation: A Market Approach," unpublished, Özyeğin University.

[9] John William Hatfield (2009), "Strategy-proof, Efficient, and Nonbossy Quota Allocations," *Social Choice and Welfare* 33 (3), 505–515.

[10] Donald E. Knuth (1996), "An Exact Analysis of Stable Allocation," *Journal of Algorithms* 20 (2), 431–442.

[11] Fuhito Kojima (2009), "Random Assignment of Multiple Indivisible Objects," *Mathematical Social Sciences* 57 (1), 134–142.

[12] Thiam Lee and Jay Sethuraman (2011), "Equivalence Results in the Allocation of Indivisible Objects: A Unified View," unpublished, Columbia University.

[13] Jinpeng Ma (1994), "Strategy-Proofness and the Strict Core in a Market with Indivisibilities," *International Journal of Game Theory* 23 (1), 75–83.

[14] Thayer Morrill (2013), "An Alternative Characterization of Top Trading Cycles," *Economic Theory* 54 (1), 181–197.

[15] Szilvia Pápai (2000), "Strategyproof Assignment by Hierarchical Exchange," *Econometrica* 68 (6), 1403–1433.

[16] Szilvia Pápai (2001), "Strategyproof and Nonbossy Multiple Assignments," *Journal of Public Economic Theory* 3 (3), 257–271.

[17] Szilvia Pápai (2010), "Matching with Minimal Priority Rights," in preparation, Concordia University.

[18] Parag A. Pathak (2008), "Lotteries in Student Assignment: The Equivalence of Queueing and a Market-Based Approach," unpublished, MIT.

[19] Parag A. Pathak and Jay Sethuraman (2011), "Lotteries in Student Assignment: An Equivalence Result," *Theoretical Economics* 6 (1), 1–17.

[20] Marek Pycia and M. Utku Ünver (2011), "Incentive Compatible Allocation and Exchange of Discrete Resources," Boston College Working Papers in Economics, #715.

[21] Alvin E. Roth (1982), "Incentive Compatibility in a Market with Indivisible Goods," *Economics Letters* 9 (2), 127–132.

[22] Alvin E. Roth and Andrew Postlewaite (1977), "Weak Versus Strong Domination in a Market with Indivisible Goods," *Journal of Mathematical Economics* 4 (2), 131–137.

[23] Alvin E. Roth, Tayfun Sönmez, and M. Utku Ünver (2004), "Kidney Exchange," *Quarterly Journal of Economics* 119 (2), 457–488.

[24] Lloyd Shapley and Herbert Scarf (1974), "On Cores and Indivisibility," *Journal of Mathematical Economics* 1 (1), 23–37.

[25] Tayfun Sönmez and M. Utku Ünver (2005), "House Allocation with Existing Tenants: An Equivalence," *Games and Economic Behavior* 52 (1), 153–185.

[26] Lars-Gunnar Svensson and Bo Larsson (2005), "Strategy-Proofness, Core, and Sequential Trade," *Review of Economic Design* 9 (2), 167–190.