Abstract

Classic cake cutting protocols — which fairly allocate a divisible good among agents with heterogeneous preferences — are susceptible to manipulation. Do their strategic outcomes still guarantee fairness? To answer this question we adopt a novel algorithmic approach, proposing a concrete computational model and reasoning about the game-theoretic properties of algorithms that operate in this model. Specifically, we show that each protocol in the class of generalized cut and choose (GCC) protocols — which includes the most important discrete cake cutting protocols — is guaranteed to have approximate subgame perfect Nash equilibria. Moreover, we observe that the (approximate) equilibria of proportional protocols — which guarantee each of the $n$ agents a $1/n$-fraction of the cake — must be (approximately) proportional, and design a GCC protocol where all Nash equilibrium outcomes satisfy the stronger fairness notion of envy-freeness. Finally, we show that under an obliviousness restriction, which still allows the computation of approximately envy-free allocations, GCC protocols are guaranteed to have exact subgame perfect Nash equilibria.
1 Introduction

A large body of literature deals with the so-called cake cutting problem — a misleadingly childish metaphor for the challenging and important task of fairly dividing a heterogeneous divisible good between multiple agents (see, e.g., the recent survey by Procaccia [20], and the books by Brams and Taylor [2] and Robertson and Webb [21]). Two formal notions of fairness have emerged as the most appealing and well-studied: proportionality, in which each of the $n$ agents receives at least a $1/n$-fraction of the entire cake according to its valuation; and envy-freeness, which stipulates that no agent would want to swap its own piece with that of another agent. At the heart of the cake cutting endeavor is the design of cake cutting protocols, which specify an interaction between agents — typically via iterative steps of manipulating the cake — such that the final allocation is guaranteed to be proportional or envy-free.

The simplest cake cutting protocol is known as cut and choose, and is designed for the case of two agents. The first agent cuts the cake into two pieces that it values equally; the second agent then chooses the piece that it prefers, leaving the first agent with the remaining piece. It is easy to see that this protocol yields a proportional and envy-free allocation (in fact these two notions coincide when there are only two agents). However, taking the game-theoretic point of view, it is immediately apparent that the agents can often do better by disobeying the protocol when they know each other’s valuations. For example, in the cut and choose protocol, assume that the first agent only desires a specific small piece of cake, whereas the second agent uniformly values the cake. The first agent can obtain its entire desired piece, instead of just half of it, by carving that piece out.

So how would strategic agents behave when faced with the cut and choose protocol? A standard way of answering this question employs the notion of Nash equilibrium: each agent would use a strategy that is a best response to the other agent’s strategy. It turns out that the cut and choose protocol has a unique Nash equilibrium: the first agent cuts two pieces that the second agent values equally; the second agent selects its more preferred piece, and the one less preferred by the first agent in case of a tie. Clearly, the second agent cannot gain from deviating, as it is selecting a piece that is at least as preferred as the other. As for the first agent, if it makes its preferred piece even bigger, the second agent would choose that piece, making the first agent worse off. Interestingly enough, in equilibrium the roles of the agents are reversed; now it is the second agent who is getting exactly half of its value for the whole cake, while the first agent generally gets more. Crucially, the equilibrium outcome is also proportional and envy-free. In other words, even though the agents are strategizing rather than following the protocol, the outcome in equilibrium has the same fairness properties as the “honest” outcome!

With this motivating example in mind, we would like to make general statements regarding the equilibria of cake cutting protocols. We wish to identify a general family of cake cutting protocols — which captures the classic cake cutting protocols — so that each protocol in the family is guaranteed to possess (approximate) equilibria. Moreover, we wish to argue that these equilibrium outcomes are fair. Ultimately, our goal is to be able to reason about cake divisions that are obtained as outcomes when agents are presented with a standard cake cutting protocol and behave strategically; ideally we would like to be able to argue that these cake divisions must nevertheless be fair.

1 Assuming there are pieces of cake that the two agents value differently, and every piece has positive value.
1.1 Model, Results, and a New Algorithmic Paradigm

To set the stage for a result that encompasses classic cake cutting protocols, we introduce (in Section 2) the class of generalized cut and choose (GCC) protocols. A GCC protocol is represented by a tree, where each node is associated with the action of an agent. There are two types of nodes: a cut node, which instructs the agent to make a cut between two existing cuts; and a choose node, which offers the agent a choice between a collection of pieces that are induced by existing cuts. Moreover, we assume that the progression from a node to one of its children depends only on the relative positions of the cuts (in a sense to be explained formally below). We argue that classic protocols — such as Dubins-Spanier [6], Selfridge-Conway (see [21]), Even-Paz [9], as well as the original cut and choose protocol — are all GCC protocols. We view the definition of the class of GCC protocols as one of our main contributions.

In Section 3, we observe that GCC protocols may not have exact Nash equilibria (NE). However, we prove that every GCC protocol has at least one \( \varepsilon \)-NE for every \( \varepsilon > 0 \), in which agents cannot gain more than \( \varepsilon \) by deviating, and \( \varepsilon \) can be chosen to be arbitrarily small. In fact, we establish this result for a stronger equilibrium notion, \((\text{approximate})\) subgame perfect Nash equilibrium (SPNE), which is, intuitively, a strategy profile where the strategies are in NE even if the game starts from an arbitrary point. We also observe that for any proportional protocol, the outcome in any \( \varepsilon \)-equilibrium must be an \( \varepsilon \)-proportional division. We conclude that under the classic cake cutting protocols listed above — which are all proportional — strategic behavior (almost) preserves the proportionality of the outcome.

The natural question at this point is whether there are GCC protocols with envy-free equilibria. We give an affirmative answer in Section 4. Specifically, we design a curious GCC protocol in which every NE outcome is a contiguous envy-free allocation and vice versa, that is, the set of NE outcomes coincides with the set of contiguous envy-free allocations. It remains open whether a similar result can be obtained for SPNE instead of NE.

In Section 5 we establish an existence result for exact SPNE by focusing on oblivious GCC protocols, where the available actions at each step are independent of the history. While the obliviousness property is restrictive, we demonstrate that oblivious GCC protocols are powerful enough to compute \( \varepsilon \)-envy-free allocations.

Taking a broader perspective, our approach involves introducing a concrete computational model that captures well-known algorithms, and reasoning about the game-theoretic guarantees of all algorithms operating in this model. This approach appears distinct from related ones, where concrete query models are defined in order to evaluate the computational complexity of economic methods [1][12], or restrictions on the output of the algorithm — such as the well-known maximal-in-range restriction [5] — give rise to desirable game-theoretic properties. Perhaps the most closely related approach was taken by Tennenholtz in his work on program equilibrium [24] (later extended by Fortnow [10]), but there the strategies are the programs themselves, whereas in our work a common algorithm induces the agents’ strategies. We therefore believe that our conceptual contributions may be of independent interest to researchers working in auction design and other areas of algorithmic game theory.

1.2 Related Work on Incentives in Cake Cutting

Nicolò and Yu [18] were the first to suggest equilibrium analysis for cake cutting protocols. Focusing exclusively on the case of two agents, they design a specific cake cutting protocol whose unique
SPNE outcome is envy-free. And while the original cut and choose protocol also provides this guarantee, it is not “procedural envy free” because the cutter would like to exchange roles with the chooser; the two-agent protocol of Nicoló and Yu aims to solve this difficulty. Brânzei and Miltersen \[3\] also investigate equilibria in cake cutting, but in contrast to our work they focus on one cake cutting protocol — the Dubins-Spanier protocol — and restrict the space of possible strategies to \textit{threshold strategies}. Under this assumption, they characterize NE outcomes, and in particular they show that in NE the allocation is envy-free. Brânzei and Miltersen also prove the existence of \(\varepsilon\)-equilibria that are \(\varepsilon\)-envy-free; again, this result relies on their strong restriction of the strategy space.

Several recent papers by computer scientists \[4, 17, 16\] take a mechanism design approach to cake cutting; their goal is to design fair cake cutting protocols that are \textit{strategyproof}, in the sense that agents can never benefit from manipulating the protocol. This turns out to be an almost impossible task \[26\], and positive results are obtained by either making extremely strong assumptions (agents’ valuations are highly structured and fully reported to a central authority), or by employing randomization and significantly weakening the desired properties. In contrast, our result of Section 3 deals with strategic outcomes under a large class of cake cutting protocols, and aims to capture well-known protocols; our result of Section 4 is a positive result that achieves fairness “only” in equilibrium, but without imposing any restrictions on the agents’ valuations.

1.3 Main Open Problem

We show that oblivious GCC protocols are guaranteed to have exact Nash equilibria. However, this is not true for GCC protocols in general, and the enigmatic question of whether classic cake cutting protocols possess exact Nash equilibria and, if so, whether the corresponding cake divisions are envy-free, remains open (because these protocols do not satisfy the obliviousness restriction). Note that the work of Brânzei and Miltersen \[3\] does not answer this question even for the Dubins-Spanier protocol, as they strongly restrict the strategy space to the subset of threshold strategies, and therefore their equilibrium results do not capture deviations to non-threshold strategies. Our model provides the formal framework that is required to tackle this question in its most general — and presumably extremely challenging — form.

2 The Model

The cake cutting literature typically represents the cake as the interval \([0, 1]\). There is a set of agents \(N = \{1, \ldots, n\}\), and each agent \(i \in N\) is endowed with a valuation function \(V_i\) that assigns a value to every subinterval of \([0, 1]\). These values are induced by a non-negative continuous value density function \(v_i\), so that for an interval \(I\), \(V_i(I) = \int_{x \in I} v_i(x)\, dx\). By definition, \(V_i\) satisfies the first two properties below; the third is an assumption that is made without loss of generality.

1. Additivity: For every two disjoint intervals \(I_1\) and \(I_2\), \(V_i(I_1 \cup I_2) = V_i(I_1) + V_i(I_2)\).

2. Divisibility: For every interval \(I \subseteq [0, 1]\) and \(0 \leq \lambda \leq 1\) there is a subinterval \(I' \subseteq I\) such that \(V_i(I') = \lambda V_i(I)\).

3. Normalization: \(V_i([0, 1]) = 1\).
Note that the valuation functions are non-atomic, i.e., they assign zero value to points. This allows us to disregard the boundaries of intervals, and in particular we treat intervals that overlap at their boundary as disjoint. We sometimes explicitly assume that the valuations are strictly positive, that is, \( v_i(x) > 0 \) for all \( x \in [0, 1] \) and for all \( i \in N \).

A piece of cake is a finite union of disjoint intervals. We are interested in allocations of disjoint pieces of cake \( X_1, \ldots, X_n \), where \( X_i \) is the piece that is allocated to agent \( i \in N \). A piece is contiguous if it consists of a single interval.

We study two fairness notions. An allocation \( X \) is proportional if for all \( i \in N \), \( V_i(X_i) \geq 1/n \); and envy-free if for all \( i, j \in N \), \( V_i(X_i) \geq V_i(X_j) \). Note that envy-freeness implies proportionality.

### 2.1 Generalized Cut and Choose Protocols

The standard communication model in cake cutting was proposed by Robertson and Webb [21], and was employed in a body of work studying the complexity of cake cutting [8, 7, 25, 19, 14]. The model restricts the interaction between the protocol and the agents to two types of queries:

- **Cut query**: \( \text{Cut}_i(x, \alpha) \) asks agent \( i \) to return a point \( y \) such that \( V_i([x, y]) = \alpha \).

- **Evaluate query**: \( \text{Evaluate}_i(x, y) \) asks agent \( i \) to return a value \( \alpha \) such that \( V_i([x, y]) = \alpha \).

However, the communication model does not give much information about the actual implementation of the protocol and what allocation it produces. For example, the protocol could allocate pieces depending on whether a particular cut was made at an irrational point (see Algorithm 2). For this reason, we define a generic class of protocols that are implementable with natural operations, which capture all the bounded cake cutting algorithms, such as cut and choose, Dubins-Spanier, Even-Paz, Successive-Pairs, and Selfridge-Conway (see, e.g., [20]). At a high level, the standard protocols are implemented using a sequence of natural instructions, each of which is either a Cut operation, in which some agent is asked to make a cut in a specified region of the cake; or a Choose operation, in which some agent is asked to take a piece from a set of already demarcated pieces indicated by the protocol. In addition, every node in the decision tree of the protocol is based exclusively on the execution history and absolute ordering of the cut points, which can be verified with any of the following operators: \(<, \leq, =, \geq, >\).

More formally, a generalized cut and choose (GCC) protocol is implemented exclusively with the following types of instructions:

- **Cut**: The syntax is “\( i \) Cuts in \( S \)”, where \( S = \{[x_1, y_1], \ldots, [x_m, y_m]\} \) is a set of contiguous pieces (intervals), such that the endpoints of every piece \([x_j, y_j]\) are 0, 1, or cuts made in the previous steps of the protocol. Agent \( i \) can make a cut at any point \( z \in [x_j, y_j] \), for some \( j \in \{1, \ldots, m\} \).

- **Choose**: The syntax is “\( i \) Chooses from \( S \)”, where \( S = \{[x_1, y_1], \ldots, [x_m, y_m]\} \) is a set of contiguous pieces, such that the endpoints of every piece \([x_j, y_j]\) in \( S \) are 0, 1, or cuts made in the previous steps of the protocol. Agent \( i \) can choose any single piece \([x_j, y_j]\) from \( S \).

- **If-Else Statements**: The conditions depend on the absolute order of all the cut points made in the previous steps and the execution history of the protocol.

\(^2\)In the sense that the number of operations is upper-bounded by a function that takes the number of agents \( n \) as input.
A GCC protocol uniquely identifies every contiguous piece by the symbolic names of all the cut points contained in it. For example, Algorithm 1 is a GCC protocol. Algorithm 2 is not a GCC protocol, because it verifies that the point where agent 1 made a cut is exactly 1/3, whereas a GCC protocol can only verify the ordering of the cut points relative to each other and the endpoints of the cake. Note that, unlike in the communication model of Robertson and Web [21], GCC protocols cannot obtain and use information about the valuations of the agents — the allocation is only decided by the agents’ Choose operations.

To gain some intuition we illustrate why the discrete version of Dubins-Spanier belongs to the class of GCC protocols. The first round of the algorithm asks each agent $i$ to make a cut at any point in $[0, 1]$ — denoted by $x_i^1$ — and allocates the first piece, $[0, x_i^1]$ to the agent $i$ that made the leftmost cut. This admits a GCC implementation as follows. First, each agent $i$ is required to make a cut in $\{[0, 1]\}$, at some point denoted by $x_i^1$. The agent $i^*$ with the leftmost cut $x_i^1$, can be determined using if-else statements whose conditions only depend on the ordering of the cut points $x_1^1, \ldots, x_n^1$. Then, agent $i^*$ is asked to choose “any” piece in the singleton set $\{[0, x_i^1]\}$. The subsequent rounds are similar: at the end of every round the agent that was allocated a piece is removed, and the protocol iterates on the remaining players and remaining cake. Note that, while the Dubins-Spanier protocol allocates one contiguous piece to each agent, other protocols such as Selfridge-Conway can assign non-contiguous pieces. In general, the GCC implementations of these protocols construct their decision trees by looking at both the ordering of the cut points and the allocation history.

2.2 The Game

We study GCC protocols when the agents behave strategically. Specifically, we consider a GCC protocol, coupled with the valuation functions of the agents, as an extensive-form game of perfect information (see, e.g., [22]). In such a game, agents execute the Cut and Choose instructions strategically. Each agent is fully aware of the valuation functions of the other agents and aims to optimize its overall utility for the chosen pieces, given the strategies of other agents. The game can be represented by a tree (called a game tree) with Cut and Choose nodes:

- In a Cut node defined by “$i$ cuts in $S$”, where $S = \{[x_1, y_1], \ldots, [x_m, y_m]\}$, the strategy space
of agent $i$ is the set $S$ of points where agent $i$ can make a cut at this step.

- In a Choose node defined by “$i$ chooses from $S$”, where $S = \{[x_1, y_1], \ldots, [x_m, y_m]\}$, the strategy space is the set $\{1, \ldots, m\}$, i.e., the indices of the pieces that can be chosen by the agent from the set $S$.

The strategy of an agent defines an action for each node of the game tree where it executes a Cut or a Choose operation. If an agent deviates, the game can follow a completely different branch of the tree, but the outcome will still be well-defined.

The strategies of the agents are in Nash equilibrium (NE) if no agent can improve its utility by unilaterally deviating from its current strategy, i.e., by cutting at a different set of points and/or by choosing different pieces. A subgame perfect Nash equilibrium (SPNE) is a stronger equilibrium notion, which means that the strategies are in NE in every subtree of the game tree. In other words, even if the game started from an arbitrary node of the game tree, the strategies would still be in NE. An $\epsilon$-NE (resp., $\epsilon$-SPNE) is a relaxed solution concept where an agent cannot gain more than $\epsilon$ by deviating (resp., by deviating in any subtree).

### 3 Existence of Approximate Equilibria

It is well-known that finite extensive-form games of perfect information can be solved using backward induction: starting from the leaves and progressing towards the root, at each node the relevant agent chooses an action that maximizes its utility, given the actions that were computed for the node’s children. The induced strategies form an SPNE. Unfortunately, although we consider finite GCC protocols, we also need to deal with Cut nodes where the action space is infinite, hence naïve backward induction does not apply.

In fact, it turns out that not every GCC protocol admits an exact NE. For example, consider Algorithm 1, and assume that the valuation function of agent 1 is strictly positive. Assume there exists a NE where agent 1 cuts at $x^*, y^*, z^*$, respectively, and chooses the piece $[x^*, y^*]$. If $x^* > 0$, then the agent can improve its utility by making the first cut at $x' = 0$ and choosing the piece $[x', y^*]$, since $V_1([x', y^*]) > V_1([x^*, y^*])$. Thus, $x^* = 0$. Moreover, it cannot be the case that $y^* = 1$, since the agent only receives an allocation if $y^* < z^* \leq 1$. Thus, $y^* < 1$. Then, by making the second cut at any $y' \in (y^*, z^*)$, agent 1 can obtain the value $V_1([0, y']) > V_1([0, y^*])$. It follows that there is no exact NE where the agent chooses the first piece. Similarly, it can be shown that there is no exact NE where the agent chooses the second piece, $[y^*, z^*]$. This illustrates why backward induction does not apply: the maximum value at some Cut nodes may not be well defined. Nevertheless, we are able to prove a strong positive result.

**Theorem 3.1.** Every GCC protocol with a bounded number of steps has an $\epsilon$-SPNE, for every $\epsilon > 0$.

The high-level idea of our proof, which appears in Appendix A, relies on discretizing the cake — such that every cell in the resulting grid has a very small value for each agent — and computing the optimal outcome on the discretized cake using backward induction. At every cut step of the protocol, the grid is refined by adding a point between every two consecutive points of the grid from the previous cut step. This ensures that any ordering of the cut points that can be enforced by playing on the continuous cake can also be enforced on the discretized instance. So, for the purpose of computing an approximate SPNE, it is sufficient to work with the discretized. Then,
we show that the backward induction outcome from the discrete game gives an $\varepsilon$-SPNE on the continuous cake.

4 Fair Equilibria

The existence of approximate equilibria gives us a tool for predicting the strategic outcomes of cake cutting protocols. In particular, classic protocols provide fairness guarantees when agents act honestly; but do they provide any fairness guarantees in equilibrium?

We first make a simple yet important observation. In a proportional protocol, every agent is guaranteed a value of at least $1/n$ regardless of what the others are doing. Therefore, in every NE (if any) of the protocol, the agent still receives a piece worth at least $1/n$; otherwise it can deviate to the strategy that guarantees it a utility of $1/n$ and do better. Similarly, an $\varepsilon$-NE must be $\varepsilon$-proportional, i.e., each agent must receive a piece worth at least $1/n - \varepsilon$. Hence, classic protocols such as Dubins-Spanier, Even-Paz, and Selfridge-Conway guarantee (approximately) proportional outcomes in any (approximate) NE (and of course this observation carries over to the stronger notion of SPNE).

We do not know whether an analogous general statement holds for envy-freeness (except for the case of two agents, where the two fairness notions coincide). However, our second result is the design of a specific GCC protocol that forces strategic agents to compute envy-free allocations in every NE. As we shall see, guaranteeing envy-free equilibria for any number of agents is quite nontrivial; in fact it remains open whether an analogous result holds when one asks for SPNE rather than just NE.

**Theorem 4.1.** There exists a GCC protocol $P$ such that on every cake cutting instance with strictly positive valuation functions, an allocation $X$ is the outcome of a NE of $P$ if and only if $X$ is an envy-free contiguous allocation that contains the entire cake.

Crucially, an envy-free contiguous allocation is guaranteed to exist [23], hence the set of NE of protocol $P$ is nonempty. The proof of the theorem uses the Thieves Protocol given by Algorithm 3. In this protocol, agent 1 first demarcates a contiguous allocation $X = \{X_1, ..., X_n\}$ of the entire cake, where $X_i$ is a contiguous piece that corresponds to agent $i$. This can be implemented as follows. First, agent 1 makes $n$ cuts such that the $i$-th cut is interpreted as the left endpoint of $X_i$. The left endpoint of the leftmost piece is reset to 0 by the protocol. Then, the rightmost endpoint of $X_i$ is naturally the leftmost cut point to its right or 1 if no such point exists. Ties among overlapping cut points are resolved in favor of the agent with the smallest index; the corresponding cut point is assumed to be the leftmost one. Notice that every allocation that assigns nonempty contiguous pieces to all agents can be demarcated in this way.

After the execution of the demarcation step, $X$ is only a tentative allocation. Then, the protocol enters a verification round, where each agent $i$ is allowed to steal some non-empty strict subset of a piece (say, $X_j$) demarcated for another agent. If this happens (i.e., the if-condition is true) then agent $i$ takes the stolen piece and the remaining agents get nothing. This indicates the failure of the verification and the protocol terminates. Otherwise, the pieces of $X$ are eventually allocated to the agents, i.e., agent $i$ takes $X_i$.

The proof of Theorem 4.1 uses two important characteristics of the protocol. First, it guarantees that no state in which some agent steals can be a NE; this agent can always steal an even more valuable piece. Second, stealing is beneficial for an envious agent.
Agent 1 demarcates a contiguous allocation $X$ of the cake.

for $i = 2, \ldots, n, 1$ do

// Verification of envy-freeness for agent $i$
Agent $i$ Cuts in $\{[0,1]\}$ // @ $w_i$
Agent $i$ Cuts in $\{[w_i, 1]\}$ // @ $z_i$

for $j = 1$ to $n$ do

if $\emptyset \neq ([w_i, z_i] \cap X_j) \subseteq X_j$ then

// Agent $i$ steals a non-empty strict subset of $X_j$
Agent $i$ Chooses from $\{[w_i, z_i] \cap X_j\}$
exit // Verification failed: protocol terminates

end if

end for

// Verification successful for agent $i$
end for

for $i = 1$ to $n$ do

Agent $i$ Chooses from $\{X_i\}$

end for

Algorithm 3: Thieves Protocol: Every NE induces a contiguous envy-free allocation that contains the entire cake and vice versa.

Proof of Theorem 4.1. Let $\mathcal{P}$ be the Thieves protocol given by Algorithm 3 and $\mathcal{E}$ be any NE of $\mathcal{P}$. Denote by $X$ the contiguous allocation of the entire cake obtained during the demarcation step, where $X_i = [x_i, y_i]$ for all $i \in N$, and let $w_i$ and $z_i$ be the cut points of agent $i$ during its verification round. Assume for the sake of contradiction that $X$ is not envy-free. Let $k^*$ be an envious agent, where $V_{k^*}(X_{j^*}) > V_{k^*}(X_{k^*})$, for some $j^* \in N$. There are two cases to consider:

Case 1: Each agent $i$ receives the piece $X_i$ in $\mathcal{E}$. This means that, during its verification round, each agent $i$ selects its cut points from the set $\bigcup_{j=1}^n \{x_j, y_j\}$. By the non-envy-freeness condition for $X$ above (and by the fact that the valuation function $V_{k^*}$ is strictly positive), there exist $w_{k^*}', z_{k^*}'$ such that $x_{j^*} < w_{k^*}' < z_{k^*}' < y_{j^*}$ and $V_{k^*}([w_{k^*}', z_{k^*}']) > V_{k^*}([x_{k^*}, y_{k^*}])$. Thus, agent $k^*$ could have been better off by cutting at points $w_{k^*}'$ and $z_{k^*}'$ in its verification round, contradicting the assumption that $\mathcal{E}$ is a NE.

Case 2: There exists an agent $i$ that did not receive the piece $X_i$. Then, it must be the case that some agent $k$ stole a non-empty strict subset $[w_k'', z_k''] = [w_k, z_k] \cap X_j$ of another piece $X_j$. However, agent $k$ could have been better off at the node in the game tree reached in its verification round by making the following marks: $w_k' = \frac{x_j + w_k''}{2}$ and $z_k' = \frac{z_k'' + y_j}{2}$. Since either $x_j \leq w_k'' < z_k'' \leq y_j$ (recall that $[w_k'', z_k'']$ is a non-empty strict subset of $X_j$ and the valuation function $V_k$ is strictly positive), it is also true that $V_k([w_k', z_k']) > V_k([w_k'', z_k''])$, again contradicting the assumption that $\mathcal{E}$ is a NE.

So, the allocation computed by agent 1 under every NE $\mathcal{E}$ is indeed envy-free; this completes the proof of the first part of the theorem.

We next show that every contiguous envy-free allocation of the entire cake is the outcome of a NE. Let $Z$ be such an allocation, with $Z_i = [x_i, y_i]$ for all $i \in N$. We define the following set of strategies $\mathcal{E}$ for the agents:

- At every node of the game tree (i.e., for every possible allocation that could be demarcated
by agent 1), agent $i \geq 2$ cuts at points $w_i = x_i$ and $z_i = y_i$ during its verification round.

- Agent 1 specifically demarcates the allocation $Z$ and cuts at points $w_1 = x_1$ and $z_1 = y_1$ during its verification round.

Observe that $[w_i, z_i] \cap Z_j$ is either empty or equal to $Z_j$ for every pair of $i,j \in N$. Hence, the verification phase is successful for every agent and agent $i$ receives the piece $Z_i$.

We claim that this is a NE. Indeed, consider a deviation of agent 1 to a strategy that consists of the demarcated allocation $Z'$ (and the cut points $w'_1$ and $z'_1$). First, assume that the set of pieces in $Z'$ is different from the set of pieces in $Z$. Then, there is some agent $k \neq 1$ and some piece $Z'_j$ such that the if-condition $\emptyset \subset [x_k, y_k] \cap Z'_j \subset Z'_j$ is true. Hence, the verification round would fail for some agent $i \in \{2, ..., k\}$ and agent 1 would receive nothing. So, both $Z'$ and $Z$ contain the same pieces, and may differ only in the way these pieces are tentatively allocated to the agents. But in this case the maximum utility agent 1 can get is $\max_j V_1(Z'_j)$, either by keeping the piece $Z'_1$ or by stealing a strict subset of some other piece $Z'_j$. Due to the envy-freeness of $Z$, we have: $\max_j V_1(Z'_j) = \max_j V_1(Z_j) = V_1(Z_1)$, hence, the deviation is not profitable in this case either.

Now, consider a deviation of agent $i \geq 2$ to a strategy that consists of the cut points $w'_i$ and $z'_i$. If both $w'_i$ and $z'_i$ belong to $\bigcup_{j=1}^n \{x_i, y_i\}$, then $[w'_i, z'_i] \cap Z_j$ is either empty or equal to $Z_j$ for some $j \in N$. Hence, the deviation will leave the allocation unaffected and the utility of agent $i$ will not increase. If instead one of the cut points $w'_i$ and $z'_i$ does not belong to $\bigcup_{j=1}^n \{x_i, y_i\}$, this implies that the condition $\emptyset \subset [w'_i, z'_i] \cap Z_j \subset Z_j$ is true for some $j \in N$, i.e., agent $i$ will steal the piece $[w'_i, z'_i] \cap Z_j$. However, the utility $V_i([w'_i, z'_i] \cap Z_j)$ of agent $i$ cannot be greater than $V_i(Z_j)$, which is at most $V_i(Z_i)$ due to the envy-freeness of $Z$. Hence, again, this deviation is not profitable for agent $i$.

We conclude that $E$ is a NE; this completes the proof of the theorem.

The Thieves Protocol is only practical insofar as it delegates the computation of the envy-free allocation from the center to the agents (specifically, to agent 1). Its shortcoming is that agent 1 must compute the entire envy-free allocation up-front. Nevertheless, we view this as a strong positive result à la implementation theory (see, e.g., [13]), which aims to construct (typically impractical) games where the NE outcomes coincide with a given specification of acceptable outcomes for each constellation of agents’ preferences (known as a social choice correspondence). Our construction guarantees that the NE outcomes coincide with (contiguous) envy-free allocations, that is, in this case the envy-freeness criterion specifies which outcomes are acceptable. We find it highly encouraging that there exist GCC protocols with this property.

## 5 Obliviousness and Exact Equilibria

In Section 3 we noted that GCC protocols may not have exact Nash equilibria. We therefore ask whether there is a reasonable set of algorithmic restrictions that give rise to a family of cake-cutting protocols that have exact equilibria. For the set of restrictions to be “reasonable”, we would need to show that there are interesting protocols satisfying the required properties.

Going back to the equilibrium-less GCC protocol — Algorithm 1 — we observe that by removing the if statement we would circumvent the lack of equilibrium. Leveraging this observation, we say that a GCC protocol is oblivious if all of the nodes at depth $d$ are of the form “$i_d$ cuts in $[0,1]$”, or all of the nodes at depth $d$ are of the form “$i_d$ chooses any available interval between adjacent
for $\forall i \in N$ do
    Agent $i$ makes $\lceil n/\varepsilon \rceil$ cuts in $[0,1]$
end for

$i = 1$

while $\exists$ available pieces between two adjacent cuts do
    Agent $i$ takes his (remaining) favorite piece
    $i = (i \mod n) + 1$
end while

Algorithm 4: $\varepsilon$-envy-free oblivious GCC protocol.

cuts”, for a fixed agent $i_d \in N$. Less formally, the next agent to play and its action space are independent of the history\footnote{We formalize this statement in Lemma 1}. In a cut step, the agent can make a cut anywhere on the cake, and in a choose step, the agent can choose any unclaimed piece which is demarcated by two adjacent cuts. One may be tempted to think that the game tree of an oblivious GCC protocol is simply a path, but in fact it is not, because there are multiple outcomes (i.e., multiple leaves) — different cuts and choices lead to different allocations and values.

Although obliviousness is a significant restriction, we show that oblivious GCC protocols are powerful enough to compute approximately envy-free allocations. Indeed, Algorithm 4 is an oblivious GCC protocol that is $\varepsilon$-envy free, and requires $O(n^2/\varepsilon)$ steps. In words, the agents cut the cake into $O(n^2/\varepsilon)$ pieces (intervals between adjacent cuts), and the pieces are allocated in a round robin fashion. The algorithm’s while loop seems like it violates the obliviousness restriction, but observe that we know in advance how many iterations are required; thus the protocol can be implemented using only Cut and Choose instructions. And why can each agent guarantee envy of at most $\varepsilon$? Taking the point of view of agent $i$, this agent can make the initial cuts so that it values each interval between its own adjacent cuts at most at $\varepsilon$; it follows that it values any of the $O(n^2/\varepsilon)$ pieces induced by everyone’s cuts at most at $\varepsilon$. Partition the choices into phases, where in each phase, $i$ chooses first, followed by agents $i+1, \ldots, n, 1, \ldots, i-1$. In each phase, $i$ prefers its own piece to the piece selected by any other agent. Agent $i$ may envy the choices made by agents $1, \ldots, i-1$ before the beginning of the first phase, but its value for each of these pieces is at most $\varepsilon$.

As we show next, the obliviousness restriction facilitates the existence of exact Nash equilibria:

**Theorem 5.1.** Every oblivious GCC protocol with a bounded number of steps has an (exact) SPNE.

The proof of Theorem 5.1 is relegated to Appendix B; it is completely different from the proof of Theorem 3.1 and in particular relies on real analysis instead of backward induction on a discretized space. In the context of the proof it is worth noting that some papers prove the existence of SPNE in games with infinite action spaces (see, e.g., [11, 13]), but our game does not satisfy the assumptions required therein.

Interestingly, Theorem 5.1 implies that any approximately proportional oblivious GCC protocol (such as Algorithm 4) has exact Nash equilibria, which must be approximately proportional (because an agent has a strategy that guarantees a value of $1/n - \varepsilon$). In contrast, if we do not assume obliviousness then we can achieve exact proportionality — but a proportional GCC protocol is only guaranteed to give rise to approximate equilibria, which are therefore only guaranteed to be approximately proportional (even though the protocol itself is exactly proportional).
References

[1] L. Blumrosen and N. Nisan. On the computational power of demand queries. *SIAM Journal on Computing*, 39(4):1372–1391, 2009.

[2] S. J. Brams and A. D. Taylor. *Fair Division: From Cake-Cutting to Dispute Resolution*. Cambridge University Press, 1996.

[3] S. Brânzei and P. B. Miltersen. Equilibrium analysis in cake cutting. In *Proceedings of the 12th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 327–334, 2013.

[4] Y. Chen, J. K. Lai, D. C. Parkes, and A. D. Procaccia. Truth, justice, and cake cutting. *Games and Economic Behavior*, 77:284–297, 2013.

[5] S. Dobzinski. Better mechanisms for combinatorial auctions via maximal-in-range algorithms? *SIGecom Exchanges*, 7(1):30–33, 2007.

[6] L. E. Dubins and E. H. Spanier. How to cut a cake fairly. *American Mathematical Monthly*, 68(1):1–17, 1961.

[7] J. Edmonds and K. Pruhs. Balanced allocations of cake. In *Proceedings of the 47th Symposium on Foundations of Computer Science (FOCS)*, pages 623–634, 2006.

[8] J. Edmonds and K. Pruhs. Cake cutting really is not a piece of cake. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 271–278, 2006.

[9] S. Even and A. Paz. A note on cake-cutting. *Discrete Applied Mathematics*, 7:285–296, 1984.

[10] L. Fortnow. Program equilibria and discounted computation time. In *Proceedings of the 12th Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, pages 128–133, 2009.

[11] C. Harris. Existence and characterization of perfect equilibrium in games of perfect information. *Econometrica*, 53(3):613–628, 1985.

[12] S. Hart and N. Nisan. The query complexity of correlated equilibria. CoRR abs/1305.4874, 2013.

[13] H. Hellwig and W. Leininger. On the existence of subgame-perfect equilibrium in infinite-action games of perfect information. *Journal of Economic Theory*, 45:55–75, 1987.

[14] D. Kurokawa, J. K. Lai, and A. D. Procaccia. How to cut a cake before the party ends. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*, 2013. Forthcoming.

[15] E. Maskin. Nash equilibrium and welfare optimality. *Review of Economic Studies*, 66:23–38, 1999. First circulated in 1977.

[16] A. Maya and N. Nisan. Incentive compatible two player cake cutting. In *Proceedings of the 8th International Workshop on Internet and Network Economics (WINE)*, pages 170–183, 2012.
A Proof of Theorem 3.1

Let $\varepsilon > 0$, and let $f(n)$ be an upper bound on the number of operations (i.e., on the height of the game tree) of the protocol. Define a grid, $G_1$, such that every cell on the grid is worth at most $\frac{\varepsilon}{2f(n)}$ to each agent. For every $n$, let $K$ denote the maximum number of cut operations, where $0 \leq K \leq f(n)$. For each $i \in \{1, \ldots, K\}$, we define the grid $G_i$ so that the following properties are satisfied:

- The grids are nested, i.e., $\{0,1\} \subset G_1 \subset G_2 \subset \ldots \subset G_K$.
- There exists a unique point $z \in G_{i+1}$ between any two consecutive points $x, y \in G_i$, such that $x < z < y$ and $z \notin G_i$, for every $i \in \{1, \ldots, K - 1\}$.
- Each cell on $G_i$ is worth at most $\frac{\varepsilon}{2f(n)}$ to any agent, for all $i \in \{1, \ldots, K\}$.

Having defined the grids, we compute the backward induction outcome on the discretized cake, where the $i$-th Cut operation can only be made on the grid $G_i$. We will show that this outcome is an $\varepsilon$-SPNE, even though agents could deviate by cutting anywhere on the cake. On the continuous cake, the agents play a perturbed version of the idealized game from the grid $G$, but maintain a bijective mapping between the perturbed game and the idealized version throughout the execution of the protocol. Thus when determining the next action, the agents use the idealized grid as a
Consider the following strategies. Let \( x_1, \ldots, x_k \) be the history of cuts made at some point during the execution of the protocol. For each \( i \in \{1, \ldots, k\} \), map the point \( x_i \) to the closest point \( M(x_i) \in G_i \) such that:

- The order of the points \( x_1, \ldots, x_i \) is the same as the order of \( M(x_1), \ldots, M(x_i) \). Note this is always possible since the grid \( G_i \) is a strict superset of \( G_{i-1} \) and has one point between every two points from \( G_{i-1} \).
- \( M(x_i) = M(x_j) \iff x_i = x_j, \forall j \in \{1, \ldots, i-1\} \).
- \( M(x_i) = 0 \iff x_i = 0 \).
- \( M(x_i) = 1 \iff x_i = 1 \).

Note that the map \( M \) is bijective. Moreover, the value of each agent for every piece with endpoints \( x_i \) and \( M(x_i) \) is at most \( \frac{\varepsilon}{2f(n)} \). That is, \( V_k([x_i, M(x_i)]) \leq \frac{\varepsilon}{2f(n)} \) if \( x_i \leq M(x_i) \), and \( V_k([M(x_i), x_i]) \leq \frac{\varepsilon}{2f(n)} \) otherwise, for all \( k \in N \).

Consider any history of cuts \( (x_1, \ldots, x_k) \). Let \( i \) be the agent that moves next. Agent \( i \) computes the mapping \( (M(x_1), \ldots, M(x_k)) \). If the next operation is:

- **Choose**: agent \( i \) chooses the available piece (identified by the symbolic names of the cut points it contains and their order) which is optimal in the idealized game, given the current state and the existing set of ordered ideal cuts, \( M(x_1), \ldots, M(x_k) \). Ties are broken according to a fixed deterministic scheme which is known to all the agents.
- **Cut**: agent \( i \) computes the optimal cut on \( G_{k+1} \), say at \( x_{k+1}^{*} \). Then \( i \) maps \( x_{k+1}^{*} \) back to a point \( x_{k+1} \) on the continuous game, such that \( M(x_{k+1}) = x_{k+1}^{*} \). That is, the cut \( x_{k+1} \) (made in step \( k+1 \)) is always mapped by the other agents to \( x_{k+1}^{*} \in G_{k+1} \). Agent \( i \) cuts at \( x_{k+1} \).

We claim that these strategies give an \( \varepsilon \)-SPNE. The proof follows from the following claim, which we show by induction on \( t \) (the maximum number of remaining steps of the protocol):

*Whenever there are at most \( t \) operations left in the execution, it is \( \frac{t}{f(n)} \)-optimal to play on the grid.*

Consider any history of play, where the cuts were made at \( x_1, \ldots, x_k \). Without loss of generality, assume it is agent \( i \)'s turn to move.

**Base case**: \( t = 1 \). The protocol has at most one remaining step. If it is a cut operation, then no agent receives any utility in the remainder of the game regardless of where the cut is made. Thus cutting on the grid \( (G_k) \) is optimal. If it is a choose operation, then let \( Z = \{Z_1, \ldots, Z_s\} \) be the set of pieces that \( i \) can choose from. Agent \( i \)'s strategy is to map each piece \( Z_j \) to its equivalent \( M(Z_j) \) on the grid \( G_k \), and choose the piece that is optimal on \( G_k \). Recall that \( V_k([x_j, M(x_j)]) \leq \frac{\varepsilon}{2f(n)} \) for all \( k \in N \). Thus a piece which is optimal on the grid, is \( \frac{\varepsilon}{f(n)} \)-optimal in the continuous game (adding up the difference on both sides). It follows that \( i \) cannot gain more than \( \frac{\varepsilon}{f(n)} \) in the last step by deviating from the optimal piece on \( G_k \).

**Induction hypothesis**: Assume that playing on the grid is \( \frac{(t-1)\varepsilon}{f(n)} \)-optimal whenever there are at most \( t - 1 \) operations left on every possible execution path of the protocol, and there exists one path that has exactly \( t - 1 \) steps.
**Induction step:** If the current operation is *Choose*, then by the induction hypothesis, playing on the grid in the remainder of the protocol is \(\frac{(t-1)\varepsilon}{f(n)}\)-optimal for all the agents, regardless of \(i\)'s move in the current step. Moreover, agent \(i\) cannot gain by more than \(\frac{\varepsilon}{f(n)}\) by choosing a different piece in the current step, compared to piece which is optimal on \(G_k\), since \(V_i([x_l, M(x_l)]) \leq \frac{\varepsilon}{2f(n)}\) for all \(l \in \{1, \ldots, k\}\).

If the current operation is *Cut*, then the following hold:

1. By construction of the grid \(G_{k+1}\), agent \(i\) can induce any given branch of the protocol using a cut in the continuous game if and only if the same branch can be induced using a cut on the grid \(G_{k+1}\).

2. Given that the other agents will play on the grid for the remainder of the protocol, agent \(i\) can change the size of at most one piece that it receives down the road by at most \(\frac{\varepsilon}{f(n)}\) by deviating (compared to the grid outcome), since \(V_j([x_l, M(x_l)]) \leq \frac{\varepsilon}{2f(n)}\) for all \(l \in \{1, \ldots, k+1\}\) and for all \(j \in N\).

Thus by deviating in the current step, agent \(i\) cannot gain more than \(t\frac{\varepsilon}{f(n)}\). Since \(t \leq f(n)\), the overall loss of any agent is bounded by \(\varepsilon\). We conclude that playing on the grid is \(\varepsilon\)-optimal for all the agents. 

**B Proof of Theorem 5.1**

We start with a lemma that crystallizes the properties of games induced by oblivious GCC protocols.

**Lemma 1.** An extensive-form game of perfect information that is induced by an oblivious GCC protocol enjoys the following properties:

1. The action space at every node of the game tree is compact.

2. Valuations are bounded.

3. Every action space is independent of the history.

4. The valuations \(V_1, \ldots, V_n\) are continuous in the actions.

**Proof sketch.** Properties 1 and 2 are trivially true, and also hold for GCC protocols in general.

Properties 3 and 4 are somewhat more delicate. For property 3, we think of a *Choose* step when there are \(k\) available pieces between adjacent cuts as choosing a number \(t \in \{1, \ldots, k\}\), and receiving the available piece that is \(t\)'th from the left (this can be defined precisely); note that \(k\) can be calculated based on the number of cuts and choices before the choose node, i.e., it is independent of the history.

Property 4 is only relevant when considering *Cut* nodes. To establish it, first consider the action in a single *Cut* node, and fix all the other actions. We claim that for every \(\varepsilon > 0\) there exists \(\delta = \delta(\varepsilon) > 0\) that is independent of the choice of actions in other nodes such that moving the cut by at most \(\delta\) changes the values by at most \(\varepsilon\). Indeed, let us examine how pieces change as the cut point moves. As long as the cut point moves without passing any other cut point, one piece shrinks as another grows. As the cut point approaches another cut point, the induced piece — say \(k\)'th from the left — shrinks. When the cut point passes another cut point \(x\), the \(k\)'th piece...
An action tuple is a tuple of \( k \) strategies, and for \( i \in \mathbb{N} \), the \( i \)-th action from the end. Note that the properties of oblivious GCC protocols allow these definitions to be well-defined.

We can now define a strategy in this context. For each \( i \), let \( s_i : X_{i+1} \times X_{i+2} \times \ldots \times X_r \to X_i \) describe the action a player will play given the history of actions played before. We denote by \( Y_i \) the set of all possible strategies for the \( i \)-th action from the end.

**Definition 1.** An action tuple is a tuple of \( r \) actions that describe an outcome of the protocol.

Often we will find it convenient to think of strategies as actions. For example, if \( s_1, s_2, \ldots, s_k \) are strategies, and \( a_{k+1}, a_{k+2}, \ldots, a_r \) are actions, we may refer to \((s_1, s_2, \ldots, s_k, a_{k+1}, a_{k+2}, \ldots, a_r)\) as an action tuple. This can be thought of as equivalent to the action tuple \((a_1, a_2, \ldots, a_r)\) where for \( i \leq k \) we have \( a_i = s_i(a_{i+1}, a_{i+2}, \ldots, a_r)\).

We now take a moment to introduce the valuation functions. Let \( V_i \) represent the valuation for the agent committing the \( i \)-th action from the end. Note that \( V_i \) is not the valuation of some agent denoted by \( i \), in contrast to the rest of the paper. As input, it will take an action tuple and output a value in some bounded range.

With this idea of valuations in hand, we are now ready to define the main notion of interest.

**Definition 2.** A \( k \)-SPNE strategy profile is a \( k \)-tuple of strategies \((s_1, s_2, \ldots, s_k)\) where for all \( i \), \( s_i \in Y_i \) and the \( s_i \) are SPNE. To be more explicit, we define the term recursively: The empty profile is vacuously 0-tail SPNE, and for \( k \geq 1 \), the tuple is \( k \)-tail SPNE if the first \( k-1 \) strategies in the tuple are \((k-1)\)-SPNE and for every \( a_k \in X_k, a_{k+1} \in X_{k+1}, \ldots, a_r \in X_r \) we have:

\[
V_k(s_1, s_2, \ldots, s_k, a_{k+1}, a_{k+2}, \ldots, a_r) \geq V_k(s_1, s_2, \ldots, s_{k-1}, a_k, a_{k+1}, \ldots, a_r).
\]

A \( k \)-SPNE action tuple is an action tuple such that the last \( k \) actions can be induced by a \( k \)-SPNE strategy profile. That is \((a_1, a_2, \ldots, a_r)\) is \( k \)-SPNE if there exists a \( k \)-SPNE strategy profile \((s_1, s_2, \ldots, s_k)\) such that:

\[
(a_1, a_2, \ldots, a_{k-1}, a_k, a_{k+1}, \ldots, a_r) = (s_1, s_2, \ldots, s_{k-1}, s_k, a_{k+1}, \ldots, a_r)
\]
Note that the classical definition of SPNE is equivalent in our setting to the existence of an r-SPNE strategy profile. With the notation and definitions out of the way, we begin our proof of the theorem. We start with the following lemma.

**Lemma 2.** Any convergent sequence of k-SPNE action tuples converges to a k-SPNE action tuple.

*Proof.* We proceed by induction.

1. **Base Case** ($k = 0$): This is vacuously true by the definition of 0-SPNE action tuples.

2. **Inductive Assumption** ($k = m-1$): Assume we have shown that the result is true for $k = m-1$.

3. **Inductive Step** ($k = m$):
   We are given that for every $i$ we have an $m$-SPNE action tuple:
   
   $$A^i = (s^i_1, s^i_2, ..., s^i_m, a^i_{m+1}, a^i_{m+2}, ..., a^i_r)$$
   where the $s_j$ are strategies such that $(s^i_1, s^i_2, ..., s^i_m)$ is $m$-SPNE and the $a^i_j$ are actions. Furthermore, we are given that the sequence $\{A^i\}_i$ converges to some action tuple $A = (a_1, a_2, ..., a_r)$. We wish to show $A$ is an $m$-SPNE action tuple itself.

   Since any $m$-SPNE action tuple is an $(m-1)$-SPNE action tuple, we know that the $A^i$ are $(m-1)$-SPNE action tuples and thus, by the inductive assumption, so too is $A$. Suppose the $(m-1)$-SPNE strategy profile $S = (s_1, s_2, ..., s_{m-1})$ induces $A$. That is, $A = (s_1, s_2, ..., s_{m-1}, a_k, a_{k+1}, ..., a_r)$.

   Now for every $\alpha \in X_m$, let us replace $s_m^i$ with the constant function $\tilde{s}_m^i = \alpha$ in the sequence $\{A^i\}_i$. Let $\{B^i\}_i$ be the resulting sequence of action tuples and call this the *completion* for $\alpha$. Due to the compactness of the action spaces, there exists some convergent subsequence of $\{B^i\}_i$. Denote by $I^\alpha$ the indices of such a convergent subsequence of $\{B^i\}_i$. That is, the definition $B = \lim_{i \in I^\alpha} B^i$ is well defined. As this subsequence is a convergent sequence of $(m-1)$-SPNE action tuples, we have by the inductive assumption that there exists some $(m-1)$-SPNE strategy profile $(t^1_1, t^2_2, ..., t^m_{m-1})$ that induces $B$.

   We are now ready to construct the desired strategy profiles $t_i : X_{i+1} \times X_{i+2} \times ... \times X_r \rightarrow X_i$. For $i < m$, define $t_i$ as:

   $$t_i(x_{i+1}, x_{i+2}, ..., x_r) = \begin{cases} 
   t^m_i(x_{i+1}, x_{i+2}, ..., x_r) & \text{if } x_m \neq a_m \text{ and for all } j > m : x_j = a_j \\
   s_1(x_{i+1}, x_{i+2}, ..., x_r) & \text{if for all } j \geq m : x_j = a_j \\
   s^i_1(x_{i+1}, x_{i+2}, ..., x_r) & \text{otherwise.}
   \end{cases}$$

   Define $t_m$ as:

   $$t_m(x_{m+1}, x_{m+2}, ..., x_r) = \begin{cases} 
   a_m & \text{if for all } j > m : x_j = a_j \\
   s^1_m(x_{m+1}, x_{m+2}, ..., x_r) & \text{otherwise.}
   \end{cases}$$

   We claim that the strategy profile $T = (t_1, t_2, ..., t_m)$ is $m$-SPNE. If the actions $x_j \in X_j$ for $j \geq m$ are fixed, then by the definition of $t_i$ for $i < m$, we have that $(t_1, t_2, ..., t_{m-1})$ is one of $(t^1_1, t^2_2, ..., t^m_{m-1}), (s_1, s_2, ..., s_{m-1}),$ or $(s^1_1, s^1_2, ..., s^1_{m-1})$. Regardless, it is equivalent to some
Now suppose $x \in X_m$, its completion $\{(b_1^i, b_2^i, ..., b_r^i)\}_i$ are the completion for $x_m$. If for all $j > m$ we have that $x_j = a_j$, then we see that:

$$V_m(t_1, t_2, ..., t_m-1, t_m, x_m+1, x_m+2, ..., x_r)$$

$$= V_m(t_1, t_2, ..., t_m-1, t_m, a_m+1, a_m+2, ..., a_r)$$

$$= V_m(t_1, t_2, ..., t_m-1, a_m, a_m+1, a_m+2, ..., a_r)$$

$$= V_m(s_1, s_2, ..., s_{m-1}, a_m+1, a_m+2, ..., a_r)$$

$$= V_m(b_1^i, b_2^i, ..., b_{m-1}^i, b_m, b_{m+1}^i, b_{m+2}^i, ..., b_r^i)$$

In particular, this (along with continuity of $V_m$) implies that:

$$\lim_{i \to \infty} V_m(s_1^i, s_2^i, ..., s_{m-1}^i, a_m^i, a_{m+1}^i, a_{m+2}^i, ..., a_r^i)$$

$$\geq \lim_{i \in I, i \to \infty} V_m(b_1^i, b_2^i, ..., b_{m-1}^i, b_m, b_{m+1}^i, b_{m+2}^i, ..., b_r^i)$$ (by the observation)

If on the other hand we have that there exists some $j > m$ for which $x_j \neq a_j$, then we see that $(t_1, t_2, ... t_m)$ is equivalent to $(s_1^i, s_2^i, ..., s_m^i)$ — which is $m$-SPNE. Therefore, $T$ is $m$-SPNE.

All that remains to prove is that $A$ is indeed induced by the strategy profile $T$. The following shows exactly this.

$$(t_1, t_2, ..., t_m-1, t_m, a_m+1, a_m+2, ..., a_r) = (s_1, s_2, ..., s_{m-1}, b_m, b_{m+1}, b_{m+2}, ..., b_r)$$

This completes the proof.

□
We are now ready for our main result.

**Lemma 3.** Every oblivious GCC protocol has a $k$-SPNE strategy profile for every $k \leq r$.

**Proof.** Again, we proceed by induction.

1. **Base Case ($k = 0$):** This is vacuously true by the definition of 0-SPNE strategy profile.

2. **Inductive Assumption ($k = m - 1$):** Assume we have shown that the result is true for $k = m - 1$.

3. **Inductive Step ($k = m$):**

   We wish to construct an $m$-SPNE strategy profile $(t_1, t_2, ..., t_m)$. By the inductive assumption, we know there exists some $(m - 1)$-SPNE strategy profile $(s_1, s_2, ..., s_{m-1})$. In addition, let $x_{m+1} \in X_{m+1}, x_{m+2} \in X_{m+2}, ..., x_r \in X_r$ be fixed. As $V_m$ is bounded, we know that

   $$SUP = \sup\{V_m(s_1, s_2, ..., s_{m-1}, x_m, x_{m+1}, ..., x_r) \mid x_m \in X_m\}$$

   is well-defined. Via the continuity of $V_m$, we can then construct a sequence $\{x_m^i\}_i$ such that:

   $$\lim_{i \to \infty} V_m(s_1, s_2, ..., s_{m-1}, x_m^i, x_{m+1}, ..., x_r) = SUP.$$

   Now note that the sequence of action tuples $\{(s_1, s_2, ..., s_{m-1}, x_m^i, x_{m+1}, x_{m+2}, ..., x_r)\}_i$ is a sequence in a compact space and thus, has a convergent subsequence. Via use of a possible redefinition, assume the sequence itself converges. Then by Lemma 2, we know that this convergent sequence of $(m - 1)$-SPNE action tuples converges to an $(m - 1)$-SPNE action tuple induced by some $(m - 1)$-SPNE strategy profile $(\tilde{s}_1, \tilde{s}_2, ..., \tilde{s}_{m-1})$.

   Suppose that the desired strategy profiles $t_i$ satisfy the following. For $i < m$, $t_i$ satisfies:

   $$t_i(x_{i+1}, x_{i+2}, ..., x_r) = \begin{cases} 
   \tilde{s}_i(x_{i+1}, x_{i+2}, ..., x_r) & \text{if } x_m = \lim_{i \to \infty} x_m^i \\
   s_i(x_{i+1}, x_{i+2}, ..., x_r) & \text{otherwise},
   \end{cases}$$

   and $t_m$ satisfies:

   $$t_m(x_{m+1}, x_{m+2}, ..., x_r) = \lim_{i \to \infty} x_m^i.$$

   Clearly, $(t_1, t_2, ..., t_{m-1})$ is $(m - 1)$-SPNE since for the fixed $x_j \in X_j$, for $j \geq m$, it is equivalent to some other $(m - 1)$-SPNE strategy profile. It therefore suffices to show that $t_m$ is weakly optimal. This is true as we have for every $a_m \in X_m$ where $a_m \neq \lim_{i \to \infty} x_m^i$:

   $$V_m(t_1, t_2, ..., t_{m-1}, t_m, a_m, x_{m+1}, x_{m+2}, ..., x_r) = V_m(s_1, s_2, ..., s_{m-1}, t_m, x_{m+1}, x_{m+2}, ..., x_r)
   \geq V_m(s_1, s_2, ..., s_{m-1}, a_m, x_{m+1}, x_{m+2}, ..., x_r)
   = V_m(t_1, t_2, ..., t_{m-1}, a_m, x_{m+1}, x_{m+2}, ..., x_r)$$

   From the above lemma, we see immediately that every bounded oblivious GCC protocol has an $r$-SPNE (where $r$ is the number of actions), and thus has an SPNE. This completes the theorem’s proof.