From Independence of Clones to Composition Consistency: A Hierarchy of Barriers to Strategic Nomination

Ratip Emin Berker, Sílvia Casacuberta, Isaac Robinson, Christopher Ong, Vincent Conitzer, and Edith Elkind

Abstract

We study two axioms for social choice functions that capture the impact of similar candidates: independence of clones (IoC) and composition consistency (CC). We clarify the relationship between these axioms by observing that CC is strictly more demanding than IoC, and investigate whether common voting rules that are known to be independent of clones (such as STV, Ranked Pairs, Schulze, and Split Cycle) are composition-consistent. While for most of these rules the answer is negative, we identify a variant of Ranked Pairs that satisfies CC. Further, we show how to efficiently modify any (neutral) social choice function so that it satisfies CC, while maintaining its other desirable properties. Our transformation relies on the hierarchical representation of clone structures via PQ-trees. We extend our analysis to social preference functions. Finally, we interpret IoC and CC as measures of robustness against strategic manipulation by candidates, with IoC corresponding to strategy-proofness and CC corresponding to obvious strategy-proofness.

1 Introduction

On November 6th, 1934, Oregonians took to the polls to elect their 28th governor. Earlier, in a contested Republican primary, Senator J. Dunne had narrowly defeated P. Zimmerman, who then decided to run as an independent. In the subsequent general election, each candidate received the following votes [48]:

Charles Martin	Peter Zimmerman	Joe Dunne
116,677	95,519	86,923

The Democratic candidate (Martin) won, even though the two Republicans collectively won nearly 60% of the vote. This example motivates the following question: *How can we ensure that similar candidates in an election do not 'spoil' the election, preventing each other from winning?*

Naturally, some winner determination rules, or *social choice functions (SCFs)*, are more resilient to this "spoilage" effect than others. The field of social choice offers a rich variety of SCFs and formulates various desirable criteria (*axioms*) for them. In particular, Tideman [62] defines a *clone set*, *i.e.*, a group of candidates (clones) that are ranked consecutively in all voter's rankings, and puts forward the axiom of *independence of clones (IoC)*, which asks that if a candidate is an election winner, this should remain the case even if we add[1] or remove clones of her opponents. In the context of political elections, IoC means that a political party need not be strategic about the number of party representatives participating in an election, as long as it does not care which of its candidates wins. Conversely, if a rule fails IoC, adding/deleting clones may be a viable strategy to change the outcome (Tideman himself recalls winning a grade school election after nominating his opponent's best friend). Moreover, Elkind et al. [22] show that the algorithmic problem of purposeful cloning (*i.e.*, to change the outcome of an election) is easy for many common SCFs. Thus, the appeal of SCFs satisfying IoC goes beyond the theoretical.

In settings with abstract candidates, it is even easier to introduce clones. For example, when candidates consist of drafts of a text—*e.g.*, we are voting over drafts of the guiding principles of our organization—it

[1] For example, under veto/anti-plurality, a non-IoC SCF that picks the candidate(s) ranked bottom by the least number of voters, introducing clones can *help* the cloned candidate, as the clones split the last places in votes.

is straightforward to introduce a near-duplicate of an existing draft. When candidates are AI systems—*e.g.*, we are ranking LLMs, as done for example on Chatbot Arena [13], to determine the best one—one can introduce a second version of a model, one that is fine-tuned only slightly differently (as has already been pointed out by Conitzer et al. [15]). Without IoC, such clones can critically affect the outcome.

On the other hand, IoC may not be *enough* to dissuade strategic cloning. While IoC dictates that the cloning of a candidate should not change whether *one* of the clones wins, it does not specify *which* clone should win, even though one of them can be significantly preferred to the others by the voters. As such, even with an IoC rule, cloning can have a significant impact on the result by changing which candidate among the clones wins. Moreover, a rule being IoC does not reveal how *obviously* robust it is against strategic nomination. As demonstrated by Li [44] in the context of obvious strategy-proofness, the benefits of a property of a mechanism might only be materialized if the agents actually *believe* that the property indeed holds. Even when using an IoC rule, it is not clear that the average voter or candidate can be easily convinced of this property—resulting, for example, in a candidate unnecessarily dropping out of the race, either out of fear of hurting their party, or of being blamed by their voters for doing so.

These drawbacks of IoC might be one possible explanation for why major parties in the United States still hold internal primaries to pick a single nominee for elections that pick a winner using single transferable vote [53], which is IoC. This happens even though consolidating party support behind a single candidate does not improve the chances of the party winning the election, and they could provide voters with a wider range of choices by letting *all* their willing candidates run in the general election.

To this end, we turn to the *stronger* axiom of *composition consistency (CC)*, introduced by Laffond et al. [36], which dictates not just which clone sets win, but which clones among those sets win too. CC, as we will argue, also exposes the *obviousness* of a rule's robustness against strategic nomination. When introducing CC, Laffond et al. were seemingly unaware of the IoC definition by Tideman [62], despite using equivalent (but differently-named) concepts. This, among other factors, has led the literature on IoC and CC to progress relatively independently, with few papers identifying them as comparable axioms. By studying these axioms in a unified framework, we hope to help dispel this ambiguity.

1.1 Our Contributions

(1) We clarify the relationship between IoC and CC—which has historically been ambiguous (see Appendix A.1 for an overview)—by formally showing that CC is strictly more demanding (Proposition 8).

(2) We provide (to the best of our knowledge) the first ever analysis of whether SCFs that are known to be IoC (*e.g.*, Ranked Pairs, Beatpath/Schulze Method, and Split Cycle) also satisfy the stronger property of CC, thereby also establishing where each rule falls in our hierarchy of barriers to strategic nomination (Section 3). While for most of these rules the answer is negative, we identify a variant of Ranked Pairs that satisfies CC. We connect our results to the literature on *tournament solutions (TSs)* in Appendix B.

(3) We introduce an efficient algorithm that modifies *any* (neutral) SCF into a new rule satisfying CC, while preserving various desirable properties, *e.g.*, Condorcet/Smith consistency, among others (Section 4). Our transformation relies on the fact that clone sets (which can be nested and overlapping) can be represented by PQ-trees [23], allowing us to recursively zoom into the "best" clone set.

(4) We formalize the connection of IoC/CC to strategic behavior by candidates via the model of *strategic candidacy* [19] (Section 5). We show that if the candidates' preferences over each other are dictated by their clone structure, IoC rules ensure running in the election is a dominant strategy, hence achieving a stronger version of *candidate stability*. However, IoC is not enough for *obvious* strategy-proofness, which we show can be achieved by CC rules using our PQ-tree algorithm.

(5) In Appendix C, we provide the first extension of CC to social preference functions (SPF) and prove that many of our characterization results generalize. Nevertheless, we give a negative result showing that no anonymous SPF can be CC, and discuss ways in which this can be circumvented.

2 Preliminaries

Profiles and clones. We consider a set of *candidates* A with $|A| = m$ and a set of *voters* $N = \{1, \dots, n\}$. A *ranking* over A is an asymmetric, transitive, and complete binary relation $\succ$ on A. Let $\mathcal{L}(A)$ denote the set of all rankings over A; $a \succ_r b$ indicates that a is ranked above b in a ranking r. Each voter $i \in N$ has a ranking $\sigma_i \in \mathcal{L}(A)$; we collect the rankings of all voters in a *preference profile* $\boldsymbol{\sigma} \in \mathcal{L}(A)^n$. The next definition helps identify sets of similar candidates (according to voters).

6 voters	5 voters	2 voters	2 voters
b	d	a	a
c	c	d	d
a	b	b	c
d	a	c	b

Figure 1: A preference profile. Columns show rankings, with the bottom row ranked last. The first row shows the number of copies of each ranking (*e.g.*, leftmost column indicates 6 voters rank $b \succ c \succ a \succ d$).

Definition 1 (Tideman [62, §I]; Laffond et al. [36, Def. 4])**.** Given a preference profile $\boldsymbol{\sigma}$ over candidates A, a nonempty subset of candidates $K \subseteq A$ is a *set of clones* with respect to $\boldsymbol{\sigma}$ if for each $a, b \in K$ and each $c \in A \setminus K$, no voter ranks c between a and b.

All preference profiles admit two types of *trivial* clone sets:[2] (1) the entire candidate set A, and (2) for each $a \in A$, the singleton $\{a\}$. We call all other clone sets *non-trivial*. For example, for the profile in Figure 1, the only non-trivial clone set is $\{b, c\}$.

Social choice functions and axioms. A *social choice function (SCF)* is a mapping f that, given a profile $\boldsymbol{\sigma}$ over candidates A, outputs a nonempty subset of A; the candidates in $f(\boldsymbol{\sigma})$ are the *winners* under f. An SCF f is *decisive* on $\boldsymbol{\sigma}$ if $|f(\boldsymbol{\sigma})| = 1$. Table 1 gives the descriptions of the SCFs we consider.[3]

We list some desirable properties (*axioms*) for SCFs. For example, an SCF is *neutral* (resp. *anonymous*) if its output is robust to relabeling the candidates (resp. voters); for a formal definition, see Zwicker [66, Def. 2.4, 2.5]. The *Smith set* of $\boldsymbol{\sigma}$ (denoted $Sm(\boldsymbol{\sigma})$) is the smallest set of candidates who all pairwise defeat (preferred to by a strict majority of voters) every candidate outside the set. An SCF f satisfies Smith (resp. Condorcet) consistency if $f(\boldsymbol{\sigma}) \subseteq Sm(\boldsymbol{\sigma})$ for all $\boldsymbol{\sigma}$ (resp. for all $\boldsymbol{\sigma}$ with $|Sm(\boldsymbol{\sigma})| = 1$).

Next, we will present two axioms that both aim to capture the idea of robustness against strategic nomination. In what follows, we write $\boldsymbol{\sigma} \setminus A'$ to denote the profile obtained by removing the elements of $A' \subset A$ from each voter's ranking in $\boldsymbol{\sigma}$ while preserving the order of all other candidates.

Definition 2 (Zavist and Tideman [65])**.** An SCF f is *independent of clones (IoC)* if for each profile $\boldsymbol{\sigma}$ over A and each non-trivial clone set $K \subset A$ with respect to $\boldsymbol{\sigma}$,

(1) for all $a \in K$, we have $K \cap f(\boldsymbol{\sigma}) \neq \emptyset \Leftrightarrow (K \setminus \{a\}) \cap f(\boldsymbol{\sigma} \setminus \{a\}) \neq \emptyset$;

(2) for all $a \in K$ and all $b \in A \setminus K$, we have $b \in f(\boldsymbol{\sigma}) \Leftrightarrow b \in f(\boldsymbol{\sigma} \setminus \{a\})$.

Intuitively, IoC dictates that deleting one of the clones in a non-trivial clone set K must not alter the winning status of K as a whole, or of any candidate not in K.

Example 3. *In the profile $\boldsymbol{\sigma}$ in Fig. 2 (left), $K = \{a_1, a_2\}$ is a clone set. Plurality Voting (PV) outputs b as the unique winner. However, $PV(\boldsymbol{\sigma} \setminus \{a_2\}) = \{a_1\}$, since with a_2 gone, a_1 now has 5 voters ranking it first (Figure 2, right). This violates both conditions (1) and (2) from Definition 2.*

In contrast, STV eliminates a_2 (whose votes then transfer to a_1), then c and finally b, so that a_1 is elected; moreover, it produces the same result on $\boldsymbol{\sigma} \setminus \{a_2\}$. This is in line with the fact that STV is IoC [62].

[2] Tideman [62] in fact excludes trivial clone sets. We use the definition from the work of Elkind et al. [23].

[3] While we describe some SCFs by their winner determination *procedures*, the SCFs themselves are the *functions* that output the respective winners, and these functions may be computed by other—possibly more efficient—algorithms.

3 voters	2 voters	4 voters	3 voters
a_1	a_2	b	c
a_2	a_1	c	a_2
b	b	a_2	a_1
c	c	a_1	b

$\xrightarrow{\text{remove } a_2}$

5 voters	4 voters	3 voters
a_1	b	c
b	c	a_1
c	a_1	b

Figure 2: (Left) Example profile $\boldsymbol{\sigma}$. (Right) $\boldsymbol{\sigma} \setminus \{a_2\}$.

5 voters	4 voters	3 voters
K_a	K_b	K_c
K_b	K_c	K_a
K_c	K_a	K_b

3 voters	9 voters
a_1	a_2
a_2	a_1

Figure 3: (Left) $\boldsymbol{\sigma}^{\mathcal{K}}$, where clone sets from $\boldsymbol{\sigma}$ in Figure 2 are condensed into candidates K_a, K_b, and K_c. (Right) $\boldsymbol{\sigma}|_{K_a}$, where $\boldsymbol{\sigma}$ is limited to members of K_a.

To define the second axiom regarding strategic nomination, we first introduce a few additional concepts.

Definition 4. Given a preference profile $\boldsymbol{\sigma}$ over candidates A, a set of sets $\mathcal{K} = \{K_1, K_2, \ldots, K_\ell\}$, where $K_i \subseteq A$ for all $i \in [\ell]$, is a *(clone) decomposition* with respect to $\boldsymbol{\sigma}$ if (1) $\mathcal{K}$ is a partition of A into pairwise disjoint subsets, and (2) each K_i is a non-empty clone set with respect to $\boldsymbol{\sigma}$.

Every profile has at least two decompositions: the *null* decomposition $\mathcal{K}_{null} = \{A\}$ and the *trivial* decomposition $\mathcal{K}_{triv} = \{\{a\}\}_{a \in A}$. Given a decomposition $\mathcal{K}$ with respect to $\boldsymbol{\sigma}$, for each $i \in N$ let $\sigma_i^{\mathcal{K}}$ be voter i's ranking over the sets in $\mathcal{K}$; this is well-defined, since each clone set forms an interval in σ_i. The profile $\boldsymbol{\sigma}^{\mathcal{K}} = \{\sigma_i^{\mathcal{K}}\}_{i \in N}$ over $\mathcal{K}$ is called the *summary* of $\boldsymbol{\sigma}$ with respect to the decomposition $\mathcal{K}$. For each $K \in \mathcal{K}$, we write $\boldsymbol{\sigma}|_K$ to denote the restriction of $\boldsymbol{\sigma}$ to K, so that $\boldsymbol{\sigma}|_K \equiv \boldsymbol{\sigma} \setminus (A \setminus K)$.

Definition 5. The *composition product* function of an SCF f is a function Π_f that takes as input a profile $\boldsymbol{\sigma}$ and a clone decomposition $\mathcal{K}$ with respect to $\boldsymbol{\sigma}$ and outputs $\Pi_f(\boldsymbol{\sigma}, \mathcal{K}) \equiv \bigcup_{K \in f(\boldsymbol{\sigma}^{\mathcal{K}})} f(\boldsymbol{\sigma}|_K)$.

Intuitively, Π_f first runs the input SCF f on the summary (specified by $\mathcal{K}$), collapsing each clone set into a meta-candidate K_i. It then "unpacks" each winning clone set, and runs f once again on each.

Example 6. *For the profile $\boldsymbol{\sigma}$ from Figure 2 (left), it holds that $\mathcal{K} = \{K_a, K_b, K_c\}$ with $K_a = \{a_1, a_2\}$, $K_b = \{b\}$, $K_c = \{c\}$ is a valid clone decomposition with respect to $\boldsymbol{\sigma}$. Figure 3 shows $\boldsymbol{\sigma}^{\mathcal{K}}$ and $\boldsymbol{\sigma}|_{K_a}$. We have $STV(\boldsymbol{\sigma}^{\mathcal{K}}) = \{K_a\}$ and $STV(\boldsymbol{\sigma}|_{K_a}) = \{a_2\}$, implying $\Pi_{STV}(\boldsymbol{\sigma}, \mathcal{K}) = \{a_2\}$.*

Together, Examples 3 and 6 imply that $STV(\boldsymbol{\sigma}) \neq \Pi_{STV}(\boldsymbol{\sigma}, \mathcal{K})$ for this $\boldsymbol{\sigma}$ and $\mathcal{K}$; *i.e.*, that STV does not respect this clone decomposition—even though the winners are from the same clone set. We now state the composition consistency axiom, which precisely requires a rule to respect *all* decompositions.

Definition 7 (Laffond et al. [36, Def. 11])**.** A neutral[4] SCF f is *composition-consistent (CC)* if for all preference profiles $\boldsymbol{\sigma}$ and all clone decompositions $\mathcal{K}$ with respect to $\boldsymbol{\sigma}$, we have $f(\boldsymbol{\sigma}) = \Pi_f(\boldsymbol{\sigma}, \mathcal{K})$.

CC rules choose the "best" candidates from the "best" clone sets. In contrast, IoC is much more permissive for choosing a candidate from a best clone set. Indeed, in Proposition 8 we show that CC implies IoC. On the other hand, Examples 3 and 6 demonstrate that the converse is false: they show STV, which is IoC, is not CC. Later, we analyze other IoC rules to show whether they are CC (Section 3).

[4]Laffond et al. [36] define composition consistency (CC) for neutral SCFs; this is without loss of generality, as they treat $\boldsymbol{\sigma}^{\mathcal{K}}$ as a profile over candidates $\{1, 2, \ldots, |\mathcal{K}|\}$, in which case CC automatically implies neutrality by the trivial decomposition. Brandl et al. [3] instead use a definition where $\boldsymbol{\sigma}^{\mathcal{K}}$ is simply $\boldsymbol{\sigma}$ with all but one candidate removed from each K_i; nevertheless, they show that in this model too CC implies neutrality (their Lemma 1). We explicitly state this as a prerequisite for simplicity.

Social choice functions considered in this paper. Prior work has shown each SCF in Table 1 (with the exception of PV) to be IoC (*cf.* Holliday and Pacuit [31] for an overview). For some, winner determination may require tie-breaking (*e.g.*, under STV, candidates may tie for the lowest plurality score). We define the output of such SCFs as the set of candidates that win for *some* tie-breaking rule, also called *parallel-universes tiebreaking* [14]. Crucially, this variant of RP is *not* IoC [65], a nuance we will address in detail in Section 3. Lastly, there exists *tournament solutions* that are known to fail or satisfy CC. However, whether they satisfy CC as SCFs is a more subtle issue, addressed in Appendix B.

Name of SCF	f	*Description of the SCF's output on input profile* $\boldsymbol{\sigma}$
Plurality	PV	Outputs the candidate(s) ranked first by the most number of voters.
Single Transferable Vote	STV	At each round, the candidate ranked top by the fewest voters is eliminated. Eventually a single candidate remains, becoming the winner.
Ranked Pairs [62, 65]	RP	Given a profile $\boldsymbol{\sigma}$ over candidates $A = \{a_i\}_{i\in[m]}$, construct the *margin matrix* M, whose ij entry is the number of voters who rank a_i ahead of a_j minus those who rank a_j ahead of a_i. Construct a digraph over A by adding edges for each $M[ij] \geq 0$ in non-increasing order, skipping those that result in a cycle. The winner is the source node.
Beatpath (Schulze Method) [56]	BP	Construct M as in RP, and the corresponding weighted digraph over A without skipping edges.† Let $S[i,j]$ be the width (min. weight edge) of the widest path from a_i to a_j, computed, *e.g.*, with the Floyd–Warshall algorithm. Then a_i is a winner iff $S[i,j] \geq S[j,i]$ for all $j \in [m]$.
Alternative-Smith [61]	AS	(1) Eliminate all candidates not in $Sm(\boldsymbol{\sigma})$. (2) In the remaining profile, eliminate the candidate ranked top by the fewest voters. Repeat (1)-(2) until a single candidate remains; this is the winner.
Split Cycle [31]	SC	Construct M as in RP and the corresponding weighted digraph G over A without skipping edges. For each *simple cycle* (cycles visiting each vertex at most once) in G, label the edge(s) with the smallest weight in that cycle. Discard all labeled edges (at once) to get G'. The winners are the candidates with no incoming edge in G'.

Table 1: SCFs considered in this paper. Second column indicates our notation for the SCF as a function.

† More generally, BP can be defined with various choices for edge weights [56]. We use the "margin" variant; our results easily generalize to others.

3 Analysis of IoC Social Choice Functions

In this section, we analyze whether the IoC rules in Table 1 satisfy CC. The answer turns out to be positive for RP with a specific tie-breaking rule by Zavist and Tideman [65], but negative for all other SCFs. We first formalize the CC to IoC relationship (*cf.* Brandl et al. [3]), which has historically been ambiguous (see Appendix A.1 for an extended discussion of the history of the axioms).

Proposition 8. *If a given SCF is composition-consistent, then it is also independent of clones.*

All omitted proofs are in the appendix. We show that the converse of Proposition 8 is not true.

Theorem 1. *STV, BP, AS, and SC all fail composition consistency.*

Proof. The statement for STV follows from Examples 3 and 6. For $\boldsymbol{\sigma}$ and $\mathcal{K}$ from these examples, $AS(\boldsymbol{\sigma}) = STV(\boldsymbol{\sigma})$ and $\Pi_{AS}(\boldsymbol{\sigma}, \mathcal{K}) = \Pi_{STV}(\boldsymbol{\sigma}, \mathcal{K})$; hence they also show AS is not CC. For BP and SC, we use the profile from Fig. 1 (say, $\boldsymbol{\sigma}'$), with $\mathcal{K}' = \{\{a\}, \{b,c\}, \{d\}\}$. We have $BP(\boldsymbol{\sigma}') = SC(\boldsymbol{\sigma}') = \{b,c\}$, whereas $\Pi_{BP}(\boldsymbol{\sigma}', \mathcal{K}) = \Pi_{SC}(\boldsymbol{\sigma}', \mathcal{K}') = \{b\}$. See Appendix D.2 for detailed calculations. □

Ranked Pairs Our definition of SCFs deals with ties by returning all candidates that win via *some* tie-breaking method. In particular, for RP the margin matrix M may contain ties, so we need a *tie-breaking order over unordered pairs* to decide the order of adding edges to the digraph. Tideman [62] originally defined Ranked Pairs as returning all candidates that win for some tie-breaking order (we refer to this rule as RP); later, Zavist and Tideman [65] showed that this rule is *not* IoC. By Proposition 8, this also implies RP fails CC. Zavist and Tideman [65] propose breaking ties based on the vote of a fixed voter $i \in N$, which makes RP satisfy IoC. Specifically, they use σ_i to construct a *tie-breaking ranking* Σ_i over unordered pairs in A as follows: (1) order the elements within each pair according to σ_i; (2) rank the pairs according to σ_i's ranking of their first elements; (3) rank pairs with the same first element according to σ_i's ranking of the second elements. For example, for $A = \{a, b, c\}$ and $\sigma_i : a \succ b \succ c$ we get $\Sigma_i : \{a, a\} \succ \{a, b\} \succ \{a, c\} \succ \{b, b\} \succ \{b, c\} \succ \{c, c\}$. Using Σ_i, we can construct a complete *priority order* $\mathcal{L}$ over ordered pairs: pairs are ordered (in non-increasing order) according to M, with ties broken by Σ_i (if $M[a, b] = 0$, we rank $(a, b) \succ_{\mathcal{L}} (b, a)$ if and only if $a \succ_i b$). Then, *Ranked Pairs using voter i as a tie-breaker* (which we call RP_i) adds edges from M to a digraph according to $\mathcal{L}$, skipping those that create a cycle. Zavist and Tideman [65] show that RP_i is IoC. We now strengthen this result.

Theorem 2. *RP_i is composition-consistent for any fixed $i \in N$.*

Proof sketch. The proof uses an equivalence between the topological orders of the final RP graphs and *stacks* over A, which are rankings r where $a \succ_r b$ implies there is a path in M from a to b consistent with the ranking r and with each link at least as strong as $M[b, a]$ [65]. We extend this equivalence to specific stacks with respect to a priority order $\mathcal{L}$, and show that this definition is satisfied by the RP_i ranking, its summary using any $\mathcal{K}$, and its restriction to any clone set. This allows us to establish an agreement between RP_i and Π_{RP_i}, proving RP_i is CC. The full proof can be found in Appendix D.3. □

Moreover, RP_i is poly-time computable, whereas the outputs of RP are NP-hard to compute [9]. However, for any fixed i this rule breaks anonymity, *i.e.*, it fails to treat all voters equally. Holliday and Pacuit [31] suggest (based on personal communication with Tideman) returning $RP_N(\boldsymbol{\sigma}) \equiv \bigcup_{i \in N} RP_i(\boldsymbol{\sigma})$, *i.e.*, declaring an $a \in A$ to be a winner if and only if $a \in RP_i(\boldsymbol{\sigma})$ for *some* $i \in N$. This modification recovers anonymity while preserving IoC and tractability, but we show that it loses CC.

Proposition 9. *RP_N is independent of clones, but not composition-consistent.*

Thus, Ranked Pairs without tie-breaking (RP) is neither IoC, CC, nor tractable. Using a voter to break ties (RP_i), we get all three, but lose anonymity. Recovering anonymity via a union over all voters (RP_N) keeps IoC and tractability, but loses CC.[5] Figure 6 (Appendix B) summarizes this section's results.

4 CC Transformation

All IoC SCFs considered in Section 3, except for RP_i, fail CC. Having more CC rules would be desirable, considering their strong guarantees against strategic behavior (Section 5). To this end, we prove any neutral SCF can be efficiently modified to satisfy CC, while preserving its various desirable properties.

4.1 Background: Clone Structures and PQ-Trees

For a profile $\boldsymbol{\sigma}$, Elkind et al. [23] define the *clone structure* $\mathcal{C}(\boldsymbol{\sigma}) \subseteq \mathcal{P}(A)$ as the family of *all* clone sets with respect to $\boldsymbol{\sigma}$. For example, for $\boldsymbol{\sigma}$ from Fig. 1, $\mathcal{C}(\boldsymbol{\sigma}) = \{\{a\}, \{b\}, \{c\}, \{d\}, \{b, c\}, \{a, b, c, d\}\}$. They identify two types of *irreducible* clone structures: a *maximal* clone structure (also called a *string of*

[5] For probabilistic SCFs (PSCFs), a tempting approach is to pick an $i \in N$ uniformly at random and return $RP_i(\boldsymbol{\sigma})$. The counterexample from Prop. 9 also shows that this variant fails the CC definition for PSCFs given by Brandl et al. [3].

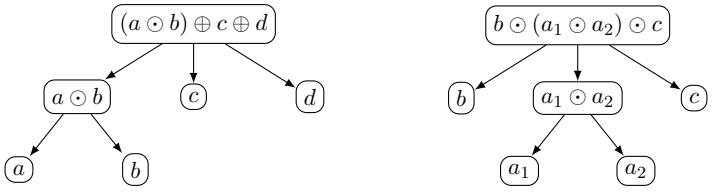

Figure 4: (Left) The PQ-tree representing $\mathcal{C}(\boldsymbol{\sigma})$ from Example 10 . (Right) The PQ-tree of $\boldsymbol{\sigma}$ from Figure 2.

sausages) and a *minimal* clone structure (also called a *fat sausage*). A string of sausages arises when each ranking in $\boldsymbol{\sigma}$ is either a fixed linear order (say, $\sigma_1 : a_1 \succ a_2 \succ \cdots \succ a_m$) or its reversal. In this case, $\mathcal{C}(\boldsymbol{\sigma}) = \{\{a_k\}_{i \le k \le j} : i \le j\}$, *i.e.*, all intervals in σ_1. The *majority ranking* of the string of sausages is σ_1 or its reverse, depending on which one appears more frequently in $\boldsymbol{\sigma}$ (breaking ties arbitrarily). A fat sausage occurs when $\mathcal{C}(\boldsymbol{\sigma}) = \{A\} \cup \{\{a_i\}\}_{i \in [m]}$, *i.e.*, the structure only has the trivial clone sets.

Our CC transformation uses *PQ-trees*: a data structure first defined by Booth and Lueker [2] and later used by Elkind et al. [23] to represent clone sets. Here, we present the definitions required for our construction; for the full treatment, see Elkind et al. [23] (and our Appendix E.1). A *PQ-tree* T over A is an ordered tree whose leaves correspond to the elements of A. To represent a clone structure $\mathcal{C}(\boldsymbol{\sigma})$ as a PQ-tree, we iteratively identify irreducible subfamilies of $\mathcal{C}(\boldsymbol{\sigma})$, and collapse them into a single meta-candidate. If the subfamily corresponds to a fat sausage, we group its members under an internal node of type P, denoted as a $\odot$-product of its children. On the other hand, if the subfamily corresponds to a string of sausages, we group its members under an internal node of type Q, denoted as a $\oplus$-product of its children. In rankings compatible with $\mathcal{C}(\boldsymbol{\sigma})$, the children of a P-node can be permuted arbitrarily; the order of the children of a Q-node must follow its majority ranking or its reversal. Crucially, the order of collapsing is not important, as the irreducible subfamilies of a clone structure are non-overlapping.

Example 10. *Let $\boldsymbol{\sigma}$ be a profile on $A = \{a, b, c, d\}$ with two rankings: $a \succ b \succ c \succ d$ and $d \succ c \succ a \succ b$. Then, $\mathcal{C}(\boldsymbol{\sigma}) = \{\{a\}, \{b\}, \{c\}, \{d\}, \{a, b\}, \{c, d\}, \{a, b, c\}, A\}$. Collapsing the irreducible subfamily $K_1 = \{a, b\}$, the updated $\mathcal{C}(\boldsymbol{\sigma})$ is $\{\{c\}, \{d\}, \{K_1\}, \{c, d\}, \{K_1, c\}, \{K_1, c, d\}\}$. With size two, K_1 is both a string of sausages and a fat sausage; by convention we treat it as a fat sausage* (i.e., *of type* P*). The updated $\mathcal{C}(\boldsymbol{\sigma})$ is a string of sausages itself, so the algorithm terminates by picking the root of the tree as a type* Q *node. The resulting PQ-tree is illustrated in Figure 4 (left).*

We now formulate two useful properties of PQ-trees, as observed by Cornaz et al. [16].

Lemma 11 (Cornaz et al. 16)**.** *PQ-trees can be constructed in $O(|N| \cdot |A|^3) = O(nm^3)$ time. Further, given $\boldsymbol{\sigma}$ and its PQ-tree T, a set of candidates $K \subseteq A$ is a clone set if and only if it satisfies one of the following: (1) K exactly corresponds to the leaves of a subtree in T, or (2) K exactly corresponds to the leaves of a set of subtrees the roots of which are adjacent descendants of a Q-node in T.*

Cornaz et al. [16] use PQ-trees to prove fixed-parameter tractability of computing a *Kemeny ranking* of a profile, which obeys a special case of CC (see Appendix C for CC properties of social preference functions, which return aggregate rankings over A rather than subsets). Similarly, Brandt et al. [5] show that any CC tournament solution is fixed-parameter tractable with respect to the properties of an analogous construct for tournaments (decomposition trees) by running the tournament solution recursively on the nodes of the tree. Our key observation, which we show next, is that even if we start with an SCF that does *not* satisfy CC (or any weaker version thereof), running it on the PQ-tree defines a *new* SCF that is in fact CC, while maintaining many desirable properties of the original SCF.

4.2 CC-Transformed SCFs

We now present Algorithm 1, based on an implementation of CC tournament solutions by Brandt et al. [5]. Given $T = PQ(\boldsymbol{\sigma})$ (the PQ-tree for a profile $\boldsymbol{\sigma}$), we refer to its nodes by the subset of candidates in

Algorithm 1: CC transformation for SCF

Input: SCF f, preference profile $\boldsymbol{\sigma}$ over candidates A
Output: Winner candidates $W \subseteq A$

$W = \emptyset$; // Winner list, initialized as empty
$T = PQ(\boldsymbol{\sigma})$; // Constructs the PQ-tree for $\boldsymbol{\sigma}$
$\mathcal{Q} = (A)$; // Queue of nodes, starting with root node
while $|\mathcal{Q}| \neq 0$ **do**
  $B = \text{Dequeue}(\mathcal{Q})$;
  if $|B| = 1$ **then** $W = W \cup B$; // B is a leaf node, add it to winners
  else
    $\boldsymbol{\sigma}_o = \boldsymbol{\sigma}|_B$; $\mathcal{K} = \text{decomp}(B, T)$; // Each $B' \in \mathcal{K}$ is a child node of B
    if *is_p_node*(B, T) **then** // B is a P-node
      for $K \in f(\boldsymbol{\sigma}_o^{\mathcal{K}})$ **do** Enqueue$(\mathcal{Q}, K)$; // Run f on summary, enqueue winners
    else // B is a Q-node
      $W' = f(\boldsymbol{\sigma}_o^{\mathcal{K}}|_{\{B_1(B,T), B_2(B,T)\}})$; // Run f on the first two child nodes
      if $W' = \{B_1(B, T)\}$ **then**
        Enqueue$(\mathcal{Q}, B_1(B, T))$ // Enqueue the first child of B
      else if $W' = \{B_2(B, T)\}$ **then**
        Enqueue$(\mathcal{Q}, B_{|\mathcal{K}|}(B, T))$ // Enqueue the last child of B
      else // $W' = \{B_1(B, T), B_2(B, T)\}$
        for $K \in \mathcal{K}$ **do** Enqueue$(\mathcal{Q}, K)$; // Enqueue *all* children of B

their subtrees. For $B \subseteq A$, is_p_node(B, T) returns True (resp. False) if the node corresponding to B in T is a P- (resp. Q-) node, raising an error if no such node exists. decomp(B, T) returns the decomposition $\mathcal{K}$ corresponding to node B, where each $K \in \mathcal{K}$ is a child node of B (these are clone sets by Lemma 11). If T is the tree from Fig. 4 (left), decomp$(A, T) = \{\{a, b\}, \{c\}, \{d\}\}$, and decomp$(\{a, b\}, T) = \{\{a\}, \{b\}\}$. For a Q-node B, let $B_i(B, T)$ be the i-th child of B according to its majority ranking σ_1.

Definition 12. Given an SCF f, the *CC-transform* of f is an SCF f^{CC} that, on input profile $\boldsymbol{\sigma}$, outputs the candidates consistent with the output of Algorithm 1 on input f and $\boldsymbol{\sigma}$.

Intuitively, f^{CC} recursively runs f on the PQ-tree of $\boldsymbol{\sigma}$, starting at the root. At every P-node B, f^{CC} runs f on the summary induced by that node ($\boldsymbol{\sigma}^{\text{decomp}(B,T)}$), and continues with the winner children. At every Q-node B, it runs f on the summary of the node *restricted* to its first two child nodes ($B_1(B, T)$ and $B_2(B, T)$). If the winner is $B_1(B, T)$ (resp. $B_2(B, T)$), it continues with the first (resp. last) child node of B; if both are winners, then f continues with *all* the children of B. The intuition for this is that for any Q-node B of T, the pairwise relationship between $B_i(B, T)$ and $B_j(B, T)$ is the same for all $i < j$, so if $B_1(\boldsymbol{\sigma}, \mathcal{K})$ defeats $B_2(\boldsymbol{\sigma}, \mathcal{K})$ according to f (in a pairwise comparison), it will also defeat $B_j(\boldsymbol{\sigma}, \mathcal{K})$ for any $j > 1$ by the neutrality of f. If $B_2(\boldsymbol{\sigma}, \mathcal{K})$ defeats $B_1(\boldsymbol{\sigma}, \mathcal{K})$ according to f, on the other hand, then $B_j(\boldsymbol{\sigma}, \mathcal{K})$ will defeat $B_i(\boldsymbol{\sigma}, \mathcal{K})$ for any $j > i$ by the neutrality of f, naturally leading us to the last child node. Lastly, if both $B_1(\boldsymbol{\sigma}, \mathcal{K})$ and $B_2(\boldsymbol{\sigma}, \mathcal{K})$ are winners, this implies f cannot choose between any pair of child nodes of B, which is why we continue with all child nodes.

We will shortly show that f^{CC} satisfies CC, even if f fails it. Of course, a useless transformation like $f^{CC'}(\boldsymbol{\sigma}) = A$ for all $\boldsymbol{\sigma}$ would also achieve this. As such, we want to show that f^{CC} *preserves* some of f's desirable properties. It is straightforward to see that anonymity and neutrality are preserved, as Algorithm 1 is robust to relabeling of candidates and/or voters. Further, as we will show, Condorcet and Smith consistency, as well as decisiveness, are among the preserved properties. Unfortunately, f^{CC} does not preserve monotonicity, independence of Smith-dominated alternatives (ISDA), or participation. This is since changing an existing vote or adding a new candidate/voter can alter the clone structure of $\boldsymbol{\sigma}$, and thus its PQ-tree. We introduce relaxations of these axioms that require robustness against

changes that *respect* the clone structure. We first define the relaxation of monotonicity.

Definition 13. An SCF f satisfies *clone-aware monotonicity (monotonicity^{ca})* if $a \in f(\boldsymbol{\sigma})$ implies $a \in f(\boldsymbol{\sigma}')$ whenever (1) $\mathcal{C}(\boldsymbol{\sigma}) = \mathcal{C}(\boldsymbol{\sigma}')$ and (2) for all $i \in N$ and $b, c \in A \setminus \{a\}$, we have $a \succ_{\sigma_i} b \Rightarrow a \succ_{\sigma_i'} b$ and $b \succ_{\sigma_i} c \Rightarrow b \succ_{\sigma_i'} c$.

The only difference between Definition 13 and the usual definition of monotonicity (*i.e.*, promoting a winner in some votes should not cause them to lose) is the requirement that $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma}'$ have the same clone structure. ISDA^{ca} and participation^{ca} are defined analogously; see Appendix E.3 for formal definitions and examples showing why we need these relaxations. These new axioms implicitly assume that clone structures are *inherent*, based on candidates' location in some shared space (in line with the original interpretation by Tideman [62]), so any "realistic" change to $\boldsymbol{\sigma}$ will not alter its clone sets.

Lastly, in order to analyze the computational complexity of f^{CC}, we introduce the *decomposition degree* of a tree, which we adapt from the definition of the decomposition degree of a tournament introduced by Brandt et al. [5]. Following their fixed-parameter tractability framework, we will state the runtime of Algorithm 1 in terms of a parameter δ (which corresponds to the decomposition degree of a PQ-tree, formalized below in Definition 14) and the running time of the input SCF f.

Definition 14. Given a PQ-tree T for a profile $\boldsymbol{\sigma}$, let $\mathcal{P}$ denote the set of P-nodes in T. The *decomposition degree* $\delta(T)$ of T is defined as $\delta(T) = \max_{B \in \mathcal{P}} |\text{decomp}(B, T)|$ if $\mathcal{P} \neq \emptyset$ and $\delta(T) = 2$ otherwise.

Intuitively, $\delta(T)$ is the maximum number of candidates with which Algo. 1 will run f; *e.g.*, if T is the left (resp. right) PQ-tree from Fig. 4, $\delta(T)$ is 2 (resp. 3). We now present our main result on CC-transforms.

Theorem 3. *For any neutral SCF f, f^{CC} satisfies: (1) If $\boldsymbol{\sigma}$ has no non-trivial clone sets, $f^{CC}(\boldsymbol{\sigma}) = f(\boldsymbol{\sigma})$; (2) f^{CC} is composition-consistent; (3) If f is composition-consistent, then $f^{CC} = f$,* i.e., *they agree for all $\boldsymbol{\sigma}$; (4) If f satisfies any of {anonymity, Condorcet consistency, Smith consistency, decisiveness (on all $\boldsymbol{\sigma}$), monotonicity^{ca}, ISDA^{ca}, participation^{ca}}, then f^{CC} satisfies this property as well; (5) Let $g(n, m)$ be an upper bound on the runtime of an algorithm that computes f on profiles with n voters and m candidates; then, $f^{CC}(\boldsymbol{\sigma})$ can be computed in time $O(nm^3) + m \cdot g(n, \delta(PQ(\boldsymbol{\sigma})))$.*

Taken together, (2) and (3) immediately imply that our CC transformation is idempotent, *i.e.*, $(f^{CC})^{CC} = f^{CC}$ for all $\boldsymbol{\sigma}$. Further, (5) from Theorem 3 implies if f is polynomial-time computable, then so is f^{CC}. Even if f is not polynomial-time computable, (5) in Theorem 3 gives us a running time that depends on the decomposition degree $\delta(T)$ of the PQ-tree. Therefore, we obtain fixed-parameter tractability for f^{CC} (in terms of $\delta(T)$) for all (neutral) SCFs f with runtime that is polynomial in n. For example, this includes SCFs that are NP-hard to compute when the number of candidates m is arbitrarily large, but is polynomial-time computable for constant m, such as the (anonymous) RP [9]. Moreover, by (3) in Theorem 3, fixed-parameter tractability also holds for SCFs that are CC to begin with.

Despite the above theoretical guarantees of Algorithm 1 one can ask how useful our CC-transform is *in practice* as it does not modify the SCF unless actual clone sets exist (by (1) of Theorem 3). While clone sets are unlikely in political elections (where the number of candidates is reasonably bounded anyway), they may easily occur in settings where candidates or voters are not human. For example, if the candidates are AI outputs—*e.g.*, for reinforcement learning from human feedback (RLHF)—it is easy to introduce minorly tweaked versions of the same output into the evaluation process (Section 6 discusses why our results may be highly relevant for RLHF). Further, voters too could be not human [64]. For example, if voters are benchmarks against which we are testing AI models (and we are supposed to choose a model by aggregating the ranking resulting from each benchmark) [37], variants of the same model are likely to have similar performance on each benchmark. More classically, meta-search engines aggregate results from various ranking algorithms, each of which plays the role of a voter [21], and cloned webpages are likely to be ranked together by each algorithm. The guarantees of Theorem 3 can be even more critical in settings such as these, as (a) the cost of cloning can be arbitrarily low, making

it all the more important that the SCF used cannot be manipulated by such clones, and (b) the number of candidates can be very large due to such cloning, making tractability a significant concern.

Before ending our discussion of CC-transforms, it is worth comparing f^{CC} to two similar notions from prior literature. First, in addition to CC, Laffond et al. [36] defined the *CC hull* of an SCF f: the smallest (by inclusion) CC solution containing f. However, the CC hull does not necessarily preserve Condorcet consistency and achieves CC by adding candidates to the returned set, which sacrifices decisiveness. Second, in an unpublished preprint, Heitzig [28] introduces a similar recurrent CC transformation for SCFs. However, his transformation does not specify the order in which clone sets need to be collapsed and requires the original SCF to satisfy additional axioms, *e.g.*, Condorcet consistency and anonymity.

5 Obvious Independence of Clones

We now investigate *strategic behavior* under IoC/CC rules. Unlike Elkind et al. [22], who study strategic cloning when each clone set has a central "manipulator," we focus on strategies of individual candidates, who can personally decide whether to run in the election. Our motivation for this is that, as seen in Section 4, a single candidate can be a member of multiple non-trivial (potentially nested) clone sets of different sizes, and its preference over these sets may vary. To formalize this intuition, given $\boldsymbol{\sigma}$, for each $a, b \in A$, define $d_{\boldsymbol{\sigma}}(a, b) = |K_{ab}| - 1$, where K_{ab} is the smallest clone set containing both a and b.

Proposition 15. *For any $\boldsymbol{\sigma}$, $d_{\boldsymbol{\sigma}}$ is a metric over the candidate set A.*

Now, given $\boldsymbol{\sigma}$ and an SCF f, consider a normal-form game $\Gamma^f_{\boldsymbol{\sigma}}$ where the players are the candidates, and each has two actions: run (R) and drop out (D). For simplicity, we assume f is decisive; *i.e.*, it outputs a single winner. If exactly $S \subseteq A$ play R, the utility of any $a \in A$ is a (strictly) decreasing function of $d_{\boldsymbol{\sigma}}(a, f(\boldsymbol{\sigma}|_S))$. This follows from the assumption that clones represent proximity in some space (*e.g.*, for political elections, this could be ideological): the closer the winner is to a candidate, the happier that candidate is. If *all* candidates pick D, the election is void, which gives everyone the worst utility.

An action is a *dominant strategy* of player a if it brings (weakly) higher utility than any other action, no matter how a's opponents play. A (pure) strategy profile $\boldsymbol{s} = (s_a)_{a \in A}$ specifies an action $s_a \in \{R, D\}$ for each player $a \in A$. We say $\boldsymbol{s}$ is a *pure-strategy Nash equilibrium (PNE)* if no player $a \in A$ can strictly increase her utility by unilaterally changing her action [47].

The setting of $\Gamma^f_{\boldsymbol{\sigma}}$ is a restriction of the more general *strategic candidacy* model introduced by Dutta et al. [19], where candidates also have a preference over each other, and accordingly choose to run or not. Since $d_{\boldsymbol{\sigma}}(a, b) = 0$ if and only if $a = b$, our setting fulfills the condition of *self-supporting* preferences (*i.e.*, all candidates like themselves the best), which is taken as a natural domain restriction by Dutta et al. An SCF is called *candidate stable* if for all profiles, the action profile where *all* candidates are running is a PNE. For example, in our setting with $\Gamma^f_{\boldsymbol{\sigma}}$, PV is not candidate stable.

Example 16. *Consider $\Gamma^{PV}_{\boldsymbol{\sigma}}$, where $\boldsymbol{\sigma}$ is the profile from Figure 2 (left). In the strategy profile in which all candidates play R (say $\boldsymbol{s}^R$), b wins. However, if a_2 deviates to D, then a_1 wins. Since $d_{\boldsymbol{\sigma}}(a_1, a_2) = 1 < 3 = d_{\boldsymbol{\sigma}}(b, a_2)$, this deviation increases the utility of a_2, proving $\boldsymbol{s}^R$ is not a PNE.*

Crucially, Dutta et al. [19] show that with no restrictions on the preferences of voters or of candidates (except the latter being self-supporting), the *only* SCF that is both unanimous (a candidate is picked if all voters rank her first) and candidate-stable is *dictatorship* (a single voter decides the outcome). As we show next, this is not the case in our restricted setting where $d_{\boldsymbol{\sigma}}$ dictates the preferences of candidates.

Proposition 17. *If f is IoC, then R is a dominant strategy in $\Gamma^f_{\boldsymbol{\sigma}}$ for all candidates.*

Thus, in our setting, IoC rules not only achieve candidate stability, but strengthen it, as all candidates running is a *dominant-strategy Nash equilibrium*. Proposition 17 is in line with prior results showing

how restricting the preferences of voters and candidates (*e.g.*, having a Condorcet winner [19], or single-peaked preferences [55]) can circumvent the impossibility result by Dutta et al. [19]. We require that all preferences are consistent with *some* clone structure, but since any voter profile has its own clone structure, this effectively puts no restriction on the preferences of the voters, but only on those of the candidates. Proposition 17 formalizes the interpretation of IoC as "strategy-proofness for candidates," in the sense that a candidate willing to run will not drop out due to fear of hurting like-minded candidates.[6]

However, as demonstrated by Li [44], the benefits of a property of a mechanism might only be materialized if the agents actually *believe* that the property does indeed hold. If it is not obvious to a candidate that a given SCF is indeed IoC, she might still drop out of the race (even though running is her dominant strategy), either out of fear of hurting her party, or of being blamed by the voters for doing so.

In order to characterize the "obviousness" of IoC, we turn to *obviously dominant strategies* [44], which inherently deal with extensive-form games (EFG), where players take actions in turns. An EFG can be represented by a rooted tree; at each node, the associated player takes an action, each leading to a child node. The nodes of each player are partitioned into *information sets*; a player cannot tell apart nodes in the same information set. Below, we introduce obviously dominant strategies informally for games where each player acts once; the full definition (along with that of EFGs) is in Appendix F.3.

Definition 18 (Li [44], Informal)**.** An action s is *obviously dominant* for player a if for any other action s', starting from the point in the game when a must take an action, the best possible outcome from s' is no better than the worst possible outcome from s.

Here, the "best" and "worst" outcomes are defined over the actions of candidates that act along with or after a. For example, interpreting $\Gamma^f_{\boldsymbol{\sigma}}$ from above as an EFG where all candidates act simultaneously, running (R) may *not* be an obviously dominant strategy, even if f is IoC, as the next example shows.

Example 19. *Consider $\Gamma^{STV}_{\boldsymbol{\sigma}}$, where $\boldsymbol{\sigma}$ is from Fig. 2. From the perspective of a_1, the* worst *outcome of running (R) is if a_2 and c also play R, but b plays D, making c the winner. The* best *outcome of a_1 dropping out (D) is if everyone else plays R, making a_2 win. Since $d_{\boldsymbol{\sigma}}(a_1, c) = 3 > 1 = d_{\boldsymbol{\sigma}}(a_1, a_2)$, R is* not *an obviously-dominant strategy for a_1. A tree representation of $\Gamma^{STV}_{\boldsymbol{\sigma}}$ is in Figure 8 (Appendix F.5).*

What if we had used a composition-consistent SCF f instead? Recall that by (3) of Theorem 3, f can be implemented using Algorithm 1, *i.e.*, by running it on the PQ-tree of the input profile. The key observation is that when running Algorithm 1, we can postpone asking a candidate if she is running until we reach the parent node of that candidate. More formally, consider then an alternate EFG $\Lambda^f_{\boldsymbol{\sigma}}$ where the actions and utilities are the same as $\Gamma^f_{\boldsymbol{\sigma}}$, but the winner is determined by running Algorithm 1 on inputs f and $\boldsymbol{\sigma}$, with the following process after each node B is dequeued from $\mathcal{Q}$:

• If B is a P-node, then all the children of B that are actual candidates (*i.e.*, leaf nodes) are asked (simultaneously) to pick R or D. Given S' are the child nodes that chose D, f is run on $\boldsymbol{\sigma}^{\text{decomp}(B,T)} \setminus S'$ to decide which branch to follow. If this branch is a leaf, the game is over, with the leaf being the winner.

• If B is a Q-node, say $W' = f(\boldsymbol{\sigma}^{\mathcal{K}})|_{\{B_1(B,T), B_2(B,T)\}}$. If $W' = B_1(B,T)$ and $B_1(B,T)$ is an internal node, then it is enqueued. Otherwise, the (single) candidate corresponding to $B_1(B,T)$ is asked to pick R, in which case it is the winner, or D, in which case the process is repeated with $B_2(B,T), B_3(B,T), \ldots$ until either an internal node or a candidate that plays R is encountered. If $W' = B_2(B,T)$, the identical process is followed, except starting from $B_{|\text{decomp}(B,T)|}(B,T)$ and moving backwards.

• In either case, if all the children of B are leaf nodes and all play D, the algorithm moves back to its parent node, and repeats the computation there (without re-asking any leaf nodes) with B dropped out.

Intuitively, $\Lambda^f_{\boldsymbol{\sigma}}$ asks each candidate whether she is running only when this decision becomes relevant. Just like in $\Gamma^f_{\boldsymbol{\sigma}}$, each player in $\Lambda^f_{\boldsymbol{\sigma}}$ has a single information set, since she is not aware of the actions

[6] This is not the only advantage of an IoC/CC rule; *e.g.*, candidates also cannot make their opponents' clone set lose by nominating more clones in this set. In our model, we focus on the choice of whether to run.

of the players that are acting before or simultaneously with her; she only knows her parent node is reached. The winner in $\Lambda^f_{\boldsymbol{\sigma}}$ is precisely the winner of applying f^{CC} directly to $\boldsymbol{\sigma} \setminus S'$, where S' are the players that picked D.[7] If f is CC to begin with, this exactly corresponds to $f(\boldsymbol{\sigma} \setminus S')$ by Theorem 3(3). Figure 9 in Appendix F.5 shows the game tree for $\Lambda^f_{\boldsymbol{\sigma}}$ for $\boldsymbol{\sigma}$ from Figure 3 and $f = STV^{CC}$.

Crucially, this implementation of f^{CC} allows us to strengthen Prop. 17, achieving obviousness.

Theorem 4. *For any neutral f, R is an obviously-dominant strategy in $\Lambda^f_{\boldsymbol{\sigma}}$ for all candidates.*

The proof relies on the observation that in $\Lambda^f_{\boldsymbol{\sigma}}$, when a candidate is asked to decide between R and D, Algorithm 1 has already reached her parent node, which is the smallest non-trivial clone set containing her. Thus, the best case of D and worst case of R are both one of her second-favorite group of candidates (after herself) winning, achieving obvious strategy-proofness (OSP). Theorem 4 has strong practical implications: since any CC rule can be implemented with Algorithm 1, the decision of a candidate to drop out of an election can be postponed until *after* she learns whether her smallest clone set has won. Hence, using a CC rule, the election result will not change if we replace the candidates' names on the ballots with party names, and hold in-party primaries for the winners afterwards. In contrast, with rules that are just IoC, the results within the party vary based on whether internal primaries are held (Example 6). Without primaries, CC rules can also derive clone sets *a posteriori* from the votes.

Obviousness is also relevant for contexts where candidates (or, for settings with abstract candidates, their deployers) are perfectly capable of reasoning about an SCF and its properties, but they worry about manipulation of the SCF by the entity implementing it (*agenda control* [38]). As shown by Li [44], choice rules that are OSP-implementable offer a significant advantage in these settings, as they are exactly those that are supported by *bilateral commitments* (partial commitments by the planner such that, if violated, those violations can be observed by the agents themselves without communicating). In the context of $\Lambda^f_{\boldsymbol{\sigma}}$, instead of committing to using a specific SCF, the planner can commit to each candidate he interacts with that, if she decides to run, the winner will be some member of her smallest non-trivial clone set. This (1) is enough to convince the candidate to run and (2) ensures that a violation of this commitment can be observed by the candidate (by looking at the outcome of the election).

In general, the connection we establish between IoC and CC yields new and natural interpretations of the two properties: we can view CC as a way of exposing the *obviousness* of IoC.

6 Conclusion and Future Work

An exciting direction of future work is to study the role that IoC and CC can play in AI alignment. Methods such as reinforcement learning from human feedback (RLHF) require aggregating data representing diverse human opinions, for which social choice methods are well-suited. It is relatively easy to copy AI model responses, or even entire models, and perform small tweaks to them (*e.g.*, via fine-tuning). Such tweaks are unlikely to outperform other significantly better models, hence forming a clone set. Xu et al. [63] demonstrate that using non-IoC aggregation rules for RLHF can result in egregious behavior,[8] a result that is especially concerning as standard RLHF approaches implicitly use Borda Count [58], which fails IoC. Thus, as pointed out by Conitzer et al. [15], IoC (and thereby CC) are highly relevant for social choice for AI models. In line with this agenda, Procaccia et al. [52] have recently studied how existing RLHF algorithms can be modified to increase their robustness against clones. A natural strengthening of this goal for future work is developing RLHF approaches that implicitly use CC rules.

[7] There is a slight caveat here: the leaf nodes that are children of internal nodes that never got visited did not get to play R or D in $\Lambda^f_{\boldsymbol{\sigma}}$. Any choice these candidates could have made does not change the result of f as long as *at least one* candidate of each non-trivial clone set were to pick R. This is in line with the assumption made by Elkind et al. [22] that at least one clone of each clone set will be in the election. This is not a far-fetched assumption: in practice, it is the leadership of a political party that decides to participate in an election, before individual members of the party make up their mind about running.

[8] While the Xu et al. [63] present these undesirable outcomes as a failure to meet the independence of irrelevant alternatives property by Luce [45], the demonstrated pathology would also be prevented if the aggregation rule being used was IoC.

References

[1] Niclas Boehmer, Robert Bredereck, and Dominik Peters. Rank aggregation using scoring rules. *arXiv preprint arXiv:2209.08856*, 2022.

[2] Kellogg S Booth and George S Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13(3): 335–379, 1976.

[3] Florian Brandl, Felix Brandt, and Hans Georg Seedig. Consistent probabilistic social choice. *Econometrica*, 84(5):1839–1880, 2016.

[4] Felix Brandt. Some remarks on Dodgson's voting rule. *Mathematical Logic Quarterly*, 55(4):460–463, July 2009.

[5] Felix Brandt, Markus Brill, and Hans Georg Seedig. On the fixed-parameter tractability of composition-consistent tournament solutions. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

[6] Felix Brandt, Christian Geist, and Dominik Peters. Optimal bounds for the no-show paradox via SAT solving. *Mathematical Social Sciences*, 90:18–27, 2017.

[7] Felix Brandt, Markus Brill, and Paul Harrenstein. Extending tournament solutions. *Social Choice and Welfare*, 51(2):193–222, 2018.

[8] Markus Brill and Vincent Conitzer. Strategic voting and strategic candidacy. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2015.

[9] Markus Brill and Felix Fischer. The price of neutrality for the ranked pairs method. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2012.

[10] Rosa Camps, Xavier Mora, and Laia Saumell. A continuous rating method for preferential voting. the incomplete case. *Social Choice and Welfare*, 40(4):1111–1142, April 2012.

[11] Christian Capelle, Michel Habib, and Fabien de Montgolfier. Graph decompositions and factorizing permutations. *Discrete Mathematics and Theoretical Computer Science*, 5:55–70, 2002.

[12] Ioannis Caragiannis, Christos Kaklamanis, Nikos Karanikolas, and Ariel D. Procaccia. Socially desirable approximations for dodgson's voting rule. In *Proceedings of the ACM Conference on Economics and Computation (EC)*, 2010.

[13] Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios N. Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael I. Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: an open platform for evaluating llms by human preference. In *Proceedings of the International Joint Conference on Artificial Intelligence (ICML)*, 2024.

[14] Vincent Conitzer, Matthew Rognlie, and Lirong Xia. Preference functions that score rankings and maximum likelihood estimation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

[15] Vincent Conitzer, Rachel Freedman, Jobst Heitzig, Wesley H. Holliday, Bob M. Jacobs, Nathan Lambert, Milan Mossé, Eric Pacuit, Stuart Russell, Hailey Schoelkopf, Emanuel Tewolde, and William S. Zwicker. Position: Social choice should guide AI alignment in dealing with diverse human feedback. In *Proceedings of the International Joint Conference on Artificial Intelligence (ICML)*, 2024.

[16] Denis Cornaz, Lucie Galand, and Olivier Spanjaard. Kemeny elections with bounded single-peaked or single-crossing width. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[17] Théo Delemazure and Dominik Peters. Generalizing instant runoff voting to allow indifferences. In *Proceedings of the ACM Conference on Economics and Computation (EC)*, 2024.

[18] Arnaud Dellis and Mandar Oak. Multiple votes, multiple candidacies and polarization. *Social Choice and Welfare*, 46(1):1–38, 2016.

[19] Bhaskar Dutta, Matthew O. Jackson, and Michel Le Breton. Strategic candidacy and voting procedures. *Econometrica*, 69(4):1013–1037, 2001.

[20] Bhaskar Dutta, Matthew O. Jackson, and Michel Le Breton. Voting by successive elimination and

strategic candidacy. *Journal of Economic Theory*, 103(1):190–218, 2002.

[21] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the World Wide Web Conference*, 2001.

[22] Edith Elkind, Piotr Faliszewski, and Arkadii Slinko. Cloning in elections: Finding the possible winners. *Journal of Artificial Intelligence Research*, 42:529–573, 2011.

[23] Edith Elkind, Piotr Faliszewski, and Arkadii Slinko. Clone structures in voters' preferences. In *Proceedings of the ACM Conference on Economics and Computation (EC)*, 2012.

[24] Edith Elkind, Piotr Faliszewski, Jean-Francois Laslier, Piotr Skowron, Arkadii Slinko, and Nimrod Talmon. What do multiwinner voting rules do? an experiment over the two-dimensional euclidean domain. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

[25] Peter C. Fishburn. Condorcet social choice functions. *SIAM Journal on Applied Mathematics*, 33(3): 469–489, 1977.

[26] Rupert Freeman, Markus Brill, and Vincent Conitzer. On the axiomatic characterization of runoff voting rules. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2014.

[27] Donald B. Gillies. *Solutions to General Non-Zero-Sum Games*, pages 47–86. Princeton University Press, 1959.

[28] Jobst Heitzig. Social choice under incomplete, cyclic preferences. *arXiv preprint arXiv:0201285*, 2002. Unpublished.

[29] Jobst Heitzig and Forest W. Simmons. Some chance for consensus: voting methods for which consensus is an equilibrium. *Social Choice and Welfare*, 38(1):43–57, November 2010.

[30] Wesley H. Holliday. A simple condorcet voting method for final four elections (march 13, 2024). *SSRN Electronic Journal*, 2024.

[31] Wesley H. Holliday and Eric Pacuit. Split cycle: a new condorcet-consistent voting method independent of clones and immune to spoilers. *Public Choice*, 197(1-2):1–62, August 2023.

[32] Alexander Karpov. On the number of group-separable preference profiles. *Group Decision and Negotiation*, 28(3):501–517, 2019.

[33] Alexander Karpov and Arkadii Slinko. Symmetric maximal condorcet domains. *Order*, 40(2): 289–309, September 2022.

[34] Semih Koray and Arkadii Slinko. Self-selective social choice functions. *Social Choice and Welfare*, 31(1):129–149, September 2007.

[35] Justin Kruger and Stéphane Airiau. Permutation-based randomised tournament solutions. In *Proceedings of the European Conference on Multi-Agent Systems (EUMAS)*, 2018.

[36] Gilbert Laffond, Jean Lainé, and Jean-FranÃğis Laslier. Composition-consistent tournament solutions and social choice functions. *Social Choice and Welfare*, 13(1):75–93, 1996.

[37] Marc Lanctot, Kate Larson, Yoram Bachrach, Luke Marris, Zun Li, Avishkar Bhoopchand, Thomas Anthony, Brian Tanner, and Anna Koop. Evaluating agents using social choice theory. *arXiv preprint arXiv:2312.03121*, 2025.

[38] Jérome Lang, Nicolas Maudet, and Maria Polukarov. New results on equilibria in strategic candidacy. In *Proceedings of the International Symposium on Algorithmic Game Theory (SAGT)*, 2013.

[39] Jean-François Laslier. *Tournament Solutions and Majority Voting*. Springer Berlin Heidelberg, 1997.

[40] Jean-François Laslier. Aggregation of preferences with a variable set of alternatives. *Social Choice and Welfare*, 17(2):269–282, March 2000.

[41] Jean-François Laslier. *And the Loser Is... Plurality Voting*, pages 327–351. Springer Berlin Heidelberg, 2012.

[42] Jean-François Laslier and Karine Van der Straeten. Strategic voting in multi-winner elections with approval balloting: a theory for large electorates. *Social Choice and Welfare*, 47(3):559–587, 2016.

[43] Patrick Lederer. Bivariate scoring rules: Unifying the characterizations of positional scoring rules and kemeny's rule. *Journal of Economic Theory*, 218, June 2024.

[44] Shengwu Li. Obviously strategy-proof mechanisms. *American Economic Review*, 107(11):3257–3287, November 2017.

[45] R. Duncan Luce. *Individual Choice Behavior: A Theoretical Analysis*. John Wiley and Sons, New

York, NY, 1959.

[46] Ross M McConnell and Fabien De Montgolfier. Linear-time modular decomposition of directed graphs. *Discrete Applied Mathematics*, 145(2):198–209, 2005.

[47] John Nash. *Non-cooperative games*. PhD thesis, Priceton University, 1950.

[48] Andrew Needham. 1934 election oregon results. Technical Report APD/21/3845, Oregon Secretary of State, Nov 2021.

[49] Svetlana Obraztsova, Edith Elkind, Maria Polukarov, and Zinovi Rabinovich. Strategic candidacy games with lazy candidates. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[50] Z. Emel Öztürk. Consistency of scoring rules: a reinvestigation of composition-consistency. *International Journal of Game Theory*, 49(3):801–831, January 2020.

[51] Maria Polukarov, Svetlana Obraztsova, Zinovi Rabinovich, Alexander Kruglyi, and Nicholas R. Jennings. Convergence to equilibria in strategic candidacy. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, 2015.

[52] Ariel D. Procaccia, Benjamin Schiffer, and Shirley Zhang. Clone-robust ai alignment. *arXiv preprint arXiv:2501.09254*, 2025.

[53] Rob Richie, Jeremy Seitz-Brown, and Lucy Kaufman. The case for instant runoff voting. *Constitutional Political Economy*, 34(3):367–377, January 2023.

[54] Hiroki Saitoh. Characterization of tie-breaking plurality rules. *Social Choice and Welfare*, 59(1): 139–173, January 2022.

[55] Yusuke Samejima. Strategic candidacy and single-peakedness. *The Japanese Economic Review*, 58 (4):423–442, 2007.

[56] Markus Schulze. A new monotonic, clone-independent, reversal symmetric, and condorcet-consistent single-winner election method. *Social Choice and Welfare*, 36(2):267–303, July 2010.

[57] Thomas Schwartz. *The logic of collective choice*. Columbia University Press, 1986.

[58] Anand Siththaranjan, Cassidy Laidlaw, and Dylan Hadfield-Menell. Distributional preference learning: Understanding and accounting for hidden context in RLHF. *arXiv preprint arXiv:2312.08358*, 2024.

[59] John H. Smith. Aggregation of preferences with variable electorate. *Econometrica*, 41(6):1027–1041, 1973.

[60] Krzysztof Sornat, Virginia Vassilevska Williams, and Yinzhan Xu. Fine-grained complexity and algorithms for the schulze voting method. In *Proceedings of the ACM Conference on Economics and Computation (EC)*, July 2021.

[61] N. Tideman. *Collective Decisions and Voting: The Potential for Public Choice (1st ed.)*. Routledge, 2006.

[62] Thorwald Nicolaus Tideman. Independence of clones as a criterion for voting rules. *Social Choice and Welfare*, 4:185–206, September 1987.

[63] Wanqiao Xu, Shi Dong, Xiuyuan Lu, Grace Lam, Zheng Wen, and Benjamin Van Roy. RLHF and IIA: Perverse incentives. *arXiv preprint arXiv:2312.01057*, 2024.

[64] Yixuan Even Xu, Hanrui Zhang, Yu Cheng, and Vincent Conitzer. Aggregating quantitative relative judgments: From social choice to ranking prediction. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[65] T. M. Zavist and T. N. Tideman. Complete independence of clones in the ranked pairs rule. *Social Choice and Welfare*, 6(2):167–173, April 1989.

[66] William S. Zwicker. Introduction to the theory of voting. In Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia, editors, *Handbook of Computational Social Choice*, chapter 2. Cambridge University Press, 2016.

Ratip Emin Berker
Foundations of Cooperative AI Lab, Carnegie Mellon University
Pittsburgh, PA, USA
Email: rberker@cs.cmu.edu

Sílvia Casacuberta
University of Oxford
Oxford, UK
Email: silvia.casacuberta.puig@cs.ox.ac.uk

Isaac Robinson
University of Oxford
Oxford, UK
Email: isaac.robinson@univ.ox.ac.uk

Christopher Ong
Harvard University
Cambridge, MA, USA
Email: christopherong@g.harvard.edu

Vincent Conitzer
Foundations of Cooperative AI Lab, Carnegie Mellon University
Pittsburgh, PA, USA
Email: conitzer@cs.cmu.edu

Edith Elkind
Northwestern University
Evanston, IL, USA
Email: eelkind@gmail.com

A Further Background

A.1 Related Work

Independence of Clones and Composition Consistency. In this section, we give a conceptual overview of the literature on independence of clones (IoC) and composition consistency (CC), and point out potential origins for some inconsistencies. We formalize these claims in Appendix A.2.

In order to identify candidates "close" to one another (at least according to voters), Tideman [62] introduces the notion of *clone sets* and the property of *independence of clones (IoC)* for social choice

functions (SCFs) that are robust to changes in these sets. Seemingly unaware of Tideman's work, Laffond et al. [36] tackle a similar question by introducing the concept of *components* of candidates and the property of *composition consistency (CC)*. Importantly, Laffond et al. provide two separate definitions for components, one for a *tournament* (a single asymmetric binary relationship on candidates) and one for a *preference profile* (individual preferences of voters over candidates), and thus two different definitions of CC for *tournament solutions* and *SCFs*, which respectively map tournaments and profiles to winners. While Laffond et al.'s components for profiles is *equivalent* to Tideman's clone sets, later work has described the former as "more liberal"/"weaker" [15, 30], potentially due to focusing on the definition of components of tournaments instead (See our Example 22).

The relationship between IoC and CC has been similarly unclear, despite the latter being strictly more demanding for SCFs (Proposition 8). Potentially due to SCFs taking a relatively small space in the work of Laffond et al. [36], subsequent papers have primarily studied CC tournament solutions [39, 5, 7], even describing components/CC to be "analogue" of clones/IoC for tournaments [22, 18, 33]. Other works, while identifying a link between IoC and CC, have not been precise on their relationship [4, 50, 34, 41, 43, 54, 24, 10, 42], describing them as "similar" notions [39, 29] or "related" [5].

There are papers that come much closer in identifying CC as a stronger axiom than IoC: working in a more general setting where voters' preferences are neither required to be asymmetric (ties are allowed) nor transitive, Laslier [40] introduces the notion of *cloning consistency*, which he explains is weaker than CC and is the "same idea" as Tideman's IoC in voting theory. However, there are significant differences between Laslier's definition and IoC: first, his "clone set" definition requires every voter being *indifferent* between any two alternatives in the set (as opposed to having the same relationship to all other candidates), and his cloning consistency dictates that if one clone wins, then so must every other member of the same clone set. Another property for tournament solutions named *weak composition consistency* (this time in fact analogous to IoC) is discussed by Brandt et al. [7], Kruger and Airiau [35], and Laslier [39], although none of them point out the connection to Tideman's IoC. Perhaps the work that does the most justice to the relationship between CC and IoC is by Brandl et al. [3], who explicitly state that Tideman's IoC (which they refer to as cloning consistency) is weaker than Laffond et al.'s CC. Since they work with the more general model of probabilistic social choice functions (PSCF), their observation that CC is stronger than IoC applies to our setting (although the adaptation is not obvious); we formalize this in Proposition 8. The PSCFs analyzed by Brandl et al. are all non-deterministic; hence, to the best of our knowledge, no previous work has studied whether IoC SCFs also satisfy CC, which we do in Section 3.

Strategic Cloning. Cloning and voting rules that are IoC are also of interest from a game theory perspective due to their connection to strategic manipulation in elections [60, 23, 17, 12]. While most of (computational) social choice research treats the set of candidates as fixed, cloning inevitably goes beyond this assumption. To study manipulative behavior in settings with varying candidates, Dutta et al. [19, 20] have initiated the study of *strategic candidacy*, where candidates too have preferences over each other and may choose whether to run in the election. They define an SCF as *candidate stable* if no candidate can benefit from not running given that all others are running, and show that this property is failed by every non-dictatorial SCF satisfying unanimity. Subsequent work has analyzed the computational aspects of strategic candidacy in extensions of this model, such as when candidates incur a small cost for running [49], when candidates can decide to rejoin the election after dropping out (albeit with possibly less support) [51], or when both voters and candidates are behaving strategically [8]. Many of these papers allude to, but do not formally define, a connection between strategic candidacy and cloning. We do so in Section 5, where we use the model of strategic candidacy to analyze the strength of robustness of IoC/CC rules against spoilage by clones.

Recently, Holliday and Pacuit [31] have introduced novel robustness criteria they call *immunity to spoilers* and *immunity to stealers*, which study the impact of adding candidates that are not necessarily

clones (but must fulfill some other conditions). These criteria are incomparable in strength to IoC/CC, and while we focus exclusively on the impact of similar candidates (*i.e.*, clones), we believe the methods we develop may be of future interest for studying different types of spoilers.

A.2 Extended Preliminaries

In this section, we give a more extended analysis of the concepts and definitions introduced by Tideman [62] and by Laffond et al. [36]. To give a more complete picture, this section repeats some of the information given in Section 2 (Preliminaries) of the main body of the paper.

Preference profiles and clones We consider a finite set of *voters* $N = \{1, \dots, n\}$ and a finite set of *candidates* A with $|A| = m$. A *ranking* over A is an asymmetric, transitive, and complete binary relation $\succ$ on A; we denote the set of all rankings over A by $\mathcal{L}(A)$. Each voter $i \in N$ has a *ranking* $\sigma_i \in \mathcal{L}(A)$; we write $a \succ_i b$ to indicate that i ranks a above b, and collect the rankings of all voters in a *preference profile* $\boldsymbol{\sigma} \in \mathcal{L}(A)^n$. Given $\boldsymbol{\sigma}$, how can we identify candidates that are "close" to one another, at least from the point of view of the voters? Tideman [62] addressed this question:

Definition 20 (Tideman 62, §I)**.** Given a preference profile $\boldsymbol{\sigma}$ over candidates A, a nonempty subset of candidates $K \subseteq A$ is a *set of clones* with respect to $\boldsymbol{\sigma}$ if no voter ranks any candidate outside of K between any two elements of K.

Note that all profiles ways have two types of *trivial* clone sets:[9] (1) the entire candidate set A, and (2) for each $a \in A$, the singleton $\{a\}$. We call all other clone sets *non-trivial*. For example, in the preference profile in Figure 5, the only non-trivial clone set is $\{b, c\}$.

Seemingly unaware of Tideman's definition, Laffond et al. [36] tackled a similar question of identifying similar candidates. First, they focused on the context of a *tournament* T, which is a complete asymmetric binary relation on the candidates A (*i.e.*, a ranking with the transitivity condition relaxed). For a tournament, they defined:

Definition 21 (Laffond et al. 36, Def. 1)**.** Given a tournament T over candidates A, a nonempty subset of candidates $C \subseteq A$ is a *component* of T if for all $y, y' \in C$ and all $x \in A \setminus C$: $y \succ_T x \Leftrightarrow y' \succ_T x$.

Of course, any preference profile $\boldsymbol{\sigma}$ with odd number of voters (to avoid ties) can be interpreted as a tournament $T_{\boldsymbol{\sigma}}$ over the same set of candidates, where the binary relation is given by the pairwise defeats of $\boldsymbol{\sigma}$; *i.e.*, $\forall a, b \in A$:

$$a \succ_{T_{\boldsymbol{\sigma}}} b \Leftrightarrow |\{i \in N : a \succ_{\sigma_i} b\}| > |\{i \in N : b \succ_{\sigma_i} a\}|.$$

Considering the similarities between Definitions 1 and 21 (they both group up candidates that have an identical relationship to other candidates), one might expect them to respect this transformation; that is, for K to be a clone set of $\boldsymbol{\sigma}$ if and only if it is a component of $T_{\boldsymbol{\sigma}}$. However, this is not the case, as we demonstrate next.

Voter 1	Voter 2	Voter 3
b	c	a
c	b	c
a	d	b
d	a	d

Figure 5: A preference profile. Columns show rankings.

[9] Tideman [62] in fact excludes trivial clone sets. We use the definition followed by Elkind et al. [23].

Example 22. *Consider again the profile in Figure 5. Here, $\{a, b\}$ is a component in $T_{\boldsymbol{\sigma}}$ (they both defeat d and both lose to c), but they are not a set of clones in $\boldsymbol{\sigma}$, since, for example, Voters 1 ranks c between them. The intuition behind this is that when interpreting a preference profile as a tournament, we lose information about the preferences of individual voters. Indeed, it is easy to see that the implication holds in one direction: if K is a set of clones of $\boldsymbol{\sigma}$, then it is a component of $T_{\boldsymbol{\sigma}}$.*

However, Laffond et al. [36] introduce a separate definition for components for a preference profile or, more accurately, to the more general notion of a *tournament profiles* $(T_i)_{i \in \boldsymbol{\sigma}}$, where voters are allowed to submit tournaments (*i.e.*, votes need not be transitive):

Definition 23 (Laffond et al. 36, Def. 4)**.** Given tournament profile $\boldsymbol{T} = (T_i)_{i \in N}$, a nonempty subset of candidates $C \subseteq A$ is a *component* of $\boldsymbol{T}$ if C is a component of T_i for all $i \in N$.

One can see from Definition 21 that if a tournament is transitive (*i.e.*, a ranking), then C is a component of T if and only if it appears as an interval in that ranking. As such, given a tournament profile $\boldsymbol{T} = \{T_i\}_{i \in N}$ where all voters submit transitive tournaments (*i.e.*, rankings), C is a component of $\boldsymbol{T}$ if and only if it appears as an interval in the vote of every voter. As such, Laffond et al.'s Definition 23 restricted to preference profiles is in fact *equivalent* to Tideman's definition of a set of clones (Definition 1)! For instance, in the profile from Figure 5, $\{a, b\}$ is *not* a component according to Definition 23, since it is not a component of Voter 1's ranking (or of any other voter's, although one is enough to disqualify).

Social choice functions and axioms Of course, identifying "similar" candidates in a voting profile is useless unless one can say something meaningful about their impact on the election result. This impact depends on the voting rule we are using to compute the winners. More formally, say $\mathcal{P}(A)$ is the power set of A (set of all subsets). Then, a *social choice function (SCF)* is a function $f : \mathcal{L}(A)^n \to \mathcal{P}(A)$ that maps each preference profile $\boldsymbol{\sigma}$ to a subset of A, which are termed the *winner(s)* of $\boldsymbol{\sigma}$ under f. In an election without ties, the output of an SCF contains a single candidate.

Before introducing axioms for robustness against strategic cloning, it is worth noting that for SCFs that are *not* robust, the exact influence of addition of similar candidates can vary: for example, introducing clones of a candidate can hurt that candidate for an SCF like plurality voting (which simply picks the candidate(s) ranked first by the most voters) by splitting the vote (as was the case from the Oregon governor race from the introduction of the main body of the paper), making plurality what we call *clone-negative*. On the other hand, having clones can somtimes help a candidate win if the SCF being used is *Borda count*, which gives each candidate one point for every other candidate it beats in each voter's ranking, and picks the candidate(s) with the most points as the winners.

Example 24. *Consider the following voting profile for candidates a and b:*

62 voters	*38 voters*
a	b
b	a

In this case, candidate a receives 62 Borda points, whereas candidate b receives 38 Borda points. Thus, candidate a wins the election. Next we introduce a clone of b, obtaining the following voting profile:

62 voters	*38 voters*
a	b
b	b_2
b_2	a

Now candidate a receives 124 Borda points, b receives 138 Borda points, and b_2 receives 38 Borda points. Hence, candidate b now becomes the winner.

Example 24 shows that unlike with plurality voting, having a clone can positively impact a candidate under Borda count, thus making Borda count *clone-positive* for this specific profile (in order profiles, having clones can in fact hurt your Borda score). Either of these impacts are undesirable, considering they incentivize strategic nomination, either of candidates similar to one's opponents, or of candidates similar to one's self, both of which can be arbitrarily easy. As such, we would like to find *axioms* such that if an SCF satisfies them, then they are in some way robust to this type of strategic nomination.

Along with their (equivalent) definitions for similar candidates in preference profiles, Tideman [62] and Laffond et al. [36] each introduce their own axiom for identifying SCFs that behave "desirably" in response to addition/removal of such candidates. Since we will deal with preference profiles (say $\boldsymbol{\sigma}$) with some candidates (say $A' \subset A$) removed, it will be useful to use $\boldsymbol{\sigma} \setminus A'$ to denote the profile obtained by removing the elements of A' from each voter's ranking in $\boldsymbol{\sigma}$ and preserving the order of all other candidates.

We begin with the axiom by Tideman [62], who explicitly identified the goal of achieving robustness against strategic nomination. Zavist and Tideman [65] later presented the definition with more precise language, which is the version we use for clarity.

Definition 25 (Zavist and Tideman 65)**.** A voting rule f is *independent of clones (IoC)* if the following two conditions are met for all profiles $\boldsymbol{\sigma}$ and for all non-trivial clone sets $K \subset A$ with respect to $\boldsymbol{\sigma}$:

1. For all $a \in K$, we have:
$$K \cap f(\boldsymbol{\sigma}) \neq \emptyset \Leftrightarrow K \setminus \{a\} \cap f(\boldsymbol{\sigma} \setminus \{a\}) \neq \emptyset.$$
2. For all $a \in K$ and all $b \in A \setminus K$ we have:
$$b \in f(\boldsymbol{\sigma}) \Leftrightarrow b \in f(\boldsymbol{\sigma} \setminus \{a\}).$$

Intuitively, IoC dictates that deleting one of the clones must not alter the winning status of the set of clones as a whole, or of any candidate not in the set of clones. This is a desirable property in SCFs, since it imposes that the winner must not change due to the addition of a non-winning candidate who is similar to a candidate already present. This prevents candidates from influencing the election by nominating new copy-cat candidates.

Example 26. *Consider running plurality voting (PV) on the profile $\boldsymbol{\sigma}$ in Figure 2 (left). We have $f_{PV}(\boldsymbol{\sigma}) = \{b\}$, as it is the top choice of 4 voters, more than any other candidate. Moreover, $\{a_1, a_2\}$ is a clone set with respect to $\boldsymbol{\sigma}$. However, $f_{PV}(\boldsymbol{\sigma} \setminus \{a_2\}) = \{a_1\}$, since with a_2 gone, a_1 now has 5 voters ranking it top, beating b (Figure 2, right). This violates both conditions 1 and 2 from Definition 25, as the removal of a clone (a_2 from clone set $\{a_1, a_2\}$) results in another clone (a_1) winning, while previously none did, and eliminates a previous winner outside the clone set (b).*

Instead, consider running Single Transferable Vote (STV) on $\boldsymbol{\sigma}$, which is an SCF that iteratively removes the candidates with the least plurality votes from the ballot, returning the last remaining candidate (see Table 1). We have $STV(\boldsymbol{\sigma}) = \{a_1\}$ as a_2 is removed with 2 plurality votes (causing a_1 to now have 5 plurality votes), followed by the c with 3 plurality votes, and finally b with 4 plurality votes. Similarly, $STV(\boldsymbol{\sigma} \setminus \{a_2\}) = \{a_1\}$, since a_2 was going to be the first candidate to be eliminates anyway. Hence, in both cases, a member of the clone set $\{a_1, a_2\}$ wins. This is in line with the fact that STV is IoC [62].

Like Tideman's presentation of IoC, Laffond et al. [36] were also concerned with the manipulability of elections through cloning. However, the core of the presentation of their axiom, aptly named *composition consistency*, focuses on the consistency between applying a rule directly, or applying it through a two-stage mechanism. In order to formalize this mechanism, we introduce a few concepts.

Definition 27. Given a preference profile $\boldsymbol{\sigma}$ over candidates A, a set of sets $\mathcal{K} = \{K_1, K_2, \ldots, K_\ell\}$ where $K_i \subseteq A$ for all $i \in [\ell]$ is a *clone decomposition* with respect to $\boldsymbol{\sigma}$ if

1. $\mathcal{K}$ is a disjoint partitioning of A, *i.e.*: $A = \bigsqcup_{i \in [\ell]} K_i$ and $K_i \cap K_j = \emptyset$ for $i \neq j$, and
2. each K_i is a non-empty clone set with respect to $\boldsymbol{\sigma}$.

A given profile $\boldsymbol{\sigma}$ can have multiple distinct clone decompositions. Indeed, every profile has at least two decompositions: the *null* decomposition $\mathcal{K}_{null} = \{A\}$ and the *trivial* decomposition $\mathcal{K}_{triv} = \{\{a\}\}_{a \in A}$. Given a clone decomposition $\mathcal{K}$ with respect to $\boldsymbol{\sigma}$, for each $i \in N$, say $\sigma_i^{\mathcal{K}}$ is voter i's ranking over the clone sets in $\mathcal{K}$ (which is well defined, since each clone set appears as an interval in σ_i). We call $\boldsymbol{\sigma}^{\mathcal{K}} = \{\sigma_i^{\mathcal{K}}\}_{i \in [n]}$ the *summary* of $\boldsymbol{\sigma}$ with respect to decomposition $\mathcal{K}$, which is a preference profile treating the elements of $\mathcal{K}$ as the set of candidates. Lastly, for each $K \in \mathcal{K}$, say $\boldsymbol{\sigma}|_K$ is $\boldsymbol{\sigma}$ with $A \setminus K$ removed (*i.e.*, $\boldsymbol{\sigma}|_K \equiv \boldsymbol{\sigma} \setminus (A \setminus K)$). We are now ready to introduce *composition products*.

Definition 28 (Composition product)**.** Given any SCF f, the *composition product* function of f is a function Π_f that takes as input a profile $\boldsymbol{\sigma}$ and a clone decomposition $\mathcal{K}$ with respect to $\boldsymbol{\sigma}$ and outputs $\Pi_f(\boldsymbol{\sigma}, \mathcal{K}) \equiv \bigcup_{K \in f(\boldsymbol{\sigma}^{\mathcal{K}})} f(\boldsymbol{\sigma}^K)$.

Intuitively, Π_f first runs the input voting rule f on the summary (as specified by $\mathcal{K}$), "packing" the candidates in each set to treat it as a meta-candidate K_i. It then "unpacks" the clones of each winner clone set, and runs f once again on each. We demonstrate this in the following example.

Example 29. *Once again consider $\boldsymbol{\sigma}$ from Figure 2. Notice $\mathcal{K} = \{K_a, K_b, K_c\}$ with $K_a = \{a_1, a_2\}$, $K_b = \{b\}$ and $K_c = \{c\}$ is a valid clone decomposition with respect to $\boldsymbol{\sigma}$. Figure 3 shows $\boldsymbol{\sigma}^{\mathcal{K}}$ and $\boldsymbol{\sigma}^{K_a}$. We have $STV(\boldsymbol{\sigma}^{\mathcal{K}}) = K_a$ (K_c gets eliminated first followed by K_b) and $STV(\boldsymbol{\sigma}|_{K_a}) = \{a_2\}$, implying $\Pi_{STV}(\boldsymbol{\sigma}, \mathcal{K}) = \{a_2\}$.*

Example 29 demonstrates that $STV(\boldsymbol{\sigma}) \neq \Pi_{STV}(\boldsymbol{\sigma}, \mathcal{K})$ for this specific $\boldsymbol{\sigma}$ and $\mathcal{K}$, showing STV is not *consistent* with respect to this decomposition, even though the winners in two cases are from the same clone set. It is also easy to see that for all rules f and all $\boldsymbol{\sigma}$, we have $f(\boldsymbol{\sigma}) = \Pi_f(\boldsymbol{\sigma}, \mathcal{K}_{null}) = \Pi_f(\boldsymbol{\sigma}, \mathcal{K}_{triv})$. To satisfy composition consistency, a rule must satisfy this equality for all non-trivial decompositions too.

Definition 30 (Laffond et al. 36, Def. 11)**.** An SCF f is *composition-consistent (CC)* if for all preference profiles $\boldsymbol{\sigma}$ and all clone decompositions $\mathcal{K}$ w.r.t. $\boldsymbol{\sigma}$, we have $f(\boldsymbol{\sigma}) = \Pi_f(\boldsymbol{\sigma}, \mathcal{K})$.

Intuitively, an SCF is CC if it chooses the "best" candidates from the "best" clone sets [36]. While any other member of the clone set winning after the removal of a winner clone is sufficient by IoC, CC also specifies which exact clones should be winning. We formalize this hierarchy in Proposition 8 by showing CC implies IoC. Example 29 already demonstrates that the other direction is untrue, as it proves that STV, which is IoC, is not CC. In Section 3 of the main body, we analyze other IoC rules to show whether they are CC.

B Majoritarian SCFs

While *tournaments* (complete and asymmetric binary relationships over A) are not the main focus of this paper, it is worth briefly discussing how our results in Section 3 relate to prior results on CC *tournament solutions* (TSs), which map tournaments to sets of winners. As noted in Appendix A.1, Laffond et al. [36] introduce two separate definitions of components, in tournaments and in profiles (see Definitions 21 and 23 in our Appendix A), and thus two separate definitions of CC for SCFs and

Name of SCF	f	*Description of the SCF's output on input profile* $\boldsymbol{\sigma}$
Schwartz [57] set (GOCHA)	Sz	Given $\boldsymbol{\sigma}$, we say that $B \subseteq A$ is *undominated* if no $a \in A \setminus B$ pairwise defeats (preferred to by a strict majority of voters) any $b \in B$. The winners are the union of minimal (by inclusion) undominated sets.
Smith [59] set (GETCHA)	Sm	Outputs the smallest set of candidates who all pairwise defeat every candidate outside the set.
Uncovered Set [27]	UC_G	Given $\boldsymbol{\sigma}$ and $a, b \in A$, we say that a *left-covers* b if any $c \in A$ that pairwise defeats a also pairwise defeats b. The winners are all $a \in A$ such that there is no $b \in A$ that left-covers *and* pairwise defeats a.

Table 2: Majoritarian SCFs considered in this paper (For non-majoritarian SCFs, see Table 1). Second column indicates our notation for the SCF as a function.

for TSs. Subsequent work has primarily focused on the latter, showing TSs such as uncovered set, the minimal covering set, and the Banks set are CC [36, 39].

Of course, if $|N|$ is odd, the pairwise defeats in $\boldsymbol{\sigma}$ define a tournament, so any TS can be thought of as an SCF that maps $\boldsymbol{\sigma}$ to the winners of this induced tournament. However, for a TS to be well-defined as an SCF (without assuming odd $|N|$), it must be *extended* to cases where the pairwise defeat relationship may contain ties (equivalently, to incomplete tournaments). Such induced SCFs are called *majoritarian*. All three of the SCFs in Table 2 are majoritarian; additionally, like the SCFs in Table 1, they are all known to be IoC (*cf.* Holliday and Pacuit [31]).

As we show next, the axiomatic properties of a TS extended to a SCF may depend on the specific extension. For example, UC_G in Table 2 is an extension of the TS *uncovered set*. Another extension of the same TS follows from the work of Fishburn [25], and is defined as follows (recall that we say a *left-covers* b if any c that pairwise defeats a also pairwise defeats b):

$$UC_F(\boldsymbol{\sigma}) = \{a \in A : \nexists b \in A \text{ such that } b \text{ left covers } a \text{ but } a \text{ does not left-cover } b\}.$$

It can be checked that $UC_F(\boldsymbol{\sigma}) = UC_G(\boldsymbol{\sigma})$ whenever pairwise defeats have no ties, *i.e.*, they are extensions of the same TS. Crucially, even though uncovered set is CC as a TS, UC_F is not even IoC [31]! This demonstrates that **a TS being CC is not sufficient for its SCF extension to be CC**. As we show next, UC_G (Table 2) in fact maintains the CC property.

Proposition 31. *UC_G is CC.*

Proof. Recall from Table 2 that given $\boldsymbol{\sigma}$ and $a, b \in A$, we say that a *left-covers* b in $\boldsymbol{\sigma}$ if any $c \in A$ that pairwise defeats a also pairwise defeats b. Then UC_G is defined as

$$UC_G = \{a \in A : \nexists b \in A \text{ such that } b \text{ left-covers AND pairwise defeats } a\}.$$

Fix any profile $\boldsymbol{\sigma}$ and clone decomposition $\mathcal{K}$ with respect to $\boldsymbol{\sigma}$. We will show that $UC_G(\boldsymbol{\sigma}) = \Pi_{UC_G}(\boldsymbol{\sigma}, \mathcal{K})$. Equivalently, for any $a \in A$, we will show that $a \notin UC_G(\boldsymbol{\sigma}) \Leftrightarrow a \notin \Pi_{UC_G}(\boldsymbol{\sigma}, \mathcal{K})$.

($\Rightarrow$) : Say $a \notin UC_G(\boldsymbol{\sigma})$, then $\exists b \in A$ such that b left-covers and pairwise defeats a in $\boldsymbol{\sigma}$. Say $K_a \in \mathcal{K}$ is the clone set that contains a. We will consider two cases:

1. $b \in K_a$. For each $c \in K_a$ that pairwise defeats b in $\boldsymbol{\sigma}|_{K_a}$, we must have that c pairwise defeats a in $\boldsymbol{\sigma}|_{K_a}$, since b left-covers a in $\boldsymbol{\sigma}$ and deletions of other candidates do not affect pairwise victories of the remaining candidates. Hence, b left-covers and pairwise defeats a in $\boldsymbol{\sigma}|_{K_a}$. This implies $a \notin UC_G(\boldsymbol{\sigma}|_{K_a})$.

2. $b \notin K_a$. Say $K_b \in \mathcal{K} \setminus \{K_a\}$ is the clone set containing b. Since b pairwise defeats a in $\boldsymbol{\sigma}$, K_b pairwise defats K_a in $\boldsymbol{\sigma}^{\mathcal{K}}$ by the clone set definition. Take any $K \in \mathcal{K}$ that pairwise defeats K_b

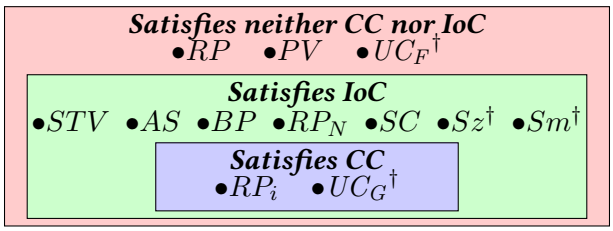

Figure 6: Behavior of SCFs from Tables 1 and 2 w.r.t IoC/CC. † indicates majoritarian SCFs.

in $\boldsymbol{\sigma}^{\mathcal{K}}$. This implies there exists some $c \in K$ that pairwise defeats b in $\boldsymbol{\sigma}$. Since b left-covers a in $\boldsymbol{\sigma}$, this implies c pairwise defeats a in $\boldsymbol{\sigma}$ and thus K pairwise defeats K_a in $\boldsymbol{\sigma}^{\mathcal{K}}$. Hence, K_b left-covers and pairwise defeats K_a in $\boldsymbol{\sigma}^{\mathcal{K}}$, implying $K_a \notin UC_G(\boldsymbol{\sigma}^{\mathcal{K}})$.

This implies we either have $a \notin UC_G(\boldsymbol{\sigma}|_{K_a})$ or $K_a \notin UC_G(\boldsymbol{\sigma}^{\mathcal{K}})$. By Definition 5, this implies $a \notin \Pi_{UC_G}(\boldsymbol{\sigma}, \mathcal{K})$.

($\Leftarrow$) : Say $a \notin \Pi_{UC_G}(\boldsymbol{\sigma}, \mathcal{K})$ and $K_a \in \mathcal{K}$ is the clone set that contains a. This implies at least one of the two following two cases must be true:

1. $a \notin UC_G(\boldsymbol{\sigma}|_{K_a})$. Then there exists $b \in K_a$ that left-covers and pairwise defeats a in $\boldsymbol{\sigma}|_{K_a}$. Since pairwise defeats are not affected by the addition of other candidates, b also pairwise defeats a in $\boldsymbol{\sigma}$. Take any $c \in A$ that pairwise defeats b. If $c \in K_a$, then c must pairwise defeat a because b left covers $\boldsymbol{\sigma}|_{K_a}$. If $c \notin K_a$, then c must pairwise defeat a by the clone set definition, since $a, b \in K_a$. Thus, b left-covers and pairwise defats a in $\boldsymbol{\sigma}$, implying $a \notin UC_G(\boldsymbol{\sigma})$.

2. $K_a \notin UC_G(\boldsymbol{\sigma}^{\mathcal{K}})$. Then there exists $K \in \mathcal{K}$ that left-covers and pairwise defeats K_a in $\boldsymbol{\sigma}^{\mathcal{K}}$. Since K_a cannot pairwise defeat itself, this implies $K \neq K_a$. Take any $b \in K$. Since K pairwise defeat K_a in $\boldsymbol{\sigma}^{\mathcal{K}}$, this implies b pairwise defeats a in $\boldsymbol{\sigma}$. Take any $c \in A$ that pairwise defeats b in $\boldsymbol{\sigma}$. We cannot have $c \in K_a$ since K pairwise defeats K_a. If $c \in K$, then c must pairwise defeat a since K pairwise defeats K_a. If $c \notin K$, say $K_c \in \mathcal{K} \setminus \{K_a, K\}$ is the clone set that contains c. Since c pairwise defeats b in $\boldsymbol{\sigma}$, K_c pairwise defeast K in $\boldsymbol{\sigma}^{\mathcal{K}}$. Since K left-covers K_a in $\boldsymbol{\sigma}^{\mathcal{K}}$, this implies K_c pairwise defeats K_a, and therefore c pairwise defeats a in $\boldsymbol{\sigma}$. Hence, b left-covers and pairwise defeats a, implying $a \notin UC_G(\boldsymbol{\sigma})$.

Hence, $a \notin \Pi_{UC_G}(\boldsymbol{\sigma}, \mathcal{K})$ implies $a \notin UC_G(\boldsymbol{\sigma})$, completing the proof. □

The disparity between UC_F and UC_G motivates future work in investigating whether other TSs known be CC can be extended into SCFs while maintaining CC (*cf.* Brandt et al. [7] for a *conservative extension* of any TS that in fact preserves CC). Existing negative results for TSs, on the other hand, readily generalize to any of their extensions. This is because Laffond et al. [36] show that for any tournament and a decomposition $\mathcal{K}$ into its (tournament) components, there exists some preference profile (that induces this tournament) for which $\mathcal{K}$ is once again a valid decomposition (their Prop. 1); this can be used to show that the CC definitions for TSs and their SCF interpretations coincide under the odd $|N|$ assumption [36, Prop. 2]. Since Sm and Sz (Table 2) are both SCF extensions of the TS *top cycle*, which is not CC, we get:

Proposition 32 (Consequence of Laffond et al. [36, Props. 2, 5])**.** *Sm and Sz are not CC.*

For a summary of our results from Section 3 and Appendix B, see Figure 6.

C Social Preference Functions

We now turn to *social preference functions (SPFs)*, which, given input $\boldsymbol{\sigma}$, return a set of rankings of A, rather than a subset of candidates. Indeed, SPFs may be more useful than SCFs in certain settings, such as the meta-search engine example in the previous section. We will first present the definition of IoC for SPFs as introduced by Freeman et al. [26].

For a ranking r over A and a non-empty $K \subseteq A$, let $r\neg_{K\to z}$ be the ranking obtained from r by replacing the highest-ranked element of K with a new candidate z and removing all other candidates in K. For example, if $r = (a \succ b \succ c \succ d)$ and $K = \{b, d\}$, then $r\neg_{\{b,d\}\to z} = (a \succ z \succ c)$. For a set of rankings R, let $R\neg_{K\to z} = \{r\neg_{K\to z} : r \in R\}$.

Definition 33. [Freeman et al. 26, Def. 4] An SPF F is *independent of clones (IoC)* if for all $\boldsymbol{\sigma}$, each non-trivial clone set K, and $a \in K$, we have $F(\boldsymbol{\sigma})\neg_{K\to z} = F(\boldsymbol{\sigma} \setminus \{a\})\neg_{(K\setminus\{a\})\to z}$.

Much like its SCF precursor, the IoC criterion for SPFs focuses on the performance of *some* clone in K, which is not necessarily the clone that would have ranked the highest if the same SPF was applied to members of K alone. Once again, we would like to strengten this property.

To the best of our knowledge, CC has not been studied for SPFs in prior work. Hence, we now introduce a natural extension of Definition 7. Given rankings r and r' over different sets, where a appears in r, let $r(a \to r')$ be r with a replaced by r', in order. For example, if $r = (a \succ b \succ c)$ and $r' = (d \succ e)$, then $r(b \to r') = a \succ d \succ e \succ c$. For sets of rankings R, R', and R'', we write $R(a \to R') = \{r(a \to r') : r \in R, r' \in R'\}$ and $R(a \to R', b \to R'') = R(a \to R')(b \to R'')$.

Definition 34. [CC for SPFs] A neutral SPF F is *composition-consistent (CC)* if for all $\boldsymbol{\sigma}$ and all clone decompositions $\mathcal{K}$, we have $F(\boldsymbol{\sigma}) = F(\boldsymbol{\sigma}^{\mathcal{K}})(K \to F(\boldsymbol{\sigma}|_K)$ for $K \in \mathcal{K})$.

If F is CC, clone sets must appear as intervals in $F(\boldsymbol{\sigma})$ in the order(s) specified by $F(\boldsymbol{\sigma}^{\mathcal{K}})$, and the order(s) within each clone set K is specified by $F(\boldsymbol{\sigma}|_K)$. Next, we show that the definition of IoC for SPFs by Freeman et al. [26] and our novel definition of CC for SPFs are consistent with the ones for SCFs.

Proposition 35. *Let f be the SCF that corresponds to SPF F,* i.e., *$f(\boldsymbol{\sigma}) = \{top(r) : r \in F(\boldsymbol{\sigma})\}$. If F is IoC, then f is IoC. If F is CC, then f is CC.*

(All proofs of claims in this section are given in subsections further below).

It is straightforward to see that the reverse of Proposition 35 is false: given an SCF f that is CC/IoC, we can always construct an SPF that picks the top ranked candidate according to f, and then orders the remaining candidates according to some arbitrary order (*e.g.*, by their plurality scores). Intuitively, such an SPF cannot be expected to obey *any* reasonable definition of IoC/CC for SPFs. Further, there are also more "natural" counterexamples, as we show in Appendix C.3.

Further, we show that the hierarchy between CC and IoC (Proposition 8) extends to SPFs.

Proposition 36. *If a given SPF is CC, then it is also IoC.*

We can interpret each version of RP as an SPF outputting the topological sorting(s) of the final graph(s); Schulze [56] shows that BP, too, admits an interpretation as an SPF. Finally, we can view STV as an SPF outputting candidates in reverse order of elimination; see Appendix C.1 for formal definitions. Our results generalize to each of these SPFs.

Theorem 5. *Each of $\{STV, BP, RP, RP_i, RP_N\}$ satisfies IoC/CC (for all $\boldsymbol{\sigma}$) if and only if its SPF version does.*

While the above results are intuitive, they rely on a careful definition of CC/IoC for SPFs. For example, Boehmer et al. [1, Appendix A] provide an alternative definition of IoC for SPFs where the bottom-ranked clone is replaced in Definition 33 rather than the top-ranked one. They show that under this alternative definition, an SPF that iteratively adds the veto winner (see our Footnote 1) to the ranking and deletes it from the profile would be IoC (with bottom replacement), whereas STV would not. Hence, both our Prop. 35 and Thm. 5 would fail under this alternative IoC definition.

Theorem 5 gives us a single CC SPF: RP_i, which (like its SCF counterpart), fails anonymity. We next give a (to us, surprising) negative result that this weakness is inevitable.

Theorem 6. *No anonymous SPF can be CC.*

Theorem 6 has strong implications. First, it shows that to design CC SPFs, a non-anonymous tie-breaker (such as the one by Zavist and Tideman [65] for RP_i) is not only sufficient but necessary. Indeed, in Appendix C.3, we design a novel CC SPF that uses a similar tie-breaking rule, inspired by a nested version of STV introduced by Freeman et al. [26]. Second, it shows that in settings where anonymity is a must and ties are likely to occur, CC is too demanding for SPFs, and motivates studying its relaxations. For example, as mentioned in Section 4.1, the Kemeny SPF (which returns the ranking(s) with the minimum total Kendall-Tau distance to voters' rankings) obeys a weaker form of CC, which requires $F(\boldsymbol{\sigma})$ and $F(\boldsymbol{\sigma}^{\mathcal{K}})(K \to F(\boldsymbol{\sigma}|_K)$ for $K \in \mathcal{K})$ to have a nonempty intersection under certain decompositions (where the summary is single-peaked or single-crossing), rather than being equal for all decompositions [16]. Since Kemeny is not IoC as an SCF [62], this relaxation is incomparable with IoC for SPFs by our Proposition 35.

We also observe that the impossibility in Theorem 6 can be circumvented if $|N|$ is assumed to be odd. In this regime, for a neutral SPF F, we can define F^{CC} analogously to Definition 12 to obtain CC (by forcing Q-nodes to output their majority ranking). We leave formalizing this transformation and identifying which SPF-specific axioms (*e.g.*, independence of the last-ranked alternatives) are preserved by F^{CC} for future work.

C.1 Definitions of social preference functions

Here, we give the descriptions of the SPF versions of several SCFs we have discussed in previous sections. Below, the "RP procedure" refers to the process of locking in edges from M in non-increasing order, skipping the ones that create a tie.

- RP: Return all rankings r that correspond to the topological ordering of the final graph constructed by the RP procedure for *some* tie-breaking order.
- RP_i: Return the topological ordering of the final graph constructed by the RP procedure using Σ_i as a tie-breaker.
- RP_N: Return the union over RP_i for all $i \in N$
- STV: At each round, eliminate the candidate with the least plurality votes, until only one candidate remains. Output the reverse order elimination (*i.e.*, the candidate eliminated first is ranked last).
- BP: Construct the strength matrix S as described in Table 1. Define relationship $\succ_{BP}$ over candidates as $a \succ_{BP} b$ if $S[a,b] > S[b,a]$ and $a =_{BP} b$ if $S[a,b] = S[b,a]$. As proven by Schulze [56], $\succeq_{BP}$ satisfies transitivity; hence, it gives a weak ordering over candidates A. Return all strict rankings r that are consistent with the weak ordering of $\succeq_{BP}$. For example, for $\boldsymbol{\sigma}'$ with the strength matrix given in the extended proof of Theorem 1 below in Appendix D.2, we have $a_1 =_{BP} a_2 \succ_{BP} b \succ_{BP} c$. This implies $BP^*(\boldsymbol{\sigma}') = \{r_1, r_2\}$ where BP^* is the SPF version of BP, with $r_1 : a_1 \succ a_2 \succ b \succ c$ and $r_2 : a_2 \succ a_1 \succ b \succ c$.

C.2 Proof of Proposition 35

We first prove that our novel definitions of IoC/CC for SPFs are consistent with the ones for SCFs.

Proposition 35. *Let f be the SCF that corresponds to SPF F,* i.e., *$f(\boldsymbol{\sigma}) = \{top(r) : r \in F(\boldsymbol{\sigma})\}$. If F is IoC, then f is IoC. If F is CC, then f is CC.*

Proof. Say F satisfies IoC, and pick any profile $\boldsymbol{\sigma}$, non-trivial clone set K with respect to $\boldsymbol{\sigma}$, and any clone $a \in K$. Note that for any ranking r over A and any candidate $b \in A \setminus K$, we have $b = top(r) \iff b = top(r\neg_{K\to z})$, since relabeling/removing lower ranked candidates does not change the fact that b is ranked top. Hence, we have

$$\begin{aligned} b \in f(\boldsymbol{\sigma}) &\iff \exists r \in F(\boldsymbol{\sigma}) \text{ s.t. } b = top(r) \iff \exists r \in F(\boldsymbol{\sigma})\neg_{K\to z} \text{ s.t. } b = top(r) \\ &\overset{(IoC)}{\iff} \exists r \in F(\boldsymbol{\sigma} \setminus \{a\})\neg_{(K\setminus\{a\})\to z} \text{ s.t. } b = top(r) \\ &\iff \exists r \in F(\boldsymbol{\sigma} \setminus \{a\}) \text{ s.t. } b = top(r) \\ &\iff b \in f(\boldsymbol{\sigma} \setminus \{a\}), \end{aligned}$$

which give proves f satisfies condition (2) from Definition 2. Next, notice that by definition of the $\neg$ operator, for any ranking r we have $top(r) \in K \iff z = top(r\neg_{K\to z})$. This implies

$$K \cap f(\boldsymbol{\sigma}) \neq \emptyset \iff \exists r \in F(\boldsymbol{\sigma}) \text{ s.t. } top(r) \in K \iff \exists r \in F(\boldsymbol{\sigma})\neg_{K\to z} \text{ s.t. } z = top(r) \tag{1}$$

$$\overset{(IoC)}{\iff} \exists r \in F(\boldsymbol{\sigma} \setminus \{a\})_{(K\setminus\{a\})\to z} \text{ s.t. } z = top(r) \tag{2}$$

$$\iff \exists r \in F(\boldsymbol{\sigma} \setminus \{a\}) \text{ s.t. } top(r) \in K \setminus \{a\} \tag{3}$$

$$\iff (K \setminus \{a\}) \cap f(\boldsymbol{\sigma} \setminus \{a\}) \neq \emptyset, \tag{4}$$

which proves f satisfies condition (1) from Definition 2. Hence, f is IoC.

Next, assume F satisfies CC. Take any profile $\boldsymbol{\sigma}$ and clone decomposition $\mathcal{K}$ with respect to $\boldsymbol{\sigma}$. By Definition 34, we have

$$\begin{aligned} F(\boldsymbol{\sigma}) = F(\boldsymbol{\sigma}^{\mathcal{K}})(K \to F(\boldsymbol{\sigma}|_K) \text{ for } K \in \mathcal{K}) \Rightarrow f(\boldsymbol{\sigma}) = top(F(\boldsymbol{\sigma})) &= \bigcup_{K \in top(F(\boldsymbol{\sigma}^{\mathcal{K}}))} top(F(\boldsymbol{\sigma}|_K)) \\ &= \bigcup_{K \in f(\boldsymbol{\sigma}^{\mathcal{K}})} f(\boldsymbol{\sigma}|_K) \end{aligned}$$

Hence, we have $f(\boldsymbol{\sigma}) = \Pi_f(\boldsymbol{\sigma}, \mathcal{K})$, so f satisfies CC. □

C.3 (Nested) nested runoff voting

In this section, we briefly discuss how non-anonymous tie-breakers (such as the one introduced by Zavist and Tideman [65]) can be used to construct CC SPFs other than RP_i. Freeman et al. [26] introduce an SPF named Nested Runoff (NR), which is a modification of STV: at each round, instead of the candidate with the lowest plurality score, the winner of $STV(\text{rev}(\boldsymbol{\sigma}))$ is eliminated, where $\text{rev}(\boldsymbol{\sigma})$ is $\boldsymbol{\sigma}$ with every voter's ranking reversed. Freeman et al. show that NR is IoC as an SPF. By our Proposition 35, this implies NR is IoC as an SCF too. However, since it is anonymous, it cannot be CC as an SPF by Theorem 6. In fact, NR is not CC as an SCF either; to see this, consider the following profile over 4 candidates with 3 voters:

$$\boldsymbol{\sigma} = \begin{cases} b \succ_1 a_2 \succ_1 a_1 \succ_1 c \\ a_2 \succ_2 a_1 \succ_2 c \succ_2 b \\ c \succ_3 b \succ_3 a_1 \succ_3 a_2 \end{cases} \tag{5}$$

Say $\mathcal{K} = \{\{a_1, a_2\}, \{b\}, \{c\}\}$. It can be checked that $a_1 \in NR(\boldsymbol{\sigma})$ but $a_1 \notin \Pi_{NR}(\boldsymbol{\sigma}, \mathcal{K})$ since $a_1 \notin NR(\boldsymbol{\sigma}|_{\{a_1,a_2\}})$, which violates CC.

Now, say STV_i is simply the version of STV that uses voter i's vote as a tie-breaker (*i.e.*, if multiple candidates tie for the lowest plurality score at any point, the one ranked lowest by voter i is eliminated). It is straightforward to check that, much like STV, STV_i is IoC as an SPF and as an SCF, but CC as neither. Unlike STV, however, STV_i is decisive on all $\boldsymbol{\sigma}$. Now, we define NR_i using STV_i on the reverse profile to decide on the order of elimination. We will show that NR_i is CC as an SCF. Take any profile $\boldsymbol{\sigma}$ and any decomposition $\mathcal{K}$ with respect to $\boldsymbol{\sigma}$. Since we are using a decisive tie-breaker, we will have $|NR_i(\boldsymbol{\sigma})| = |\Pi_{NR_i}(\boldsymbol{\sigma}, \mathcal{K})| = 1$, so it is sufficient to show containment in one direction. Say $\Pi_{NR_i}(\boldsymbol{\sigma}, \mathcal{K}) = \{a\}$ and $K_a \in \mathcal{K}$ is the clone set containing a. This implies $NR_i(\boldsymbol{\sigma}^{\mathcal{K}}) = \{K_a\}$ and $NR_i(\boldsymbol{\sigma}|_{K_a}) = \{a\}$. Say $K_1, K_2, \ldots, K_\ell$ is the order in which clone sets are eliminated when NR_i is run on $\boldsymbol{\sigma}^{\mathcal{K}}$. This implies $STV_i(\text{rev}(\boldsymbol{\sigma}^{\mathcal{K}})) = \{K_1\}$. By successive application of the IoC for SCF property, we must have $STV_i(\text{rev}(\boldsymbol{\sigma})) = \{b_1\}$ for some $b_1 \in K_1$, implying that the first candidate eliminated by NR_i on input $\boldsymbol{\sigma}$ belongs to K_1. If $|K_1| > 1$, this argument can be repeated again with $STV_i(\text{rev}(\boldsymbol{\sigma} \setminus \{b_1\}))$, implying the next eliminated candidate too will belong to K_1. Applying this argument repeatedly gives us that NR_i on input $\boldsymbol{\sigma}$ will eliminate all elements of K_1 before any other candidate. Now, by the assumption on the order of elimination in $NR_i(\boldsymbol{\sigma}^{\mathcal{K}})$ we have $STV_i(\text{rev}(\boldsymbol{\sigma}^{\mathcal{K}} \setminus K_1)) = \{K_2\}$, and the same IoC argument can be applied to show that $STV_i(\text{rev}(\boldsymbol{\sigma} \setminus K_1)) = \{b_2\}$ for some $b_2 \in K_2$. Inductively applying this argument gives that NR_i will eliminate all candidates of K_i before any candidate of K_{i+1} for all $i \in [\ell]$, where $K_{\ell+1} = K_a$ (since it never gets eliminated). Hence, at some point in the execution of NR_i on $\boldsymbol{\sigma}$, we will have $\boldsymbol{\sigma} \setminus \left(\bigcup_{i \in [\ell]} K_i\right)$ left. However, this is precisely $\boldsymbol{\sigma}|_{K_a}$, and by assumption we have $NR_i(\boldsymbol{\sigma}|_{K_a}) = \{a\}$, showing that we must indeed have $NR_i(\boldsymbol{\sigma}) = \{a\}$, proving that NR_i is CC as an SCF.

To see that NR_i is not CC as an SPF, once again consider the profile from (5) and NR_3^* (*i.e.*, the SPF version of NP_i using $i = 3$ as the tie-breaker). It can be checked that $NR_3^*(\boldsymbol{\sigma}) = c \succ b \succ a_1 \succ a_2$, but $NR_3^*(\boldsymbol{\sigma}|_{\{a_1,a_2\}}) = a_2 \succ a_1$, hence violating CC as an SPF. Thus, NR_i serves as a "natural" counterexample showing that the reverse of Proposition 35 does not always hold.

Finally, let us use NR_i to design a CC SPF. Define the *nested nested runoff rule* using voter i as a tie-breaker (NNR_i) as a modification of NR_i that, instead of STV_i, runs NR_i on the reverse profile to decide the next eliminated candidate. Given any profile $\boldsymbol{\sigma}$ and decomposition $\mathcal{K}$, say $NNR_i(\boldsymbol{\sigma}^{\mathcal{K}}) = K_\ell \succ K_{\ell-1} \succ \ldots \succ K_2 \succ K_1$. This implies $NR_i(\text{rev}(\boldsymbol{\sigma}^{\mathcal{K}})) = \{K_1\}$. Since NR_i is CC as an SCF, it is IoC as an SCF (by Proposition 8). Hence, we must have $NR_i(\text{rev}(\boldsymbol{\sigma})) = \{b_1\}$ for some $b_1 \in K_1$. By the CC property of NR_i, this implies $NR_i(\text{rev}(\boldsymbol{\sigma})|_{K_1}) = \{b_1\}$, implying that the candidate ranked at the bottom of $NNR_i^*(\boldsymbol{\sigma}|_{K_1})$ is b_1, the same as the candidate ranked at the bottom of $NNR_i^*(\boldsymbol{\sigma})$. If $|K_1| > 1$, applying the same argument again gives us $NR_i(\text{rev}(\boldsymbol{\sigma} \setminus \{b_1\})) = NR_i(\text{rev}(\boldsymbol{\sigma}|_{K_1} \setminus \{b_1\}))$. Thus, all the candidates in K_1 appear in the bottom of $NNR_i(\boldsymbol{\sigma}|_{K_1})$, and they appear exactly in the order they do in $NNR_i(\boldsymbol{\sigma})$. Applying this argument inductively to all K_i for $i \in [\ell]$ gives us exactly the CC definition for SPFs (Definition 34), completing the proof.

Hence, we have arrived at an interesting hierarchy. STV_i is IoC as an SPF and an SCF, but CC as neither. NR_i, which uses STV_i to eliminate candidates, is CC as an SCF, but still only IoC as an SPF. Lastly, NNR_i, which uses NR_i to eliminate candidates, is CC both as an SPF and an SCF. Based on this observation, we believe studying the axiomatic properties of this type of (nested) nested rules is an interesting future direction.

C.4 Proof of Proposition 36

We first prove that the CC to IoC relationship extends to the definitions for SCFs we have introduced.

Proposition 36. *If a given SPF is CC, then it is also IoC.*

Proof. Say F satisfies CC. By Definition 34, this implies that F is neutral. Pick any profile $\boldsymbol{\sigma}$, non-trivial clone set K with respect to $\boldsymbol{\sigma}$, a $a \in K$. Consider the clone decomposition $\mathcal{K} = \{K\} \cup \{\{b\}\}_{b \in A \setminus K}$ for $\boldsymbol{\sigma}$ and the clone decomposition $\mathcal{K}' = \{K \setminus \{a\}\} \cup \{\{b\}\}_{b \in A \setminus K}$ for $\boldsymbol{\sigma} \setminus \{a\}$ (i.e., the decomposition which groups all existing members of K together, and everyone else is a singleton). Notice that $\boldsymbol{\sigma}^{\mathcal{K}}$ and $(\boldsymbol{\sigma} \setminus \{a\})^{\mathcal{K}'}$ are identical except the meta-candidate for K in the former is replaced with the meta-candidate for $K \setminus \{a\}$ in the latter. By neutrality, this implies: $F(\boldsymbol{\sigma}^{\mathcal{K}})\neg_{\{K\} \to K \setminus \{a\}} = F((\boldsymbol{\sigma} \setminus \{a\})^{\mathcal{K}'})$. Each $K' \in \mathcal{K} \setminus \{K\}$ is a singleton, and hence $F(\boldsymbol{\sigma}|_{K'})$ is just a single ranking with the only element in K'. For any $b \in A$, say r_b is the trivial ranking over $\{b\}$. Using CC, we get

$$\begin{aligned}
F(\boldsymbol{\sigma})\neg_{K \to z} &= \left(F(\boldsymbol{\sigma}^{\mathcal{K}})(K' \to F(\boldsymbol{\sigma}|_{K'}) \text{ for } K' \in \mathcal{K})\right) \neg_{K \to z} \\
&= \left(F(\boldsymbol{\sigma}^{\mathcal{K}})(K \to F(\boldsymbol{\sigma}|_K); \{b\} \to r_b \text{ for } b \in A \setminus K)\right) \neg_{K \to z} \\
&= F(\boldsymbol{\sigma}^{\mathcal{K}})(K \to r_z; \{b\} \to r_b \text{ for } b \in A \setminus K) \\
&= F((\boldsymbol{\sigma} \setminus \{a\})^{\mathcal{K}'})((K \setminus \{a\}) \to r_z; \{b\} \to r_b \text{ for } b \in A \setminus K) \\
&= \left(F((\boldsymbol{\sigma} \setminus \{a\})^{\mathcal{K}'})((K \setminus \{a\}) \to F(\boldsymbol{\sigma}|_{K \setminus \{a\}}); \{b\} \to r_b \text{ for } b \in A \setminus K)\right) \neg_{(K \setminus \{a\}) \to z} \\
&= F(\boldsymbol{\sigma} \setminus \{a\})\neg_{(K \setminus \{a\}) \to z}.
\end{aligned}$$

Hence, F satisfies IoC. □

C.5 Proof of Theorem 5

We now prove that for SCFs for which we described the SPF version above, our results generalize.

Theorem 5. *Each of $\{STV, BP, RP, RP_i, RP_N\}$ satisfies IoC/CC (for all $\boldsymbol{\sigma}$) if and only if its SPF version does.*

Proof. We prove the axioms satisfied by each SPF as a seperate Lemma.

Lemma 37. *The SPF version of STV is IoC but not CC.*

Proof. The fact that SPF version of STV is IoC is shown by Freeman et al. [26]. Since SCF version of STV is not CC (Theorem 1), then the SPF version of STV is also not CC by Proposition 35. □

Lemma 38. *The SPF version of BP is IoC but not CC.*

Proof. Take any profile $\boldsymbol{\sigma}$, clone set K, and $a \in K$. Say S and S' (resp. M and M') are the strength (resp. majority) matrices that result from running the BP procedure on $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma} \setminus \{a\}$, respectively. First, notice that for any $b, c \in A \setminus \{a\}$, we have $M[b, c] = M'[b, c]$, since the removal of candidate does not change the pairwise relationship between the remaining candidates. Take any $x \in A \setminus \{a\}$ and $y, z \in A \setminus K$. We would like to show that:

$$S[x, y] = S'[x, y] \quad S[y, x] = S'[y, x] \quad S[y, z] = S'[y, z] \tag{6}$$

Since M' is simply M with a removed, any path in M' exists in M. This gives you the $\geq$ direction of all of the equalities in (6). For the reverse direction, consider any path P from x to y in M. If the path does not contain a, then it exists in M'. If it does contain a, consider the alternative path P' that starts from the last element belonging to K in P, but replaces it with x (so P' is also a path from x to y). By the clone definition, the first edge in the path is equally strong, and the remaining edges are the same. Since the strength of a path is the minimum weight over the edges in the path, this shows that P' is at least as strong as P. The same method can be applied for paths from y to x by replacing the first occurrence of a member of K with x. Now take any path P in M from y to z. Again, if it does not contain a, it still exists in M'. If it does contain it, then pick any $b \in K \setminus \{a\}$ (exists sicne

K is non-trivial) and construct path P' by replacing the interval in P from the first occurrence of a member of K to the last occurrence of a member of K with b. By the clone definition, the incoming and outgoing edge of b will have the same weight as the incoming and outgoing edge to this interval. Since the remaining paths are only subtracted, again P' is at least as strong as P. This finishes the $\leq$ direction of all of the equalities in (6).

This implies that for any $b, c \in A \setminus \{a\}$ such that at least one of them is not in K, we will have $b \succeq_{BP} c \iff b \succeq'_{BP} c$, where $\succeq_{BP}$ and $\succeq'_{BP}$ are the (weak) linear orderings resulting from running BP on $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma} \setminus \{a\}$, respectively. This implies that $BP^*(\boldsymbol{\sigma})\neg_{K\to z} = BP^*(\boldsymbol{\sigma} \setminus \{a\})\neg_{(K\setminus\{a\})\to z}$, proving that BP^* (the SPF version of BP) is IoC.

Since SCF version of BP is not CC (Theorem 1), then BP^* is also not CC by Proposition 35. □

Lemma 39. *The SPF version of RP neither IoC nor CC.*

Proof. Since the SCF version of RP is not IoC [65] and therefore not CC (by Proposition 8), SPF version of RP is neither IoC nor CC Proposition 35. □

Lemma 40. *The SPF version of RP_i is both IoC and CC.*

Proof. The proof that RP_i satisfies CC follows easily from the proof of Theorem 2. There, (using Lemma 43) we showed that given any decomposition $\mathcal{K}$ the RP ranking resulting from running RP_i on $\boldsymbol{\sigma}$ has each clone set in $\mathcal{K}$ as an interval, in the order specified by the RP ranking resulting from running RP_i on $\boldsymbol{\sigma}^{\mathcal{K}}$. Moreover, we showed that the clone set ranked first in this ranking (say K_1) appeared in the order specified by the RP ranking resulting from running RP_i on $\boldsymbol{\sigma}|_{K_1}$. This last proof did not use the fact that K_1 was the first clone set to appear in the ranking, but only that it appeared as an interval. Hence, the same proof can be easily applied to all $K \in \mathcal{K}$, since each appear as an interval. As a result, we have $RP_i^*(\boldsymbol{\sigma}) = RP_i^*(\boldsymbol{\sigma}^{\mathcal{K}})(K \to RP_i^*(\boldsymbol{\sigma})$ for $K \in \mathcal{K})$, where RP_i^* is the SPF version of RP_i. □

Lemma 41. *The SPF version of RP_N is IoC but not CC.*

Proof. Say RP_N^* and RP_i^* are the SPF versions of RP_N and RP_i, respectively. Fix any profile $\boldsymbol{\sigma}$, non-trivial clone set K, and $a \in K$. Since RP_i^* is IoC for each $i \in N$ by Lemma 40 and by definition of the $\neg$ operator, we have

$$\begin{aligned} RP_N^*(\boldsymbol{\sigma})\neg_{K\to z} &= \left(\bigcup_{i\in N} RP_i^*(\boldsymbol{\sigma})\right)\neg_{K\to z} = \bigcup_{i\in N} RP_i^*(\boldsymbol{\sigma})\neg_{K\to z} = \bigcup_{i\in N} RP_i^*(\boldsymbol{\sigma})\neg_{K\to z} \\ &= \bigcup_{i\in N} RP_i^*(\boldsymbol{\sigma} \setminus \{a\})\neg_{(K\setminus\{a\})\to z} = \left(\bigcup_{i\in N} RP_i^*(\boldsymbol{\sigma} \setminus \{a\})\right)\neg_{(K\setminus\{a\})\to z} \\ &= RP_N^*(\boldsymbol{\sigma} \setminus \{a\})\neg_{(K\setminus\{a\})\to z}, \end{aligned}$$

proving RP_N^* satisfies IoC. Since the SCF version of RP_N does not satisfy CC (Proposition 9), this proves that RP_N^* is not CC by Proposition 35. □

Lemmata 37 to 41, together with our result from Theorems 1 and 2, prove the theorem statement. □

C.6 Proof of Theorem 6

We next prove a surprising negative resulting showing the incompatibility of anonymity and composition consistency for SPFs.

Theorem 6. *No anonymous SPF can be CC.*

Proof. Assume for the sake of contradiction that we have an SPF F that is CC and anonymous. By Definition 34, this implies F is also neutral. Consider a profile $\boldsymbol{\sigma}$ over $A = \{a, b, c\}$ with two votes. Voter 1 ranks $a \succ_1 b \succ_1 c$ and Voter 2 ranks $c \succ_2 b \succ_2 a$. Define $K_1 = \{a, b\}$ and $K_2 = \{b, c\}$, which are both clone sets with respect to $\boldsymbol{\sigma}$. Further, define $\mathcal{K}_1 = \{K_1, \{c\}\}$ and $\mathcal{K}_2 = \{\{a\}, K_2\}$, which are both clone decompositions with respect to $\boldsymbol{\sigma}$. Consider $\boldsymbol{\sigma}^{\mathcal{K}_1}$, which consists of $K_1 \succ_1 \{c\}$ and $\{c\} \succ_2 K_1$. Since F is neutral and anonymous, we must have $F(\boldsymbol{\sigma}^{\mathcal{K}_1}) = \{K_1 \succ \{c\}, \{c\} \succ K_1\}$, otherwise permuting Voter 1 with Voter 2 and K_1 with $\{c\}$ gives a contradiction. By the same reasoning, we must have $F(\boldsymbol{\sigma}|_{K_1}) = \{a \succ b, b \succ a\}$. By composition consistency (Definition 34), we must have

$$F(\boldsymbol{\sigma}) = F(\boldsymbol{\sigma}^{\mathcal{K}_1})(K \to F(\boldsymbol{\sigma}|_K) \text{ for } K \in \mathcal{K}_1) = \left\{ \begin{array}{l} a \succ b \succ c, \\ b \succ a \succ c, \\ c \succ a \succ b, \\ c \succ b \succ a \end{array} \right\}. \tag{7}$$

Similarly, $\boldsymbol{\sigma}^{\mathcal{K}_2}$, which consists of $\{a\} \succ_1 K_2$ and $K_2 \succ_2 \{a\}$. By neutrality and anonymity, we must have $F(\boldsymbol{\sigma}^{\mathcal{K}_2}) = \{\{a\} \succ_1 K_2, K_2 \succ_2 \{a\}\}$, otherwise permuting Voter 1 with 2 and $\{a\}$ with K_2 gives a contradiction. By the same reasoning, we must have $F(\boldsymbol{\sigma}|_{K_2}) = \{b \succ c, c \succ b\}$. By CC, we must have

$$F(\boldsymbol{\sigma}) = F(\boldsymbol{\sigma}^{\mathcal{K}_1})(K \to F(\boldsymbol{\sigma}|_K) \text{ for } K \in \mathcal{K}_1) = \left\{ \begin{array}{l} a \succ b \succ c, \\ a \succ b \succ c, \\ b \succ c \succ a, \\ c \succ b \succ a \end{array} \right\}. \tag{8}$$

Comparing (7) with (8), we immediately get a contradiction. □

D On Section 3 (Analysis of IoC Social Choice Functions)

In this section, we provide the proofs omitted from Section 3 of the main body.

D.1 Proof of Proposition 8

We first prove the relationship between CC and IoC:

Proposition 8. *If a given SCF is composition-consistent, then it is also independent of clones.*

Proof. For a CC rule f, take any profile $\boldsymbol{\sigma}$ over candidates A, non-trivial clone set $K \subset A$, and candidate $a \in A$. Consider the clone decomposition $\mathcal{K} = \{K\} \cup \{\{b\}\}_{b \in A \setminus K}$ for $\boldsymbol{\sigma}$ and the clone decomposition $\mathcal{K}' = \{K \setminus \{a\}\} \cup \{\{b\}\}_{b \in A \setminus K}$ for $\boldsymbol{\sigma} \setminus \{a\}$ (i.e., the decomposition which groups all existing members of K together, and everyone else is a singleton). Notice that $\boldsymbol{\sigma}^{\mathcal{K}}$ and $(\boldsymbol{\sigma} \setminus \{a\})^{\mathcal{K}'}$ are identical except the meta-candidate for K in the former is replaced with the meta-candidate for $K \setminus \{a\}$ in the latter. Since f is neutral by Definition 7, this implies:

$$K \in f\left(\boldsymbol{\sigma}^{\mathcal{K}}\right) \iff K \setminus \{a\} \in f\left((\boldsymbol{\sigma} \setminus \{a\})^{\mathcal{K}'}\right) \tag{9}$$

$$\forall b \in A \setminus C: \quad \{b\} \in f\left(\boldsymbol{\sigma}^{\mathcal{K}}\right) \iff \{b\} \in f\left((\boldsymbol{\sigma} \setminus \{a\})^{\mathcal{K}'}\right) \tag{10}$$

By Definition 5, it is easy to see that for any $K \in \mathcal{K}$, we have $K \cap \Pi_f(\boldsymbol{\sigma}, \mathcal{K}) \neq \emptyset \iff K \in f\left(\boldsymbol{\sigma}^{\mathcal{K}}\right)$. Based on this, (9) and (10) respectively imply:

$$K \cap \Pi_f(\boldsymbol{\sigma}, \mathcal{K}) \neq \emptyset \iff K \setminus \{a\} \cap \Pi_f(\boldsymbol{\sigma} \setminus \{a\}, \mathcal{K}') \neq \emptyset \tag{11}$$

$$\forall b \in A \setminus K: \quad b \in \Pi_f(\boldsymbol{\sigma}, \mathcal{K}) \iff b \in \Pi_f(\boldsymbol{\sigma} \setminus \{a\}, \mathcal{K}') \tag{12}$$

Since f is CC, we have $\Pi_f(\boldsymbol{\sigma}, \mathcal{K}) = f(\boldsymbol{\sigma})$ and $\Pi_f(\boldsymbol{\sigma} \setminus \{a\}, \mathcal{K}') = f(\boldsymbol{\sigma} \setminus \{a\})$ so (11) and (12) respectively imply conditions 1 and 2 in Definition 2, proving that f is IoC. □

D.2 (Extended) Proof of Theorem 1

The proof of Theorem 1 is given in the main body of the paper, but the winners under each SCF are stated without detailed calculations. Here, we give a more extensive proof that walks through the implementation of each SCF.

Theorem 1. *STV, BP, AS, and SC all fail composition consistency.*

Proof. Since CC implies $f = \Pi_f(\boldsymbol{\sigma}, \mathcal{K})$ for *all* profiles $\boldsymbol{\sigma}$ and clone decomposition $\mathcal{K}$, a single counterexample is sufficient to show a rule failing CC.

For STV and AS, we will be using $\boldsymbol{\sigma}$ over $A = \{a_1, a_2, b, c\}$ from Figure 2. Consider $\mathcal{K} = \{K_a, K_b, K_c\}$, with $K_a = \{a_1, a_2\}$, $K_b = \{b\}$, and $K_c = \{c\}$. Figure 3 shows $\boldsymbol{\sigma}^{\mathcal{K}}$ and $\boldsymbol{\sigma}|_{K_a}$. The procedure for STV is detailed in Examples 3 and 6. To run AS on $\boldsymbol{\sigma}$, we will alternate between eliminating all non-Smith candidates and eliminating the candidate with the least plurality score:

- First, we have $Sm(\boldsymbol{\sigma}) = A$ due to the cyclicity of the profile, hence no one gets eliminated.
- Then, we eliminate a_2, the candidate with the least plurality score.
- Once again, $Sm(\boldsymbol{\sigma} \setminus \{a_2\}) = A \setminus \{a_2\}$, so no candidate is eliminated.
- c gets eliminated as the next candidate with least plurality scores.
- $Sm(\boldsymbol{\sigma} \setminus \{b, a_2\}) = \{a_1\}$ since a_1 pairwise defeats b.

Therefore $AS(\boldsymbol{\sigma}) = \{a_1\}$. Running AS on $\boldsymbol{\sigma}^{\mathcal{K}}$, on the other hand, we get:

- We have $Sm(\boldsymbol{\sigma}^{\mathcal{K}}) = \mathcal{K}$ due to the cyclicity of the profile, so no (meta-)candidate is eliminated.
- Then, we eliminate K_c, the candidate with the least plurality score.
- $Sm(\boldsymbol{\sigma}^{\mathcal{K}} \setminus \{K_c\}) = \{K_a\}$ since K_a pairwise defeats K_b.

Therefore $AS(\boldsymbol{\sigma}^{\mathcal{K}}) = \{K_a\}$. Further, $AS(\boldsymbol{\sigma}|_{K_a}) = \{a_2\}$ since a_1 is pairwise defeated by a_2 and therefore eliminated in the first step. Thus, we have $AS(\boldsymbol{\sigma}) = \{a_1\} \neq \{a_2\} = \Pi_{AS}(\boldsymbol{\sigma}, \mathcal{K})$, proving AS is not CC. For both STV and AS, the main idea is that a_2 gets eliminated first even though it is a majority winner over a_1, since the few voters that prefer a_1 over a_2 happens to put them to the top of their ballot, giving a_1 more plurality votes than a_2.

For BP and SC, consider the following profile $\boldsymbol{\sigma}'$ (the same as the one from Figure 1, with relabeled candidates):

6 voters	5 voters	2 voters	2 voters
a_1	c	b	b
a_2	a_2	c	c
b	a_1	a_1	a_2
c	b	a_2	a_1

To find $BP(\boldsymbol{\sigma}')$ and $SC(\boldsymbol{\sigma}')$, we construct the margin matrix $M_{\boldsymbol{\sigma}'}$ below:

$M_{\boldsymbol{\sigma}'}$	a_1	a_2	b	c
a_1	0	1	7	-3
a_2	-1	0	7	-3
b	-7	-7	0	5
c	3	3	-5	0

Notice that there are 3 simple cycles in $M_{\boldsymbol{\sigma}'}$: (a_1, b, c), (a_2, b, c), and (a_1, a_2, b, c), with smallest margins $[c, a_1]$, $[c, a_2]$, and $[a_1, a_2]$, respectively. Removing these three edges form the graph leaves a_1 and a_2 without any incoming edges, indicating $SC(\boldsymbol{\sigma}') = \{a_1, a_2\}$.

Similarly, $M_{\boldsymbol{\sigma}'}$ induces the following strength matrix $S_{\boldsymbol{\sigma}'}$, which shows that $BP(\boldsymbol{\sigma}') = \{a_1, a_2\}$, since $S[x, y] \geq S[y, x]$ for $x \in \{a_1, a_2\}$ and all $y \in A$.

$S_{\boldsymbol{\sigma}'}$	a_1	a_2	b	c
a_1	0	3	7	5
a_2	3	0	7	5
b	3	3	0	5
c	3	3	3	0

However, using clone decomposition $\mathcal{K}$ from above (which is also a valid decomposition with respect to $\boldsymbol{\sigma}'$), the graph for $M_{\boldsymbol{\sigma}'^{\mathcal{K}}}$ is composed of a single simple cycle, with $M[K_a, K_b] = 7$, $M[K_b, K_c] = 5$ and $M[K_c, K_a] = 3$. Clearly, we have $SC(\boldsymbol{\sigma}'^{\mathcal{K}}) = BP(\boldsymbol{\sigma}^{\mathcal{K}}) = \{K_a\}$. However, a_1 is the majoritarian winner against a_2 in $\boldsymbol{\sigma}'|_{K_a}$, without any cycles. Hence $\prod_{SC}(\boldsymbol{\sigma}', \mathcal{K}) = \prod_{BP}(\boldsymbol{\sigma}', \mathcal{K}) = \{a_1\}$, showing both rules fail CC. Intuitively, both BP and SC, while picking their winners for $\boldsymbol{\sigma}'$, 'discard' the relationship between a_1 and a_2, BP since neither the strongest path from a_1 to a_2 nor vice versa go through the (a_1, a_2) edge, and SC since the (a_1, a_2) edge forms the weakest margin in a 4-candidate cycle. As a result, both rules pick both candidates as winner, even though they both agree a_1 wins over a_2 when applied to $\boldsymbol{\sigma}'|_{K_a}$ alone. □

D.3 Proof of Theorem 2

We now prove our main positive characterization result from Section 3.

Theorem 2. *RP_i is composition-consistent for any fixed $i \in N$.*

Without loss of generality, fix $1 \in N$. We will show that RP_1 satsifes CC. The same proof follows for RP_i for any $i \in N$. We write $\{a, b\} \succ_{\Sigma_1} \{c, d\}$ if Σ_1 ranks $\{a, b\}$ before $\{c, d\}$. Zavist and Tideman show that Σ_1 is *impartial*; that is, for all $a, b, c, d \in A$, if $\{a, c\} \succ_{\Sigma_1} \{b, c\}$ then $\{a, d\} \succ_{\Sigma_1} \{b, d\}$. Using Σ_1, we construct a complete *priority order* $\mathcal{L}$ over ordered pairs: pairs are ordered (in decreasing order) according to M, and ties are broken by Σ_1 (and according to σ_1 when $M[a, b] = 0$). Formally, given distinct ordered pairs (a, b) and (c, d) such that $(c, d) \neq (b, a)$, we have:

$$(a, b) \succ_{\mathcal{L}} (c, d) \text{ iff: } \begin{cases} M[a, b] > M[c, d] \text{ or} \\ M[a, b] = M[c, d], \{a, b\} \succ_{\Sigma_1} \{c, d\}, \end{cases}$$

and if $(c, d) = (b, a)$ we have:

$$(a, b) \succ_{\mathcal{L}} (b, a) \text{ iff: } \begin{cases} M[a, b] > M[b, a] \text{ or} \\ M[a, b] = M[b, a] = 0, a \succ_{\sigma_1} b. \end{cases}$$

Then, *the Ranked Pairs method using voter 1 as a tie-breaker* (hereon referred to as RP_1) add edges from M to a digraph according to $\mathcal{L}$, skipping those that create a cycle.

Zavist and Tideman [65] show that RP_1 is indeed IoC. We now strengthen this result:

Proof. Zavist and Tideman [65] show that the original RP rule (without tie-breaking) has an equivalent definition using "stacks". We introduce an analogous notion and equivalency with respect to a specific $\mathcal{L}$.

Definition 42. Given a complete ranking R over candidates A and a priority order over ordered pairs $\mathcal{L}$, we say x *attains* y through R and with respect to $\mathcal{L}$ if there exists a sequence of candidates $a_1, a_2, \ldots, a_j$ such that $a_1 = x$, $a_j = y$ and for all $i \in [j-1]$, we have $a_i \succ_R a_{i+1}$ and $(a_i, a_{i+1}) \succ_{\mathcal{L}} (a_j, a_1)$. We say R is a *stack* with respect to $\mathcal{L}$ if $x \succ_R y$ implies x attains y through R with respect to $\mathcal{L}$.

Lemma 43. *RP_1 with Σ_1 as a tie-breaker will pick candidate a as a winner if and only if there exists a stack with respect to $\mathcal{L}$ that ranks a first, where $\mathcal{L}$ is the priority order over ordered pairs constructed using Σ_1 as a tie-breaker.*

Proof. $(\Rightarrow)$: Say a is the RP_1 winner with $\mathcal{L}$ as the priority order over ordered pairs (constructed from Σ_1). Notice that the final graph from the RP procedure will be a DAG. Say R is the topological ordering of this DAG and a is the source node (hence ranked first by R), which we call the *winning ranking*. By definition, the rule will pick a as the winner. For any $x, y \in A$ such that $x \succ_R y$, the edge (y, x) was skipped in the RP procedure, implying it would have created a cycle. Hence, there exists candidates $a_1, \ldots a_j$ such that $a_1 = x$ and $a_j = y$, and each (a_i, a_{i+1}) was locked in the RP graph before (y, x) was considered, implying $(a_i, a_{i+1}) \succ_{\mathcal{L}} (y, x) = (a_j, a_1)$. Moreover, since each (a_i, a_{i+1}) was locked, we must have $a_i \succ_R a_{i+1}$ in the final ranking. This implies R is indeed a stack with respect to $\mathcal{L}$, with a ranked first.
$(\Leftarrow)$: Say R is a stack with respect to $\mathcal{L}$, with a ranked first. We argue this is the final ranking produced by running RP_1 with $\mathcal{L}$ as priority order (constructed using Σ_1). Assume instead that RP outputs final ranking R^* with $R^* \neq R$. Then there exists at least one pair x, y such that $x \succ_R y$ but $y \succ_{R'} x$, so (y, x) was locked by the RP procedure. Of all such pairs, say x^*, y^* is the one where (y^*, x^*) was locked by the RP procedure first. Since $x^* \succ_R y^*$ and since R is a stack with respect to $\mathcal{L}$, there exists a series of candidates $a_1, \ldots a_j$ such that $a_1 = x^*$, $a_j = y^*$, and for all $i \in [j-1]$, we have $a_i \succ_R a_{i+1}$ and $(a_i, a_{i+1}) \succ_{\mathcal{L}} (a_j, a_1) = (y^*, x^*)$. Since $(a_i, a_{i+1}) \succ_{\mathcal{L}} (y^*, x^*)$ for all i, all such edges were considered by the RP procedure before (y^*, x^*). At least one of these edges must have been skipped, otherwise locking (y^*, x^*) would have caused a cycle. Say (a_k, a_{k+1}) was the first edge that was skipped. This implies locking this edge would have caused a cycle with the already-locked edges; however, since $a_k \succ_R a_{k+1}$, this cycle must contain an edge (z, ℓ) such that $\ell \succ_R z$. However, this implies $z \succ_{R'} \ell$ in the final ranking and that $(z, \ell) \succ_{\mathcal{L}} (a_k, a_{k+1}) \succ_{\mathcal{L}} (y^*, x^*)$. Since (y^*, x^*) was assumed to be the first such edge to be considered, this is a contradiction. □

Note that since RP_1 results in a single unique ranking over candidates (as a single tie-breaker is fixed), the proof of Lemma 43 also shows that there is a unique stack with respect to $\mathcal{L}$. We also use an existing lemma by Zavist and Tideman [65].

Lemma 44 (Zavist and Tideman 65, §VII)**.** *Say C is a clone set with respect to profile $\boldsymbol{\sigma}$. The winning ranking R resulting from running RP_1 on $\boldsymbol{\sigma}$ with an impartial tie-breaker Σ_1 based on a ranking σ_1 will have no element of $A \setminus C$ appear between two elements of C in R.*

We will now prove that RP_1 is composition consistent. Given $\boldsymbol{\sigma}$, say $\mathcal{K} = \{K_1, K_2, \ldots, K_k\}$ is a clone decomposition. Say $\boldsymbol{\sigma}^{\mathcal{K}}$ is the summary of $\boldsymbol{\sigma}$ with respect to $\mathcal{K}$ (where the clone sets in each σ_i is replaced by the meta candidates $\{K_i\}_{i\in[k]}$) and $\boldsymbol{\sigma}|_{K_i}$ is $\boldsymbol{\sigma}$ restricted to the candidates in K_i. We would like to show that $RP_1(\boldsymbol{\sigma}) = \bigcup_{K\in RP_1(\boldsymbol{\sigma}^{\mathcal{K}})} RP_1(\boldsymbol{\sigma}|_K)$. Since RP with a specific tie-breaking order always produces a single unique winner, showing containment in a single direction is sufficient.

Say $RP_1(\boldsymbol{\sigma}) = \{a\}$, implying a comes first in the winning ranking R. By the proof of the forward direction of Lemma 43, R is a stack with respect to $\mathcal{L}$ (the order that the RP procedure follows, which uses tie-breaking order Σ_1 based on vote σ_1). By Lemma 44, each $K_i \in \mathcal{K}$ appears as an interval in R, hence we can define a corresponding ranking $R^{\mathcal{K}}$ over clone sets in $\mathcal{K}$. We would like to show that $R^{\mathcal{K}}$ is a stack with respect to $\mathcal{L}^{\mathcal{K}}$, which is the order of ordered pairs in $\mathcal{K}$ according to decreasing order of $M^{\mathcal{K}}$ (the margin matrix of $\boldsymbol{\sigma}^{\mathcal{K}}$), using $\sigma_1^{\mathcal{K}}$ (voter 1's vote in the summary) as a tie-breaker.

We can relabel the clone sets in $\mathcal{K}$ such that $R^{\mathcal{K}} = (K_1 \succ K_2 \succ \ldots \succ K_k)$. Since a is the ranked first in R, we have $a \in K_1$. Notice that if $k = 1$, then $R^{\mathcal{K}}$ vacously. Otherwise, take any K_x, K_y such that $K_x \succ_{R^{\mathcal{K}}} K_y$. Say x is the element of K_x that appears last in R and y is the element of K_y that appears first in R. Since R is a stack with respect to $\mathcal{L}$, and since $x \succ_R y$, there exists a sequence of candidates $a_1, \ldots a_j$ and for all $i \in [j-1]$, we have $a_i \succ_R a_{i+1}$ and $(a_i, a_{i+1}) \succ_{\mathcal{L}} (a_j, a_1)$. Since the a_i in this sequence appear according to their order in R, by Lemma 44, consecutive candidates in the sequence $a_1, \ldots, a_j$ can be grouped up to form a sequence $K'_1, \ldots K'_{j'}$ such that $K'_{i'} \in \mathcal{K}$ for each $i' \in [j']$, $K'_1 = K_x$, $K'_{j'} = K_y$, and $K'_{i'} \succ_{R^{\mathcal{K}}} K'_{i'+1}$ for each $i' \in [j'-1]$. Notice that since x and y are in different clone sets, $j' > 1$. Take any $i' \in [j'-1]$ and consider the last element $K'_{i'}$ and the first element of $K'_{i'+1}$ to appear in $(a_1, a_2, \ldots, a_j)$. By construction, these two elements appear consecutively in $(a_1, a_2, \ldots, a_j)$, so they are a_i and a_{i+1}, respectively, for some $i \in [j]$. Since $(a_i, a_{i+1}) \succ_{\mathcal{L}} (a_j, a_1) = (y, x)$, based on the way $\mathcal{L}$ was constructed, there are two possible cases:

1. $M[a_i, a_{i+1}] > M[y, x]$, in which case we must have $M^{\mathcal{K}}[K'_{i'}, K'_{i'+1}] = M[a_i, a_{i+1}] > M[y, x] = M^{\mathcal{K}}[K_y, K_x]$ by definition of clones, and hence $(K'_{i'}, K'_{i'+1}) \succ_{\mathcal{L}^{\mathcal{K}}} (K_y, K_x) = (K'_{j'}, K'_1)$.

2. $M[a_i, a_{i+1}] = M[y, x]$. In this case, we also have $M^{\mathcal{K}}[K'_{i'}, K'_{i'+1}] = M^{\mathcal{K}}[K_y, K_x]$ by definition of clones. However, since $(a_i, a_{i+1}) \succ_{\mathcal{L}} (y, x)$, there are four options:

   (a) $i' = 1$ and $i' + 1 = j'$, so $a_i = x$ and $a_{i+1} = y$. In this case, $(a_i, a_{i+1}) \succ_{\mathcal{L}} (y, x)$ implies $x \succ_{\sigma_1} y$ and hence $K_x \succ_{\sigma_1^{\mathcal{K}}} K_y$ by definition of clone sets, and hence: $(K'_{i'}, K'_{i'+1}) = (K_x, K_y) \succ_{\mathcal{L}^{\mathcal{K}}} (K_y, K_x) = (K'_{j'}, K'_1)$

   (b) $i' = 1$ and $i' + 1 \neq j'$, so $a_i = x$ and $a_{i+1} \neq y$. In this case, $(a_i, a_{i+1}) \succ_{\mathcal{L}} (y, x)$ implies $a_{i+1} \succ_{\sigma_1} y$ and hence $K'_{i'+1} \succ_{\sigma_1^{\mathcal{K}}} K_y$ by definition of clone sets, and hence: $(K'_{i'}, K'_{i'+1}) = (K_x, K'_{i'+1}) \succ_{\mathcal{L}^{\mathcal{K}}} (K_y, K_x) = (K'_{j'}, K'_1)$.

   (c) $i' \neq 1$ and $i' + 1 = j'$, so $a_i \neq x$ and $a_{i+1} = y$. In this case, $(a_i, a_{i+1}) \succ_{\mathcal{L}} (y, x)$ implies $a_i \succ_{\sigma_1} x$ and hence $K'_{i'} \succ_{\sigma_1^{\mathcal{K}}} K_x$ by definition of clone sets, and hence: $(K'_{i'}, K'_{i'+1}) = (K'_{i'}, K_y) \succ_{\mathcal{L}^{\mathcal{K}}} (K_y, K_x) = (K'_{j'}, K'_1)$.

   (d) $i' \neq 1$ and $i' + 1 \neq j'$, so $a_i \neq x$ and $a_{i+1} \neq y$.
   In this case, $(a_i, a_{i+1}) \succ_{\mathcal{L}} (y, x)$ implies for some $\alpha \in \{0, 1\}$, we have $a_{i+\alpha} \succ_{\sigma_1} z$ for each $z \in \{x, y, a_{i+1-\alpha}\}$, and hence $K'_{i'+\alpha} \succ_{\sigma_1^{\mathcal{K}}} Z$ for each $Z \in \{K_x, K_y, K'_{i'+1-\alpha}\}$ by definition of clone sets, and hence: $(K'_{i'}, K'_{i'+1}) \succ_{\mathcal{L}^{\mathcal{K}}} (K_y, K_x) = (K'_{j'}, K'_1)$.

In each case, we end up having $(K'_{i'}, K'_{i'+1}) \succ_{\mathcal{L}^{\mathcal{K}}} (K'_{j'}, K'_1)$, which proves that K_x attains K_y through $R^{\mathcal{K}}$ with respect to $\mathcal{L}^{\mathcal{K}}$, and hence that $R^{\mathcal{K}}$ is a stack with respect to $\mathcal{L}^{\mathcal{K}}$. By Lemma 43, this implies $RP_1(\boldsymbol{\sigma}^{\mathcal{K}}) = \{K_1\}$, as K_1 comes first in $R^{\mathcal{K}}$.

Since $a \in K_1$, a will be a competing candidate in $\boldsymbol{\sigma}|_{K_1}$. Again by Lemma 44, we know that all elements of K_1 appears as a block in the start of R. Say $R|_{K_1}$ is this section of R. We would like to show that

$R|_{K_1}$ is a stack with respect to $\mathcal{L}^{K_1}$, which is the priority order of ordered pairs in $\mathcal{K}$ according to decreasing order of M^{K_1} (the margin matrix of $\boldsymbol{\sigma}|_{K_1}$), using $\sigma_1|_{K_1}$ (voter 1's vote restricted to K_1) as a tie-breaker. Note that for any $a, b, c, d \in K_1$, $(a, b) \succ_{\mathcal{L}} (c, d)$ implies $(a, b) \succ_{\mathcal{L}^{K_1}} (c, d)$, since $\mathcal{L}$ is entirely based on pairwise comparisons and the relative ranking of candidates in σ_1, neither of which is affected by the deletion of candidates in $A \setminus K_1$ and hence is directly carried to $\mathcal{L}^{K_1}$. Now take any $x, y \in K_1$ such that $x \succ_{R|_{K_1}} y$. Since $R|_{K_1}$ is just an interval of R, we must have $x \succ_R y$. Since R is a stack, this implies there exists a sequence of candidates $a_1, a_2, \ldots, a_j$ such that $a_1 = x$, $a_j = y$ and for all $i \in [j-1]$, we have $a_i \succ_R a_{i+1}$ and $(a_i, a_{i+1}) \succ_{\mathcal{L}} (a_j, a_1)$. Since all elements of K_1 appear as an interval in R by Lemma 44, $x = a_1 \succ_R a_2 \succ_R \ldots \succ_R a_j = y$ and $x, y \in K_1$ implies $a_i \in K_1$ for all $i \in [j]$. This implies $a_i \succ_{R|_{K_1}} a_{i+1}$ and $(a_i, a_{i+1}) \succ_{\mathcal{L}^{K_1}} (a_j, a_1)$, implying $R|_{K_1}$ is a stack with respect to $\mathcal{L}^{K_1}$. Since a is first in $R|_{K_1}$, by Lemma 43, this implies $RP_1(\boldsymbol{\sigma}|_{K_1}) = \{a\}$. Since $= RP_1(\boldsymbol{\sigma}^{\mathcal{K}}) = \{K_1\}$, we have $\bigcup_{K \in RP_1(\boldsymbol{\sigma}^{\mathcal{K}})} f(\boldsymbol{\sigma}|_K) = \{a\} = RP_1(\boldsymbol{\sigma})$, completing the proof. □

D.4 Proof of Proposition 9

We next prove that $RP_N(\boldsymbol{\sigma}) = \cup_{i \in [n]} RP_i(\boldsymbol{\sigma})$, which recovers anonymity for RP_i while preserving independence of clones, loses composition-consistency.

Proposition 9. *RP_N is independent of clones, but not composition-consistent.*

Proof. IoC holds as RP_i is IoC for all $i \in N$. For CC, consider $\boldsymbol{\sigma}$ with $n = 2$, $A = \{a, b, c\}$, and $a \succ_1 b \succ_1 c$ and $c \succ_2 b \succ_2 a$. We have $RP_N(\boldsymbol{\sigma}) = \{a, c\}$. Using $\mathcal{K} = \{K, \{c\}\}$ with $K = \{a, b\}$, we get $RP_N(\boldsymbol{\sigma}^{\mathcal{K}}) = \{K, \{c\}\}$ and $RP_N(\boldsymbol{\sigma}|_K) = \{a, b\}$. Hence, $\Pi_{RP_N}(\boldsymbol{\sigma}, \mathcal{K}) = \{a, b, c\} \neq RP_N(\boldsymbol{\sigma})$. □

E On Section 4 (CC Transformation)

In this section, we provide the proofs omitted from Section 4 of the main body, as well as an extended discussion of PQ-trees and clone-aware axioms.

E.1 Extended discussion of clone structures and PQ-trees

Here, we expand on our discussion of the PQ-trees, first defined by Booth and Lueker [2] and later used by Elkind et al. [23] for representing clone sets. For the full set of formal definitions, see Elkind et al. [23].

Given $\boldsymbol{\sigma}$ we can use a PQ-tree to represent its *clone structure* $\mathcal{C}(\boldsymbol{\sigma}) \subseteq \mathcal{P}(A)$, which is the collection of *all* clone sets on $\boldsymbol{\sigma}$. For example, if $\boldsymbol{\sigma}$ is the profile from Figure 1, then $\mathcal{C}(\boldsymbol{\sigma}) = \{\{a\}, \{b\}, \{c\}, \{d\}, \{b, c\}, \{a, b, c, d\}\}$. Given a set of candidates $A = \{a_i\}_{i \in [m]}$, a PQ-tree T over A is an ordered tree where the leaves of the tree correspond to a particular permutation of the elements of A. Each internal node is either of "type P" or of "type Q". If a node is of type P, then its children can be permuted arbitrarily. If a node is of type Q, then the only allowable operation is the reversal of its children's order.

Elkind et al. [23] begin by defining two special types of clone structures: a *maximal* clone structure (which they also call a *string of sausages*) and a *minimal* clone structure (also called a *fat sausage*). A string of sausages corresponds to the clone structure that arises when all rankings in the profile $\boldsymbol{\sigma}$ consist of a single linear order (WLOG, $\sigma_1 : a_1 \succ a_2 \succ \cdots \succ a_m$) or its reversal. Then $\mathcal{C}(\boldsymbol{\sigma}) = \{\{a_k\}_{i \leq j} : i \leq j\}$, meaning that the clone structure contains all intervals of candidates in σ_1. The *majority ranking* of the Q-node is σ_1 or its reverse, depending on which one appears more in $\boldsymbol{\sigma}$. The "opposite" scenario (a fat sausage) is when $\mathcal{C}(\boldsymbol{\sigma}') = \{A\} \cup \{\{a_i\}\}_{i \in [m]}$, meaning we only have the trivial clone sets in our structure.

This arises, for example, when $\boldsymbol{\sigma}'$ corresponds to a cyclic profile on A, *i.e.*, $\boldsymbol{\sigma}' = (\sigma'_1, \ldots, \sigma'_m)$, and the preferences of the i-th voter are given by $\sigma'_i : a_i \succ_{\sigma_i} a_{i+1} \succ_{\sigma_i} \cdots \succ_{\sigma_i} a_m \succ_{\sigma_i} a_1 \succ_{\sigma_i} \cdots \succ_{\sigma i} a_{i-1}$.

We need a few more definitions before describing the construction of the PQ-tree. Let $\mathcal{F}$ be a family of subsets on a finite set F, and likewise $\mathcal{E}$ for E, where $F \cap E = \emptyset$. Then, we can *embed* $\mathcal{F}$ into $\mathcal{E}$ as follows: given $e \in E$, we replace each set X containing e with $(X \setminus \{e\}) \cup F$. The resulting family of subsets is denoted by $\mathcal{E}(e \to \mathcal{F})$. The inverse operation of embedding is called *collapsing*; note that for a family of subsets $\mathcal{C}$ on A to be collapsible, it should contain a set that does not intersect non-trivially (*i.e.*, not as a sub/superset) with any other set in $\mathcal{C}$), which motivates the definition of a *proper subfamily* of $\mathcal{F}$:

Definition 45. Let $\mathcal{F}$ be a family of subsets on a finite set F. A subset $\mathcal{E} \subseteq \mathcal{F}$ is called a *proper subfamily* of $\mathcal{F}$ if there is a set $E \in \mathcal{F}$ such that (i) $\mathcal{E} = \{F \in \mathcal{F} \mid F \subseteq E\}$; (ii) for any $X \in \mathcal{F} \setminus \mathcal{E}$, either $E \subseteq X$ or $X \cap E = \varnothing$, (iii) E is a proper subset of F. A family of subsets with no proper subfamily is called *irreducible*.

The key result that we require in the construction of a PQ-tree is that *any irreducible clone structure is either a fat sausage or a string of sausages* [23, Thm. 3.10]. Given a clone structure $\mathcal{C} \subseteq \mathcal{P}(A)$, we construct its corresponding PQ-tree $T(\mathcal{C})$ iteratively:

1. Pick some non-singleton, irreducible minimal set of clones $\mathcal{E}_1 \subseteq \mathcal{C}$. By Definition 45, there exists $C_1 \in \mathcal{C}$ such that $\mathcal{E}_1 = \{F \in \mathcal{F} \mid F \subseteq C_1\}$.

2. Update $\mathcal{C}$ to $\mathcal{C}(\mathcal{E}_1 \to C_1)$, *i.e.*, substitute all appearances of the members of C_1 in $\mathcal{C}$ by a meta-candidate C_1, and remove the sets in $\mathcal{C}$ that correspond to subsets of C_1. Since C_1 is either a subset of a superset of each $K \in \mathcal{C}$ is overlaps with, this transformation is well-defined.

3. Build the subtree for C_1. By Theorem 3.10 in Elkind et al. [23], C_1 is either a fat sausage or a string of sausages. If it is a fat sausage, then C_1 is set to be of type P- and label it as the $\odot$-product of the candidates in C_1. If $|C_1| = 2$, then it is both a fat sausage and a string of sausages. In this case, we treat it as a string of sausages sausage (following the convention by Elkind et al. [23]). The candidates in C_1 are placed as the children leaves in the subtree. If it is a string of sausages, then C_1 is of type Q-, and we label it as the $\oplus$-product of the candidates in C_1. The candidates in C_1 are similarly placed as the children leaves in the subtree, following the order dictated by C_1.

4. We repeat the previous three steps for C_i, with $i = 2, \ldots$, until we cannot find any non-singleton, irreducible, minimal set. For any child node of C_i that corresponds to a previously-collapsed subset C_j with $j < i$, the node is replaced with the subtree of C_j, already constructed by assumption. For child nodes of C_i that correspond to original candidates from A, the node is a leaf.

5. Eventually, no proper irreducible subfamilies are left, and all of the remaining candidates form either a string of sausages or a fat sausage, so we place them as children of the root of $T(\mathcal{C})$, similarly labeling it as type P or Q.

The order in which we choose C_i does not impact the final construction, as the irreducible proper subsets of a clone structure $\mathcal{C}$ is non-overlapping [23, Proposition 4.2.], implying a unique decomposition of candidates into irreducible proper subsets at each step. This ensures that the PQ-tree of a preference profile is unique [32].

E.2 Discussion of PQ-tree algorithms

The original PQ-tree algorithm is due to Booth and Luecker, who introduced it as a way to represent a family of permutations on a set of elements [2]. Later, Elkind et al. [23] showed its use in the

context of computational social choice. Cornaz, Galand, and Spanjaard carefully analyze the Booth and Luecker algorithm in the context of voting rules and establish the runtime of $O(nm^3)$ that we use in Lemma 11 [16]. In this section, we provide some more context on the general relationship between PQ-tree constructions (and tournament decomposition tree constructions) with the graph theoretic literature on modular tree decomposition (particularly with the modular tree decomposition algorithm by Capelle et al. [11], following the observations made by Brandt et al. [5].

Brandt et al. [5] study CC tournament solutions, following the definition of composition consistency for tournaments first given by [36]. They provide a *decomposition tree of a tournament* T meant for efficiently implementing CC tournament solutions. In their analysis, they use what they call the *decomposition degree* of a tournament, which is a parameter that reflects its decomposability (the lower the degree, the better well-behaved its decomposition). Their decomposition tree is the tournament version of the PQ-tree construction of Elkind et al. [23]: both use trees with two different types of internal nodes as a suitable way of representing clone structures. The correspondence between the two constructions is the following:

1. Elkind et al. [23] use a PQ-tree to represent clone structures given a profile $\boldsymbol{\sigma}$, while Brandt et al. [5] use a decomposition tree to represent components (Definition 21) given a tournament T.
2. Elkind et al. [23] divide the internal nodes into types P and Q, whereas Brandt et al. [5] calls them *irreducible* and *reducible*, respectively (but the definition is the same one).

Besides the naming of the internal nodes and the difference between having a profile σ versus a tournament T as input, given the equivalence between components in T and clones in σ, Brandt et al. [5]'s definition of a decomposition tree of T is equivalent to Elkind et al. [23]'s definition of a PQ-tree for σ. In particular, claims relating to the running time required to compute a decomposition tree of a tournament can be transferred to the running time required to compute PQ-trees.

Brandt et al. [5] make the following two observations about the running time required to compute the decomposition tree of a tournament T:

1. First, we compute a *factorization permutation* of T, which is a permutation of the alternatives in A such that each component of T is a contiguous interval in the permutation. McConnell and De Montgolfier [46] provide a linear time algorithm for computing a factorizing permutation of a tournament in linear time
2. Second, given T and a factorization permutation of T, we can use the graph theoretic algorithm by Capelle et al. [11] to obtain the decomposition tree of T.

In our settings of profiles, we can adapt the running time argument from [5] as follows:

1. In the case of profiles $\boldsymbol{\sigma}$, we do not need to do more work to compute the factorization permutation of T; we can read it directly from $\boldsymbol{\sigma}$ (from any one voter's ranking). By definition of a clone set, *every* voter ranks the members of a clone set continguously in their ranking. Therefore, every single voter's ranking is a factorization permutation of $\boldsymbol{\sigma}$. Thus, computing the factorization permutation requires $O(|A|)$ running time.
2. The graph theoretic algorithm Capelle et al. [11] that Brandt et al. [5] use for computing decomposition tree tournaments is not directly related to tournaments (or to computational social choice). Rather, Capelle et al. [11] deal with a broad definition of a *modular decomposition* of a directed graph. Zooming out from tournaments, for a given graph $G = (V, E)$, a decomposition tree T_G is such that the vertices of G are in one-to-one correspondence with the leaves of T_G, and the internal nodes correspond to subsets of V. They call the nodes of a decomposition tree (and the sets of vertices they induce) *decomposition sets*. They study the general case where the decomposition sets correspond to *modules*:

Definition 46. A *module* in a graph $G = (V, E)$ is a set X of vertices such that 1) if $y \in V \setminus X$, then y has either directed edges to all members of X or to none of them, and 2) all members of X have either directed edges to y, or none of them do.

Intuitively, a set of vertices in a graph forms a module if every vertex in $V \setminus X$ has a "uniform" relationship to all members of X [46]. Note that the definition of a module imposes no requirements on whether the vertices in X should be connected or not. Observe also that connected components are a particular case of modules. A module is *strong* if it does not overlap with any other module. Then, Capelle et al. [11] call the decomposition tree of a graph G into its strong modules the *modular decomposition tree* of G, and they provide a (complicated) linear time algorithm for computing it (which we can treat as a black-box algorithm).

The literature on modular decomposition graphs is extensive and a popular topic in graph theory. However, as noted in Brandt et al. [5], the literature on composition-consistency (and in social choice more broadly) and on modular decompositions in graph theory is not well-connected. In this section, we help clarify part of this connection by detailing how we can use the modular decomposition algorithm by Capelle et al. [11] to compute the PQ-tree. Given that the notion of a module is the graph-theoretic generalization of clone sets in profiles and components in tournaments, we hope that there can be further interesting connections between the two fields.

As observed by Brandt et al. [5], to compute the decomposition tree of a tournament, we can simply input the graph induced by the tournament (*i.e.*, we draw an edge from a to b if a beats b) to the modular decomposition tree algorithm of Capelle et al. [11]. In our case, for computing the PQ-tree using the algorithm of [11], we need to input a graph G built from $\boldsymbol{\sigma}$ such that the modules of G are in bijection with the clone sets of $\boldsymbol{\sigma}$.

E.3 Clone-aware axioms

First, we introduce three axioms show that they are not necessarily satisfied by f^{CC} (Definition 12), even if f satisfies them, implying our CC transformation does not preserve them. We will then introduce *clone-aware* relaxations of these axioms, which are in fact preserved by the CC transformation (see Theorem 3).

Definition 47. An SCF f satisfies *monotonicity* if $a \in f(\boldsymbol{\sigma})$ implies $a \in f(\boldsymbol{\sigma}')$ if for all $i \in N$ and $b, c \in A \setminus \{a\}$, we have $a \succ_{\sigma_i} b \Rightarrow a \succ_{\sigma'_i} b$ and $b \succ_{\sigma_i} c \Rightarrow b \succ_{\sigma'_i} c$.

Inuitively, monotonicity dictates that promoting a winner in a profile while keeping all else constant should not cause them to lose. We see that monotonicity is not necessarily preserved by our CC transformation.

Example 48. *Consider Plurality Voting (PV), which is monotonic, and the profile $\boldsymbol{\sigma}$ from Figure 7a. Notice $\{a_1, a_2, a_3\}$ is a fat sausage and is grouped up by the PQ tree. $\{a_1, a_2, a_3\}$ wins against b in the root, and the a_1 wins against a_2 and a_3 with 5 plurality votes, hence $PV^{CC}(\boldsymbol{\sigma}) = \{a_1\}$. However, say* one *of the rightmost voters move a_1 up, submitting $a_1 \succ b \succ a_2 \succ a_3$ instead. Then there are no longer any nontrivial clone sets, and a_3 wins with 4 plurality votes (a_1 only has 3). Hence, with this new profile (call it $\boldsymbol{\sigma}$'), we have $PV^{CC}(\boldsymbol{\sigma}') = PV(\boldsymbol{\sigma}') = \{a_3\}$, showing that PV^{CC} is* not *monotone.*

Given a preference profile $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_n) \in \mathcal{L}(A)^n$ over voters $N = [n]$ and a new $(n+1)$th voter with ranking σ_{n+1} over A, we denote by $\boldsymbol{\sigma} + \sigma_{n+1}$ the profile $(\sigma_1, \sigma_2, \ldots, \sigma_n, \sigma_{n+1}) \in \mathcal{L}(A)^{n+1}$. Also, given any voter $i \in N \cup \{n+1\}$ with ranking σ_i over A and a non-empty subset $B \subseteq A$, we denote by $\max_i(B)$ the candidate in B that is ranked highest by σ_i. For example, if $\sigma_i = a \succ b \succ c \succ d$, and $B = \{b, c, d\}$, then $\max_i(B) = b$.

2 voters	4 voters	2 voters	3 voters
a_1	a_3	a_2	b
a_2	a_1	a_3	a_1
a_3	a_2	a_1	a_2
b	b	b	a_3

(a) Example profile $\boldsymbol{\sigma}$

6 voters	5 voters	2 voters	2 voters
a_1	c	b	b
a_2	a_2	c	c
b	a_1	a_1	a_2
c	b	z	a_1
z	z	a_2	z

(b) Example profile $\boldsymbol{\sigma}$

Figure 7: Two example profiles

Definition 49 (Brandt et al. 6)**.** An SCF f satisfies *(optimistic) participation* if given any profile $\boldsymbol{\sigma} \in \mathcal{L}(A)^n$ and any ranking $\sigma_{n+1} \in \mathcal{L}(A)$, we have $\max_{n+1}(f(\boldsymbol{\sigma})) \succeq_{n+1} \max_{n+1}(f(\boldsymbol{\sigma} + \sigma_{n+1}))$.

Participation dictates that a new voter cannot hurt themselves (in terms of their most preferred winner[10]) by participating in the election.

Example 50. *Once again, Plurality Voting (PV), which satisfies participation, and the profile $\boldsymbol{\sigma}$ from Figure 7a. As explained in Example 48, we have $PV^{CC}(\boldsymbol{\sigma}) = \{a_1\}$. Consider $\sigma_{n+1} : a_1 \succ b \succ a_2 \succ a_3$ and $\boldsymbol{\sigma}' = \boldsymbol{\sigma} + \sigma_{n+1}$. Since there are no non-trivial clone sets in $\boldsymbol{\sigma}'$, we have $PV^{CC}(\boldsymbol{\sigma}') = PV(\boldsymbol{\sigma}') = \{a_3\}$. Since $a_1 \succ_{n+1} a_3$, this shows PV^{CC} violates participation, and that someone ranking σ_{n+1} is better off staying away from this election.*

Given a set E and a family of its subsets $\mathcal{E} \subseteq 2^E$, for any $a \in E$, we denote $\mathcal{E} - \{a\} = \{K \setminus \{a\} : K \in \mathcal{E}\}$

Definition 51. An SCF f satisfies *independence of Smith-dominated alternatives (ISDA)* if given any profile $\boldsymbol{\sigma} \in \mathcal{L}(A)^n$ over candidates A and any candidate $a \in A$ such that $a \notin Sm(\boldsymbol{\sigma})$ and $\mathcal{C}(\boldsymbol{\sigma} \setminus \{a\}) = \mathcal{C}(\boldsymbol{\sigma}) - \{a\}$, we have $f(\boldsymbol{\sigma}) = f(\boldsymbol{\sigma} \setminus \{a\})$

In words, the winner(s) under any rule satisfying ISDA is not affected by the addition of a non-Smith candidate.

Example 52. *Consider Beatpath (BP), which satisfies ISDA [56] and the profile $\boldsymbol{\sigma}$ from Figure 7b, which is a minor modification from the counterexample for BP in the proof of Theorem 1. With z in the ballot, there are no non-trivial clone sets, so $BP^{CC}(\boldsymbol{\sigma}) = BP(\boldsymbol{\sigma}) = \{a_1, a_2\}$. Notice also that $Sm(\boldsymbol{\sigma}) = A \setminus \{z\}$, so z is indeed a non-Smith candidate. With z gone however, $\{a_1, a_2\}$ is a clone set again, and hence BP^{CC} first groups them up, picks $\{a_1, a_2\}$, and then picks a_1 in the restriction. Hence, $BP^{CC}(\boldsymbol{\sigma} \setminus \{z\}) = \{a_1\}$, showing that the CC-transformation does not necessarily preserve ISDA.*

The common thread in Examples 48, 50 and 52 is that the changes in profile (whether promoting a winner on a ranking or the addition/removal of a voter/candidate) significantly alters the clone structure of the profile, causing the behavioral of any f^{CC} to significantly change. Instead, we can relax each of these axioms by limiting the changes they consider to those that leave the clone structure unaffected. We present these relaxations (called the clone-aware version of each axiom) below. These new axioms implicitly assume that the clone structures are *inherent*, based on the candidates' location is some perceptual space (which is in fact the interpretation put forward by Tideman [62]), so any "realistic" change we will do to the profile will not alter the clone sets.

Definition 13. An SCF f satisfies *clone-aware monotonicity (monotonicity^{ca})* if $a \in f(\boldsymbol{\sigma})$ implies $a \in f(\boldsymbol{\sigma}')$ whenever (1) $\mathcal{C}(\boldsymbol{\sigma}) = \mathcal{C}(\boldsymbol{\sigma}')$ and (2) for all $i \in N$ and $b, c \in A \setminus \{a\}$, we have $a \succ_{\sigma_i} b \Rightarrow a \succ_{\sigma_i'} b$ and $b \succ_{\sigma_i} c \Rightarrow b \succ_{\sigma_i'} c$.

[10] One can alternatively use a pessimistic definition focusing on the new voter's lowest ranked candidate in the winner set.

Definition 53. An SCF f satisfies *clone-aware (optimistic) participation (participation^{ca})* if given any profile $\boldsymbol{\sigma} \in \mathcal{L}(A)^n$ and any ranking $\sigma_{n+1} \in \mathcal{L}(A)$ such that $\mathcal{C}(\boldsymbol{\sigma}) = \mathcal{C}(\boldsymbol{\sigma} + \sigma_{n+1})$, we have $\max_{n+1}(f(\boldsymbol{\sigma})) \succeq_{n+1} \max_{n+1}(f(\boldsymbol{\sigma} + \sigma_{n+1}))$.

Definition 54. An SCF f satisfies *clone-aware ISDA (ISDA^{ca})* if given any profile $\boldsymbol{\sigma} \in \mathcal{L}(A)^n$ over candidates A and any candidate $a \in A$ such that $a \notin Sm(\boldsymbol{\sigma})$ and $\mathcal{C}(\boldsymbol{\sigma} \setminus \{a\}) = \mathcal{C}(\boldsymbol{\sigma}) - \{a\}$, we have $f(\boldsymbol{\sigma}) = f(\boldsymbol{\sigma} \setminus \{a\})$

E.4 Proof of Theorem 3

In this section, we prove the theoretical guarantees of our CC-transform for SCFs.

Theorem 3. *For any neutral SCF f, f^{CC} satisfies: (1) If $\boldsymbol{\sigma}$ has no non-trivial clone sets, $f^{CC}(\boldsymbol{\sigma}) = f(\boldsymbol{\sigma})$; (2) f^{CC} is composition-consistent; (3) If f is composition-consistent, then $f^{CC} = f$,* i.e., *they agree for all $\boldsymbol{\sigma}$; (4) If f satisfies any of {anonymity, Condorcet consistency, Smith consistency, decisiveness (on all $\boldsymbol{\sigma}$), monotonicity^{ca}, ISDA^{ca}, participation^{ca}}, then f^{CC} satisfies this property as well; (5) Let $g(n, m)$ be an upper bound on the runtime of an algorithm that computes f on profiles with n voters and m candidates; then, $f^{CC}(\boldsymbol{\sigma})$ can be computed in time $O(nm^3) + m \cdot g(n, \delta(PQ(\boldsymbol{\sigma})))$.*

Proof. We prove each condition one by one.

Condition 1. We first prove an intermediary lemma.

Lemma 55. *Given neutral SCF f and profile $\boldsymbol{\sigma}$ over candidates A, we have $f(\boldsymbol{\sigma}) = \Pi_f(\boldsymbol{\sigma}, \mathcal{K}_{triv})$, where $\mathcal{K}_{triv} = \{\{a\}\}_{a \in A}$.*

Proof. Note that $\boldsymbol{\sigma}^{\mathcal{K}_{triv}}$ is isomorphic to $\boldsymbol{\sigma}$, with each $a \in A$ replaced with $\{a\}$. By neutrality, we must have $f(\boldsymbol{\sigma}^{\mathcal{K}_{triv}}) = \{\{a\}\}_{a \in f(\boldsymbol{\sigma})}$. Moreover, since an SCF always returns a non-empty subset, $f(\boldsymbol{\sigma}|_{\{a\}}) = \{a\}$ for any $a \in A$. This gives us

$$\Pi_f(\boldsymbol{\sigma}, \mathcal{K}_{triv}) = \bigcup_{K \in f(\boldsymbol{\sigma}^{\mathcal{K}_{triv}})} f(\boldsymbol{\sigma}|_K) = \bigcup_{a \in f(\boldsymbol{\sigma})} f(\boldsymbol{\sigma}|_{\{a\}}) = \bigcup_{a \in f(\boldsymbol{\sigma})} \{a\} = f(\boldsymbol{\sigma}).$$

□

If $\boldsymbol{\sigma}$ has no non-trivial clone sets, then $\mathcal{C}(\boldsymbol{\sigma})$ is a fat sausage, so the PQ tree of $\boldsymbol{\sigma}$ (say T) is simply a single P-node (say B) with all of the candidates in A as its children leaf nodes. Since $\text{decomp}(B, T) = \{\{a\}\}_{a \in A} = \mathcal{K}_{triv}$, Algorithm 1 simply outputs $f^{CC}(\boldsymbol{\sigma}) = \Pi_f(\boldsymbol{\sigma}, \mathcal{K}_{triv})$. By Lemma 55, this implies $f^{CC}(\boldsymbol{\sigma}) = f(\boldsymbol{\sigma})$.

Condition 2. The fact that f^{CC} is neutral follows from the neutrality of f and that $Algorithm\ 1$ is robust to relabeling of candidates. To prove f^{CC} satisfies CC, we first prove an important lemma.

Lemma 56. *Given neutral SCF f and profile $\boldsymbol{\sigma}$, say $\mathcal{K}, \mathcal{K}'$ are two clone decomposition with respect to $\boldsymbol{\sigma}$, such that $\mathcal{K} = \{K_1, K_2, \ldots, K_z\} \cup \{\{a\}\}_{a \in A \setminus (\bigcup_{i \in [z]} K_i)}$ for some $z \in \mathbb{Z}_{\geq 0}$, and there exists some $K \subseteq A \setminus \left(\bigcup_{i \in [z]} K_i\right)$ with $|K| > 1$ that satisfies $\mathcal{K}' = \mathcal{K} \setminus \mathcal{D} \cup \{K\}$, where $\mathcal{D} = \{\{a\}\}_{a \in K}$. In words, $\mathcal{K}'$ is the same decomposition as $\mathcal{K}$, except a group of singleton clone sets in $\mathcal{K}$ is now combined into a single new clone set K. Then $\Pi_{f^{CC}}(\boldsymbol{\sigma}, \mathcal{K}) = \Pi_{f^{CC}}(\boldsymbol{\sigma}, \mathcal{K}')$.*

The proof of Lemma 56 relies on the observation that the PQ trees for $\boldsymbol{\sigma}^{\mathcal{K}}$ and $\boldsymbol{\sigma}^{\mathcal{K}'}$ are identical, except the subtree(s) corresponding to $\mathcal{D}$ in the former (by Lemma 11) is replaced by a single leaf node K in the latter. Hence, we first show that Algorithm 1 proceeds identically on inputs $\boldsymbol{\sigma}^{\mathcal{K}}$ and $\boldsymbol{\sigma}^{\mathcal{K}'}$, picking the same set of leaves from $\mathcal{K} \setminus \mathcal{D}$ in both cases and returning *some* descendants of $\mathcal{D}$ in the former case if it returns K in the latter. We then show that *if* some descendants of $\mathcal{D}$ are returned by the algorithm on input $\boldsymbol{\sigma}^{\mathcal{K}}$, these are exactly the same as the output of the algorihtm when run on input $\boldsymbol{\sigma}|_K$. Combining these gives us the lemma statement.

Proof of Lemma 56. Say $\mathcal{K}, \mathcal{K}'$ satisfies the conditions in the lemma statement. Say T and T' are the PQ trees of $\boldsymbol{\sigma}^{\mathcal{K}}$ and $\boldsymbol{\sigma}^{\mathcal{K}'}$, respectively.[11] Given an interval node $\mathcal{B} \subseteq \mathcal{K}$ (resp., $\mathcal{B}' \subseteq \mathcal{K}'$) in T (resp., T'), we denote by $T(\mathcal{B})$ (resp., $T'(\mathcal{B}')$) the subtree of T (resp., T') rooteed at $\mathcal{B}$ (resp., $\mathcal{B}'$). We will be comparing the structure of T and T', using the fact that PQ trees are built by iteratively collapsing irreducible subfamilies (see Appendix E.1 above, and also Elkind et al. [23]). Since $\mathcal{D} = \{\{a\}\}_{a \in K}$ is a clone set with respect to $\boldsymbol{\sigma}^{\mathcal{K}}$ (which follows from the assumption that K is a clone set with respect to $\boldsymbol{\sigma}$), by Lemma 11, there are two options:

- $\mathcal{D}$ is a node of T, in which case its members are leaves of a subtree ($T(\mathcal{D})$). In this case, the tree T' is identical to T, except $T(\mathcal{D})$ is replaced by a single leaf node K. The PQ tree for $\boldsymbol{\sigma}|_K$, on the other hand, is exactly $T(\mathcal{D})$ (except the leaf for each singleton $\{a\}$ is replaced with the leaf for a).
- $\mathcal{D}$ union of an interval of nodes ($\{B_k(\mathcal{B}, T)\}_{i \leq k \leq j}$ for some $i < j$) that are adjacent children of the same Q-node $\mathcal{B} \subseteq \mathcal{K}$, in which case its members are leaves of the same interval of subtrees ($\{T(B_k(\mathcal{B}, T))\}_{i \leq k \leq j}$) . In this case, the tree T' is identical to T, except the children of $\mathcal{B}$ corresponding to $\mathcal{D}$ ($\{T(B_k(\mathcal{B}, T))\}_{i \leq k \leq j}$) are now replaced by a single leaf node K, placed in appropriate place in the majority ranking of $\mathcal{B}$ (in this case, ith position), which is well-defined, since the replaced children formed an interval. The PQ tree for $\boldsymbol{\sigma}|_K$, on the other hand, is exactly $\{T(B_k(\mathcal{B}, T))\}_{i \leq k \leq j}$, united by a single Q-node that is the root of the tree (except the leaf for each singleton $\{a\}$ is replaced with the leaf for a).

Now take any internal node $\mathcal{B} \subseteq \mathcal{K}$ of the tree T such that either $\mathcal{D} \subsetneq \mathcal{B}$ or $\mathcal{D} \cap \mathcal{B} = \emptyset$ (in words, $T(\mathcal{B})$ either strictly contains $\mathcal{D}$, or is not overlapping with it $\mathcal{D}$ at all). In both cases, there is (by the analysis above) a corresponding node $\mathcal{B}'$ in the tree T': if $\mathcal{D} \subseteq \mathcal{B}$, then $\mathcal{B}' = \mathcal{B} \setminus \mathcal{D} \cup \{K\}$, and if $\mathcal{D} \cap \mathcal{B} = \emptyset$ then $\mathcal{B}' = \mathcal{B}$. We would like to compare the children node that are enqueued by Algorithm 1 if (**case (a)**) it enqueues $\mathcal{B}$ when run on input $\boldsymbol{\sigma}^{\mathcal{K}}$ versus if (**case (b)**) it enqueues $\mathcal{B}'$ when run on input $\boldsymbol{\sigma}^{\mathcal{K}'}$. We consider each possible scenario:

1. If $\mathcal{D} \cap \mathcal{B} = \emptyset$. In this case, $\mathcal{B}' = \mathcal{B}$ so the algorithm proceeds the same way in both cases **(a)** and **(b)** , enqueing the same children regardless of whether $\mathcal{B}$ is a Q-node or a P-node.
2. If $\mathcal{D} \subsetneq \mathcal{B}$, and $\mathcal{B}$ has a child node $\mathcal{E}$ such that $\mathcal{D} \subseteq \mathcal{E}$. In words, $\mathcal{B}$ is either a non-immediate ancestor of the subtree(s) corresponding to $\mathcal{D}$ or the parent node of a single subtree $T(\mathcal{D})$. In this case, decomp($\mathcal{B}', T'$)= decomp($\mathcal{B}, T$) $\setminus \{\mathcal{E}\} \cup \{\mathcal{E}'\}$, where $\mathcal{E}' = \mathcal{E} \setminus \mathcal{D} \cup \{K\}$. Then, $\boldsymbol{\sigma}^{\text{decomp}(\mathcal{B},T)}$ and $\boldsymbol{\sigma}^{\text{decomp}(\mathcal{B}',T')}$ are isomorphic (with $\mathcal{E}$ relabeled as $\mathcal{E}'$). Hence, the algorithm proceeds the same way in both cases **(a)** and **(b)** , enqueing the same children regardless of whether $\mathcal{B}$ is a Q-node or a P-node, since f is neutral. In other words, any child node $\mathcal{F} \neq \mathcal{E}$ will be enqueued in case **(a)** iff it is enqueued in case **(b)** ; $\mathcal{E}$ will be enqueued in case **(a)** iff $\mathcal{E}'$ is enqueued in case **(b)** .
3. If $\mathcal{D} \subsetneq \mathcal{B}$ and no child node of $\mathcal{B}$ entirely contains $\mathcal{D}$. By Lemma 11, this implies that $\mathcal{B}$ is a Q-node and $\exists i, j : 0 \leq i < j \leq |\text{decomp}(\mathcal{B}, T)|$ such that $\mathcal{D} = \bigcup_{k:i \leq k \leq j} B_k(\mathcal{B}, T)$. In words,

[11] It is worth making a notational point here: when dealing with PQ trees of a profile $\boldsymbol{\sigma}$ over a candidates A, and each leaf node corresponded to a candidate in $a \in A$ and each internal node could be represented as a subset $B \subseteq A$. Since T (resp., T') is the PQ trees of a summary $\boldsymbol{\sigma}^{\mathcal{K}}$ (resp., $\boldsymbol{\sigma}^{\mathcal{K}'}$), each leaf node now corresponds to a clone set $K \in \mathcal{K}$ (resp., $K' \in \mathcal{K}'$) and each internal node can be represented as a subset $\mathcal{B} \subseteq \mathcal{K}$ (resp., $\mathcal{B}' \subseteq \mathcal{K}'$).

$\mathcal{D}$ corresponds to the leaves of multiple (specifically, $j-i+1$) subtrees ($\{T(B_k(\mathcal{B},T))\}_{i\le k\le j}$) whose roots ($\{B_k(\mathcal{B},T)\}_{i\le k\le j}$) are an interval of children nodes of $\mathcal{B}$. We cannot have $j-i+1 = |\text{decomp}(\mathcal{B},T)|$, since this would imply $\mathcal{B}=\mathcal{D}$, even though we assumed $\mathcal{D}$ is a strict subset of $\mathcal{B}$. Hence, $\mathcal{B}'$ (the node in T' corresponding to $\mathcal{B}$) is a Q-node with $|\text{decomp}(\mathcal{B},T)|-(j-i)$ children nodes,[12] with $\{T(B_k(\mathcal{B},T))\}_{i\le k\le j}$ replaced by a single leaf node K (respecting the rest of the order). Say $\ell = |\text{decomp}(\mathcal{B},T)|$ and $\ell' = |\text{decomp}(\mathcal{B}',T')| = |\text{decomp}(\mathcal{B},T)|-(j-i)$. By the clone set definition, we have that $\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(\mathcal{B},T),B_2(\mathcal{B},T)\}}$ and $\boldsymbol{\sigma}^{\mathcal{K}'}|_{\{B_1(\mathcal{B}',T'),B_2(\mathcal{B}',T')\}}$ are isomorphic, and since f is neutral, we have

$$B_i(\mathcal{B},T)\in f(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(\mathcal{B},T),B_2(\mathcal{B},T)\}}) \Leftrightarrow B_i(\mathcal{B}',T')\in f(\boldsymbol{\sigma}^{\mathcal{K}'}|_{\{B_1(\mathcal{B}',T'),B_2(\mathcal{B}',T')\}})$$

for $i\in\{1,2\}$. Then we consider the three possible cases separately:

3a. If $f(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(\mathcal{B},T),B_2(\mathcal{B},T)\}}) = \{B_1(\mathcal{B},T)\}$, then $B_1(\mathcal{B},T)$ gets enqueued in case **(a)** , and $B_1(\mathcal{B}',T')$ gets enqueued in case **(b)** . If $i>1$ (*i.e.*, $B_1(\mathcal{B},T)\cap\mathcal{D}=\emptyset$) then $B_1(\mathcal{B},T) = B_1(\mathcal{B}',T')$, so the same node gets enqueued in both cases. If $i=1$, then $B_1(\mathcal{B}',T')=\{K\}$, so $B_1(\mathcal{B},T)\subseteq\mathcal{D}$ gets enqueued in **(a)** and $\{K\}$ gets enqueued in **(b)** .

3b. If $f(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(\mathcal{B},T),B_2(\mathcal{B},T)\}}) = \{B_2(\mathcal{B},T)\}$, then $B_\ell(\mathcal{B},T)$ gets enqueued in case **(a)** and $B_{\ell'}(\mathcal{B}',T')$ gets enqueued in case **(b)** . If $j<\ell$ (*i.e.*, $B_\ell(\mathcal{B},T)\cap\mathcal{D}=\emptyset$) then $B_\ell(\mathcal{B},T) = B_{\ell'}(\mathcal{B}',T')$, so the same node gets enqueued in both cases. If $j=\ell$, then $B_{\ell'}(\mathcal{B}',T')=\{K\}$, so $B_\ell(\mathcal{B},T)\subseteq\mathcal{D}$ gets enqueued in **(a)** and $\{K\}$ gets enqueued in **(b)** .

3c. $f(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(\mathcal{B},T),B_2(\mathcal{B},T)\}}) = \{B_1(\mathcal{B},T),B_2(\mathcal{B},T)\}$, then all the children nodes get enqueued in both cases

Together, the cases above imply that starting from corresponding nodes $\mathcal{B}$ and $\mathcal{B}'$ in T and T' (respectively) that either contain $\mathcal{D}$ and $\{K\}$ (respectively) or do not overlap with them,

- Algorithm 1 enqueues any child node of $\mathcal{B}$ that either contains $\mathcal{D}$ or do not overlap with it in **(a)** if and only if it enqueues corresponding childnode of $\mathcal{B}'$ in **(b)**
- Algorithm 1 enqueues some subtree(s) corresponding to $\mathcal{D}$ in **(a)** if and only if it outputs $|K|$ as one of the winners in **(b)** .

Since Algorithm 1 run on both input $\boldsymbol{\sigma}^{\mathcal{K}}$ or input $\boldsymbol{\sigma}^{\mathcal{K}'}$ start at their root nodes (which indeed contain $\mathcal{D}$ and $\{K\}$, respectively), inductively applying this argument implies that for all $K'\in\mathcal{K}\setminus\mathcal{D}$, we have:

$$K'\in f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}}) \iff K'\in f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}'}) \tag{13}$$

$$\mathcal{D}\cap f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}})\neq\emptyset \iff K\in f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}'}) \tag{14}$$

What remains to be shown is if $\mathcal{D}\cap f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}})\neq\emptyset$, then $\mathcal{D}\cap f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}}) = \{\{a\}\}_{a\in f^{CC}(\boldsymbol{\sigma}|_K)}$. In words, we must show that if Algorithm 1 outputs any descendants of $\mathcal{D}$ in case **(a)** , then these decedents are the same as those that are output by the algorithm on input $\boldsymbol{\sigma}|_K$. Consider the three cases, assuming $\mathcal{D}\cap f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}})\neq\emptyset$:

- $\mathcal{D}$ corresponds to a single subtree $T(\mathcal{D})$. Then by case (2.) above, we have that $\mathcal{D}$ will be enqueued by Algorithm 1 when running on input $\boldsymbol{\sigma}^{\mathcal{K}}$ annd $\{K\}$ will be enqueued by the algorithm when run on input $\boldsymbol{\sigma}^{\mathcal{K}'}$. Since the PQ-tree for $\mathcal{K}|_K$ is identical to $T(\mathcal{D})$, the descendants of $\mathcal{D}$ that will be output by Algorithm 1 when running on input $\boldsymbol{\sigma}^{\mathcal{K}}$ (after dequeuing $\mathcal{D}$) are the same as the ones the algorithm would output on input $\boldsymbol{\sigma}|_K$.

[12] If $|\text{decomp}(\mathcal{B},T)|-(j-i)=2$, then $\mathcal{B}'$ is technically both a Q- and a P- node, which does not affect our analysis since Algorithm 1 treats these cases identically.

- $\mathcal{D}$ corresponds to an interval of children nodes ($\{B_k(\mathcal{B},T)\}_{i\le k\le j}$) under a Q-node ($\mathcal{B}$) in T and $f(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(\mathcal{B},T),B_2(\mathcal{B},T)\}=\{B_1(\mathcal{B},T),B_2(\mathcal{B},T)\}}$. Then by case (3c.) above, Algorithm 1 will enqueue all of these children nodes when running on input $\boldsymbol{\sigma}^{\mathcal{K}}$ and will enqueue $\{K\}$ when running on input $\boldsymbol{\sigma}^{\mathcal{K}'}$. The root of the PQ tree of $\boldsymbol{\sigma}|_K$ (say T_K) is a Q-node (denoted K) connecting these subtrees ($\{T(B_k(\mathcal{B},T))\}_{i\le k\le j}$). Since f is neutral, we have $f(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(K,T_K),B_2(K,T_K)\}}) = \{B_1(\mathcal{B},T_K),B_2(\mathcal{B},T_K)\}$ as, by definition of Q-nodes, any voter $i\in N$ will have $B_1(\mathcal{B},T)\succ_i B_2(\mathcal{B},T)$ if and only if $B_1(K,T_K)\succ_i B_2(K,T_K)$. Hence, once again all of these subtrees will be enqueued on the first step of Algorithm 1 when it is run on input $\boldsymbol{\sigma}|_K$. The rest of the algorithm will follow identically in both cases.
- $\mathcal{D}$ corresponds to an interval of children nodes ($\{B_k(\mathcal{B},T)\}_{i\le k\le j}$) under a Q-node ($\mathcal{B}$) in T with $f(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(\mathcal{B},T),B_2(\mathcal{B},T)\})=\{B_1(\mathcal{B},T)\}}$ (resp., $f(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(\mathcal{B},T),B_2(\mathcal{B},T)\})=\{B_2(\mathcal{B},T)\}}$). Then by case (3a.) (resp., (3b.)) above, Algorithm 1 will only enqueue $\mathcal{E}\equiv B_1(\mathcal{B},T)\subsetneq\mathcal{D}$ (resp., $\mathcal{E}\equiv B_j(\mathcal{B},T)\subsetneq\mathcal{D}$). Then, the root of the PQ tree of $\boldsymbol{\sigma}|_K$ (say T_k) is a Q-node (denoted K) with $B_1(K,T_K)$ (resp., $B_{j-i+1}(K,T_K)$) corresponding to $\mathcal{E}$. By neutrality of f, we have $f(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(K,T_K),B_2(K,T_K)\}}) = \{B_1(\mathcal{B},T_K)\}$ (resp., $f(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(K,T_K),B_2(K,T_K)\}}) = \{B_2(\mathcal{B},T_K)\}$). Therefore, the first step of Algorithm 1 when it is run on input $\boldsymbol{\sigma}|_K$ will pick the subtree corresponding to $\mathcal{E}$. The rest of the algorithm will follow identically in both cases.

Together, these cases show that the if any descendent of $\mathcal{D}$ will be output when Algorithm 1 is run on $\boldsymbol{\sigma}^{\mathcal{K}}$, then they are the same as those output when its run on $\boldsymbol{\sigma}|_K$. In other words, if $\mathcal{D}\cap f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}})\neq\emptyset$, then $\mathcal{D}\cap f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}}) = \{\{a\}\}_{a\in f^{CC}(\boldsymbol{\sigma}|_K)}$. Combined with (13) and (14), this gives us

$$\begin{aligned}\mathcal{D}\cap f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}})=\emptyset\Rightarrow\Pi_{f^{CC}}(\boldsymbol{\sigma},\mathcal{K}) &= \bigcup_{K'\in f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}})} f^{CC}(\boldsymbol{\sigma}|_{K'})\\ &= \bigcup_{K'\in f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}'})} f^{CC}(\boldsymbol{\sigma}|_{K'}) = \Pi_{f^{CC}}(\boldsymbol{\sigma},\mathcal{K}'),\text{ and}\end{aligned}$$

$$\begin{aligned}\mathcal{D}\cap f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}})\neq\emptyset\Rightarrow\Pi_{f^{CC}}(\boldsymbol{\sigma},\mathcal{K}) &= \left(\bigcup_{K'\in f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}})\setminus\mathcal{D}} f^{CC}(\boldsymbol{\sigma}|_{K'})\right)\cup\left(\bigcup_{K'\in f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}})\cap\mathcal{D}} f^{CC}(\boldsymbol{\sigma}|_{K'})\right)\\ &= \left(\bigcup_{K'\in f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}'})\setminus\{K\}} f^{CC}(\boldsymbol{\sigma}|_{K'})\right)\cup\left(\bigcup_{a\in f^{CC}(\boldsymbol{\sigma}|_K)} f^{CC}(\boldsymbol{\sigma}|_{\{a\}})\right)\\ &= \left(\bigcup_{K'\in f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}'})\setminus\{K\}} f^{CC}(\boldsymbol{\sigma}|_{K'})\right)\cup f^{CC}(\boldsymbol{\sigma}|_K)\\ &= \bigcup_{K'\in f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}'})} f^{CC}(\boldsymbol{\sigma}|_{K'}) = \Pi_{f^{CC}}(\boldsymbol{\sigma},\mathcal{K}').\end{aligned}$$

In both cases, we have $\Pi_{f^{CC}}(\boldsymbol{\sigma},\mathcal{K}) = \Pi_{f^{CC}}(\boldsymbol{\sigma},\mathcal{K}')$, completing the proof of the lemma. □

We now turn to proving f^{CC} satisfies CC for any neutral f. Now, given any neutral SCF f, profile $\boldsymbol{\sigma}$ over A, and decomposition $\mathcal{K}=\{K_1,K_2,\ldots,K_\ell\}$ with respect to $\boldsymbol{\sigma}$, define $\mathcal{K}_i=\{K_1,K_2,\ldots,K_i\}\cup\{\{a\}\}_{a\in A\setminus(\bigcup_{j\in[i]}K_j)}$ for each $i\in[\ell]\cup\{0\}$. In words, $\mathcal{K}_i$ is the decomposition with the first i clone sets in $\mathcal{K}$, and the remaining candidates are left as singletons. We have $\mathcal{K}_0=\{\{a\}\}_{a\in A}=\mathcal{K}_{triv}$ and $\mathcal{K}_\ell=\mathcal{K}$. Then

$$f^{CC}(\boldsymbol{\sigma}) = \Pi_{f^{CC}}(\boldsymbol{\sigma},\mathcal{K}_0) = \Pi_{f^{CC}}(\boldsymbol{\sigma},\mathcal{K}_1) = \ldots = \Pi_{f^{CC}}(\boldsymbol{\sigma},\mathcal{K}_{\ell-1}) = \Pi_{f^{CC}}(\boldsymbol{\sigma},\mathcal{K}_\ell) = \Pi_{f^{CC}}(\boldsymbol{\sigma},\mathcal{K}),$$

where first equality follows from Lemma 55 and subsequent inequalities follow from Lemma 56. Since $\mathcal{K}$ was arbitrarily chosen, this proves that f^{CC} satisfies CC.

Condition 3. Say f is a composition-consistent SCF. By Definition 7, f must be neutral. We will prove $f(\boldsymbol{\sigma}) = f^{CC}(\boldsymbol{\sigma})$ by inducting on the depth of the PQ-tree of $\boldsymbol{\sigma}$ (say T). As a base case, say T has depth one (*i.e.*, it is a single leaf node). This implies there is a single candidate in $\boldsymbol{\sigma}$, so both f and f^{CC} will return that candidate. Now, assume f and f^{CC} agree on all profiles with PQ-trees of depth 1,2,...,i-1, and say $\boldsymbol{\sigma}$ has a PQ-tree (say T) of depth i. Say A is the root node of T and consider two cases:

1. A is a P-node. Say $\mathcal{K}$ =decomp(K, T). By the recursive construction of Algorithm 1, we will have $f^{CC}(\boldsymbol{\sigma}) = \bigcup_{B \in f(\boldsymbol{\sigma}^{\mathcal{K}})} f^{CC}(\boldsymbol{\sigma}|_B)$. Since f is CC, we must also have $f(\boldsymbol{\sigma}) = \Pi_f(\boldsymbol{\sigma}, \mathcal{K}) = \bigcup_{B \in f(\boldsymbol{\sigma}^{\mathcal{K}})} f(\boldsymbol{\sigma}|_B)$. For each $B \in \mathcal{K}$, $\boldsymbol{\sigma}|_B$ can have a PQ-tree of depth at most $i-1$. Thus, by the inductive hypothesis we have $f^{CC}(\boldsymbol{\sigma}|_B) = f(\boldsymbol{\sigma}|_B)$, implying $f^{CC}(\boldsymbol{\sigma}) = f(\boldsymbol{\sigma})$, as desired.

2. A is a Q-node. Say $\mathcal{K}$ =decomp(K, T) and $\ell = |\mathcal{K}|$. For each $i, j \in [\ell]$, say $B_i = B_i(A, T)$ and $B_{i,j} = \cup_{[k \in [i,j]} B_k$. Consider three cases:

   (2a) $f(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1, B_2\}}) = \{B_1\}$. By Algorithm 1, we have $f^{CC}(\boldsymbol{\sigma}) = f^{CC}(\boldsymbol{\sigma}|_{B_1})$. Consider the decomposition $\mathcal{K}_1 = \{B_1, B_{2,\ell}\}$ (this is indeed a valid decomposition by the definition of a Q-node). Since f is CC and neutral, we must have $f(\boldsymbol{\sigma}) = \Pi_f(\boldsymbol{\sigma}, \mathcal{K}_1) = \bigcup_{B \in f(\boldsymbol{\sigma}^{\mathcal{K}_1})} f(\boldsymbol{\sigma}|_B) = f(\boldsymbol{\sigma}|_{B_1})$. Since $\boldsymbol{\sigma}|_{B_1}$ must have a PQ-tree of depth at most $i-1$, by the inductive hypothesis we must have $f^{CC}(\boldsymbol{\sigma}|_{B_1}) = f(\boldsymbol{\sigma}|_{B_1})$, implying $f^{CC}(\boldsymbol{\sigma}) = f(\boldsymbol{\sigma})$, as desired.

   (2b) $f(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1, B_2\}}) = \{B_2\}$. By Algorithm 1, we have $f^{CC}(\boldsymbol{\sigma}) = f^{CC}(\boldsymbol{\sigma}|_{B_\ell})$. Consider the decomposition $\mathcal{K}_2 = \{B_{1,\ell-1}, B_\ell\}$ (this is indeed a valid decomposition by the definition of a Q-node). Since f is CC and neutral, we must have $f(\boldsymbol{\sigma}) = \Pi_f(\boldsymbol{\sigma}, \mathcal{K}_2) = \bigcup_{B \in f(\boldsymbol{\sigma}^{\mathcal{K}_2})} f(\boldsymbol{\sigma}|_B) = f(\boldsymbol{\sigma}|_{B_\ell})$. Since $\boldsymbol{\sigma}|_{B_\ell}$ must have a PQ-tree of depth at most $i-1$, by the inductive hypothesis we must have $f^{CC}(\boldsymbol{\sigma}|_{B_\ell}) = f(\boldsymbol{\sigma}|_{B_\ell})$, implying $f^{CC}(\boldsymbol{\sigma}) = f(\boldsymbol{\sigma})$, as desired.

   (2c) $f(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1, B_2\}}) = \{B_1, B_2\}$. By Algorithm 1, we have $f^{CC}(\boldsymbol{\sigma}) = \bigcup_{i \in [\ell]} f^{CC}(\boldsymbol{\sigma}|_{B_i})$. For each $i \in [\ell - 1]$, define $\mathcal{K}_i = \{B_i, B_{i+1,\ell}\}$ as a decomposition of $\boldsymbol{\sigma}|_{B_{i,\ell}}$. By successively using the fact that f is CC and neutral, we get

$$\begin{aligned} f(\boldsymbol{\sigma}) = \Pi_f(\boldsymbol{\sigma}, \mathcal{K}_1) &= \bigcup_{B \in f(\boldsymbol{\sigma}_1^{\mathcal{K}})} f(\boldsymbol{\sigma}|_B) = f(\boldsymbol{\sigma}|_{B_1}) \cup f(\boldsymbol{\sigma}|_{B_{2,\ell}}) = f(\boldsymbol{\sigma}|_{B_1}) \cup \Pi_f(\boldsymbol{\sigma}|_{B_{2,\ell}}, \mathcal{K}_2) \\ &= f(\boldsymbol{\sigma}|_{B_1}) \cup f(\boldsymbol{\sigma}|_{B_2}) \cup f(\boldsymbol{\sigma}|_{B_{3,\ell}}) = f(\boldsymbol{\sigma}|_{B_1}) \cup f(\boldsymbol{\sigma}|_{B_2}) \cup \Pi_f(\boldsymbol{\sigma}|_{B_{3,\ell}}, \mathcal{K}_3) = \dots \\ &= \bigcup_{i \in [\ell]} f(\boldsymbol{\sigma}|_{B_i}). \end{aligned}$$

   For each $i \in [\ell]$, $\boldsymbol{\sigma}|_{B_i}$ must have a PQ-tree of depth at most $i-1$. Thus by the inductive hypothesis we must have $f^{CC}(\boldsymbol{\sigma}|_{B_i}) = f(\boldsymbol{\sigma}|_{B_i})$, implying $f^{CC}(\boldsymbol{\sigma}) = f(\boldsymbol{\sigma})$, as desired.

The inductive proof above shows that in all cases, we have $f(\boldsymbol{\sigma}) = f^{CC}(\boldsymbol{\sigma})$ for all $\boldsymbol{\sigma}$, as long as f is CC.

Condition 4. The statement that f being anonymous implies f^{CC} being anonymous follows from the fact that Algorithm 1 is robust to relabeling of voters. For each of the remaining properties, we will prove that it is preserved as a separate lemma. We start with Condorcet consistency. Recall that f is Condorcet-consistent if it returns $Sm(\boldsymbol{\sigma})$ whenever $|Sm(\boldsymbol{\sigma})| = 1$, where Sm is defined in Table 2. In words, if there is a candidate $a \in A$ that pairwise defeats every other candidate in $\boldsymbol{\sigma}$ (*i.e.*, a is the *Condorcet winner*), then we must have $f(\boldsymbol{\sigma}) = \{a\}$.

Lemma 57. *If (neutral) f is Condorcet-consistent, then f^{CC} is Condorcet-consistent.*

Proof. Consider running Algorithm 1 on input SCF f, which is Condorcet-consistent, and profile $\boldsymbol{\sigma}$, where $a \in A$ is the Condorcet winner.

Assume the algorithm dequeues node $B \subseteq A$ whose subtree contains a. If $|B| = 1$, then B is the leaf corresponding to $\{a\}$, and a is added to the winner list W. If $|B| > 1$, then say $\mathcal{K} = \text{decomp}(B, T)$. Since $|B|$ is an internal node, we have $|\mathcal{K}| > 1$ (all internal nodes in the PQ-tree has at least two children—see Appendix E.1 above). We would like to the show that *only* the child node that contains a (say $K_a \in \mathcal{K}$) will be enqueued by the algorithm. Given any $K' \in \mathcal{K} \setminus \{K_a\}$ and $b \in K'$, we have $M[a, b] = M^{\mathcal{K}}[K_a, K']$ by the clone definition (*i.e.*, the majority relationship between a and b is the same as the majority relationship between their clone sets). Since a pairwise defeats all $b \in A \setminus \{a\}$, this implies K_a pairwise defeats all $K' \in \mathcal{K} \setminus \{K_a\}$, *i.e.*, K_a is the Condorcet winner of $\boldsymbol{\sigma}^{\mathcal{K}}$. Then, there are two options:

1. If B is a P-node: since K_a is the Condorcet winner of $\boldsymbol{\sigma}^{\mathcal{K}}$ and f is Condorcet-consistent we have that $f(\boldsymbol{\sigma}^{\mathcal{K}}) = \{K_a\}$. Hence, only K_a is enqueued by the algorithm among the children nodes of K_a.
2. If B is a Q-node: since K_a is a Condorcet winner of $\boldsymbol{\sigma}^{\mathcal{K}}$, we must have $K_a = B_1(B, T)$. Moreover, since f is Condorcet-consistent, we must have $f(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(B,T), B_2(B,T)\}}) = \{B_1(B, T)\}$ (as $B_1(B, T) = K_a$ pairwise defeats $B_2(B, T)$). Hence, only K_a is enqueued by the algorithm among the children nodes of B.

This implies that starting from a node whose subtree contains a, Algorithm 1 will iteratively pick only the children node containing a, until arriving at a's leaf node and adding it to the winner list. Since the queue $\mathcal{Q}$ initially has only the root node (denoted A), whose subtree (T) indeed contains a, this implies only a will be added to W by the algorithm. Hence, $f^{CC}(\boldsymbol{\sigma}) = \{a\}$, *i.e.*, f^{CC} is Condorcet-consistent. □

Before proving the preservation of the stronger axiom of Smith consistency, which dictates $f(\boldsymbol{\sigma}) \subseteq Sm(\boldsymbol{\sigma})$ for *all* profiles $\boldsymbol{\sigma}$, we first prove a useful intermediary lemma.

Lemma 58. *Given any profile $\boldsymbol{\sigma}$ and clone set K with respect to $\boldsymbol{\sigma}$, it must be that K and $Sm(\boldsymbol{\sigma})$ cannot intersect nontrivially. That is, it must be that either $Sm(\boldsymbol{\sigma}) \subseteq K$, $K \subseteq Sm(\boldsymbol{\sigma})$, or $Sm(\boldsymbol{\sigma}) \cap K = \emptyset$.*

Proof. Suppose K and $Sm(\boldsymbol{\sigma})$ intersects nontrivially. Take any $a \in K \setminus Sm(\boldsymbol{\sigma})$, $b \in K \cap Sm(\boldsymbol{\sigma})$, and $c \in Sm(\boldsymbol{\sigma}) \setminus K$. We must have that c pairwise defeats a, since $a \notin Sm(\boldsymbol{\sigma})$ and $c \in Sm(\boldsymbol{\sigma})$. By the clone definition this implies c also pairwise defeats b. Thus each candidate in $Sm(\boldsymbol{\sigma}) \setminus K$ pairwise defeats any candidate out of it, and is strictly smaller than $Sm(\boldsymbol{\sigma})$. This contradicts the definition of $Sm(\boldsymbol{\sigma})$. □

Corollary 59. *If $\mathcal{K}$ is a clone decomposition with respect to $\boldsymbol{\sigma}$, either there exists $K \in \mathcal{K}$ such that $Sm(\boldsymbol{\sigma}) \subseteq K$, or there exists $\mathcal{K}' \subseteq \mathcal{K}$ such that $Sm(\boldsymbol{\sigma}) = \bigsqcup_{K \in \mathcal{K}'} K$.*

Lemma 60. *If (neutral) f is Smith-consistent, then f^{CC} is Smith-consistent.*

Proof. We first show that when run on SCF f (which is Smith-consistent) and profile $\boldsymbol{\sigma}$ (with PQ tree T), for any node $B \subseteq A$ whose subtree contains the entirety of $Sm(\boldsymbol{\sigma})$, Algorithm 1 either only enqueues a single child node that is also a superset of $Sm(\boldsymbol{\sigma})$, or only enqueues (possibly multiple) children nodes that are subsets of $Sm(\boldsymbol{\sigma})$. By Corollary 59, when the algorithm is at node B that is a superset of $Sm(\boldsymbol{\sigma})$, the corresponding decomposition $\mathcal{K} = \text{decomp}(B, T)$ will satisfy one of two cases:

1. One child node ($K_S \in \mathcal{K}$) will contain all candidates in the Smith set. By the clone definition, this implies K_S pairwise defeats every other clone set in $\boldsymbol{\sigma}^{\mathcal{K}}$. This means that $\{K_S\}$ is the Smith set of $\boldsymbol{\sigma}^{\mathcal{K}}$. Consider the cases for B:

   (1a) If B is P-node, then because f is Smith-consistent, we have that $f(\boldsymbol{\sigma}^{\mathcal{K}}) \subseteq Sm(\boldsymbol{\sigma}^{\mathcal{K}}) = \{K_S\}$. Hence, again, only K_S gets enqueued.

   (1b) If B is a Q-node, then we must have $B_1(B,T) = K_S$, which is the only child that gets enqueued. Moreover, since f is Smith-consistent, we must have $f(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(B,T),B_2(B,T)\}}) = \{B_1(B,T)\}$ (as $B_1(B,T) = K_a$ pairwise defeats $B_2(B,T)$, so $Sm(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(B,T),B_2(B,T)\}}) = \{B_1(B,T)\}$). Hence, only K_S is enqueued by the algorithm among the children nodes of B.

2. There exists some $\mathcal{K}' \subseteq \mathcal{K}$ such that $Sm(\boldsymbol{\sigma}) = \bigsqcup_{K \in \mathcal{K}'} K$. In this case, $\mathcal{K}'$ is the Smith set of $\boldsymbol{\sigma}^{\mathcal{K}}$ (since $Sm(\boldsymbol{\sigma}^{\mathcal{K}}) \subseteq \mathcal{K}'$ by the clone definition, and $\mathcal{K}' \subseteq Sm(\boldsymbol{\sigma}^{\mathcal{K}})$ by the minimality of $Sm(\boldsymbol{\sigma})$). Consider the cases for B:

   (2a) B is a P-node. Since f is Smith-consistent, it must be that $f(\boldsymbol{\sigma}^{\mathcal{K}}) \subseteq Sm(\boldsymbol{\sigma}^{\mathcal{K}}) = \mathcal{K}'$. Therefore, only child nodes that are subsets of $Sm(\boldsymbol{\sigma})$ will be enqueued.

   (2b) B is a Q-node and $\mathcal{K} \setminus \mathcal{K}' = \emptyset$. Then we have $B = Sm(\boldsymbol{\sigma}^{\mathcal{K}})$, so any children of B that is enqueued is a subset of $Sm(\boldsymbol{\sigma}^{\mathcal{K}})$ by definition.

   (2c) B is a Q-node and $\mathcal{K} \setminus \mathcal{K}' \neq \emptyset$. Since any $K \in \mathcal{K}'$ must pairwise any K', we must have $\mathcal{K}' = \{B_i(B,T)\}_{i=1}^{j}$ for some $j < |\mathcal{K}|$. Further, we must $j = 1$, as otherwise $B_1(B,T)$ also pairwise defeats the remaining members of $\mathcal{K}' = Sm(\boldsymbol{\sigma}^{\mathcal{K}})$, which contradicts the minimality of Sm. Therefore this case is identical to that of (1b), and only the Smith set gets enqueued.

Starting from a node B that is a superset of $Sm(\boldsymbol{\sigma})$, there can only $|B \setminus Sm(\boldsymbol{\sigma})|$ number of subsequent nodes that falls into case (1) above, since each time this happens at least *some* non-Smith candidates are dropped by the algorithm. Hence, starting from a node B that is a superset of $Sm(\boldsymbol{\sigma})$, the algorithm will eventually come to a node that fulfills case (2) above, in which case only the child nodes that are entirely subsets of $Sm(\boldsymbol{\sigma})$ are enqueued, after which it is impossible for any $B \setminus Sm(\boldsymbol{\sigma})$ to win. Since the root node of the tree (A), where the algorithm starts, is by definition a superset of $Sm(\boldsymbol{\sigma})$, this implies that $f^{CC}(\boldsymbol{\sigma}) \subseteq Sm(\boldsymbol{\sigma})$, *i.e.*, f^{CC} satisfies Smith-consistency. □

Recall that unlike the other axioms, we have so far only defined decisiveness on a specific profile $\boldsymbol{\sigma}$, *i.e.* $|f(\boldsymbol{\sigma})| = 1$ (see Section 2). Having fixed the voters N and candidates A, we say f is (overall) decisive if it is decisive on all $\boldsymbol{\sigma} \in \mathcal{L}(A)^n$.

Lemma 61. *If (neutral) f is decisive, then f^{CC} is decisive.*

Proof. When run on f and any profile $\boldsymbol{\sigma}$ (with PQ-tree T), for each node B that is dequeued, Algorithm 1 will always enqueue a single child node of B: if B is a P-node, this is $f(\boldsymbol{\sigma}^{\mathcal{K}})$ (where $\mathcal{K}$ =decomp(B,T)), which has cardinality 1 since f is decisive; if B is a Q-node, this is $B_1(B,T)$ or $B_{|\mathcal{K}|}(B,T)$ (we cannot have $f(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(B,T),B_2(B,T)\}} = \{B_1(B,T), B_2(B,T)\}$ since f is decisive). This implies that Algorithm 1 will start from the root node of T and go down one child node at a time, until reaching a leaf node, which will be the single winner added to W. Hence, $|f^{CC}(\boldsymbol{\sigma})| = 1$ for all $\boldsymbol{\sigma}$, *i.e.* f^{CC} is decisive. □

We now move to the clone-aware axioms, formally defined in the preceding section.

Lemma 62. *If (neutral) f satisfies monotonicity^{ca} (Def. 13), then f^{CC} satisfies monotonicity^{ca}.*

Proof. Fix a profile $\boldsymbol{\sigma}$, a candidate $a \in f^{CC}(\boldsymbol{\sigma})$, and a second profile $\boldsymbol{\sigma}'$ with (1) $\mathcal{C}(\boldsymbol{\sigma}) = \mathcal{C}(\boldsymbol{\sigma}')$ and (2) for all $i \in N$ and $b, c \in A \setminus \{a\}$, we have $a \succ_{\sigma_i} b \Rightarrow a \succ_{\sigma_i'} b$ and $b \succ_{\sigma_i} c \Rightarrow b \succ_{\sigma_i'} c$. We would like to show that $a \in f^{CC}(\boldsymbol{\sigma}')$. Since, $\mathcal{C}(\boldsymbol{\sigma}) = \mathcal{C}(\boldsymbol{\sigma}')$ the node structure of the PQ-trees of the two profiles (say T and T', respectively) are identical, but the number of each vote in $\boldsymbol{\sigma}^{\mathcal{K}}$ and $\boldsymbol{\sigma}'^{\mathcal{K}}$ might be different for a given $\mathcal{K}$. Hence, we only need to show that at each node $B \subseteq A$ that contains a, the child node containing a (and possibly others) will be enqueued. Fix an internal node B cotaining a in the PQ-tree, and say $\mathcal{K} = \text{decomp}(B, T)$=decomp$(B, T')$ and K_a is the clone set in $\mathcal{K}$ containing a (*i.e.*, $a \in K_a \in \mathcal{K}$). Consider two options:

1. B is a P-node. Since $a \in f^{CC}(\boldsymbol{\sigma})$, we must have $K_a \in f(\boldsymbol{\sigma}^{\mathcal{K}})$. Furthermore, for any two clone sets $K_b, K_c \in \mathcal{K} \setminus \{K_a\}$, it must be that $K_a \succ_{\sigma_i^{\mathcal{K}}} K_b \implies K_a \succ_{\sigma_i^{\mathcal{K}'}} K_b$ and $K_b \succ_{\sigma_i^{\mathcal{K}}} K_c \implies K_b \succ_{\sigma_i^{\mathcal{K}'}} K_c$ for all $i \in N$ by the clone set definition, since the only difference between $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma}'$ is a moving up in some rankings. As f satisfies clone-aware monotonicity, $K_a \in f(\boldsymbol{\sigma}^{\mathcal{K}}) \implies K_a \in f(\boldsymbol{\sigma}'^{\mathcal{K}})$, implying K_a is enqueued in both cases.

2. B is a Q-node. This implies everyone in $\boldsymbol{\sigma}^{\mathcal{K}}$ has either ranked $B_1(B,T) \succ B_2(B,T) \succ \ldots \succ B_{|\mathcal{K}|}(B,T)$ or $B_{|\mathcal{K}|}(B,T) \succ B_{|\mathcal{K}|-1}(B,T) \succ \ldots \succ B_1(B,T)$. Say $K_a = B_k(B,T)$ for some $k \in [|\mathcal{K}|]$. Consider two cases:

   (2a) $|\mathcal{K}| > 2$. For any $i \in N$, we will show that $\sigma_i^{\mathcal{K}} = \sigma_i'^{\mathcal{K}}$. By construction, the order in which all $B_i(B,T)$ for $i \in [|\mathcal{K}|] \setminus \{k\}$ are the same in the two rankings, as only K_a can move up. Assume for the sake of contradiction $\sigma_i'^{\mathcal{K}}$ ranks K_a in the jth position for some $j < k$. If $j > 1$, this implies $\{B_{j-1}(B,T), B_j(B,T)\}$ is a clone set in $\boldsymbol{\sigma}^{\mathcal{K}}$ (by definition of a Q-node) but not a clone set in $\boldsymbol{\sigma}'^{\mathcal{K}}$ (as they are interrupted by K_a in $\sigma_i'^{\mathcal{K}}$), and therefore $B_{j-1}(B,T) \cup B_j(B,T) \in \mathcal{C}(\boldsymbol{\sigma}) \setminus \mathcal{C}(\boldsymbol{\sigma}')$, which contradicts the assumption that $\mathcal{C}(\boldsymbol{\sigma}) = \mathcal{C}(\boldsymbol{\sigma}')$. Similarly, if $k < |\mathcal{K}|$, then $B_k(B,T) \cup B_{k+1}(B,T) \in \mathcal{C}(\boldsymbol{\sigma}) \setminus \mathcal{C}(\boldsymbol{\sigma}')$, once again leading to a contradiction. Lastly, if $j = 1$ and $k = |\mathcal{K}|$, we have $B_{k-1}(B,T) \cup B_k(B,T) \in \mathcal{C}(\boldsymbol{\sigma}) \setminus \mathcal{C}(\boldsymbol{\sigma}')$, as they are now interrupted by $B_1(B,T)$ (we have $k - 1 > 1$ since $k > 2$). In all cases, assuming $j < k$ leads to a contradiction. Hence, $\sigma_i'^{\mathcal{K}}$ ranks K_a in the kth position, implying $\sigma_i^{\mathcal{K}} = \sigma_i'^{\mathcal{K}}$ and therefore $\boldsymbol{\sigma}^{\mathcal{K}} = \boldsymbol{\sigma}'^{\mathcal{K}}$. Thus, Algorithm 1 enqueues the same children nodes (and by assumption K_a) in both cases.

   (2b) $|\mathcal{K}| = 2$, in this case, B is a P-node and a Q-node at the same time (Q-nodes with two children are treated identically to P-nodes by Algorithm 1), therefore we have this case is identical to case (1) above.

Therefore, at every step in the PQ-tree, since the clone set that contains a is enqueued when running Algorithm 1 on $\boldsymbol{\sigma}$, it will also be enqueued when running Algorithm 1 on $\boldsymbol{\sigma}'$. Hence, $a \in f^{CC}(\boldsymbol{\sigma}')$, as desired. □

Lemma 63. *If (neutral) f satisfies ISDA^{ca} (Def. 54), then f^{CC} satisfies ISDA^{ca}.*

Proof. Assume f satisfies ISDA^{ca}, and take any profile $\boldsymbol{\sigma} \in \mathcal{L}(A)^n$ over candidates A and any candidate $a \in A$ such that $a \notin Sm(\boldsymbol{\sigma})$ and $\mathcal{C}(\boldsymbol{\sigma} \setminus \{a\}) = \mathcal{C}(\boldsymbol{\sigma}) - \{a\}$. Denote $\boldsymbol{\sigma}' = \boldsymbol{\sigma} \setminus \{a\}$. Say T is the PQ-tree of $\boldsymbol{\sigma}$ and $\mathcal{K} = \{K_1, K_2, \ldots, K_\ell\}$ =decomp(A, T) are children nodes of the root node of T. WLOG, say $a \in K_\ell$. Since $\mathcal{C}(\boldsymbol{\sigma}') = \mathcal{C}(\boldsymbol{\sigma}) - \{a\}$, we have that $\mathcal{K}' = \{K_1, ..., K_\ell \setminus \{a\}\}$ is a clone decomposition with respect to $\boldsymbol{\sigma}'$. We will argue $f^{CC}(\boldsymbol{\sigma}) = f^{CC}(\boldsymbol{\sigma} \setminus \{a\})$ by induction on the depth of the PQ-tree of $\boldsymbol{\sigma}$ (say T).

Base case: Say T has depth 2 (depth 1 is impossible, since $a \notin Sm(\boldsymbol{\sigma})$ implies $\boldsymbol{\sigma}$ is over at least 2 candidates). In this case, $|K_i| = 1$ for each $i \in [\ell]$. If the root is a P-node, this implies $\boldsymbol{\sigma}$ has no

non-trivial clone sets. Since $\mathcal{C}(\boldsymbol{\sigma}') = \mathcal{C}(\boldsymbol{\sigma}) - \{a\}$, this implies $\boldsymbol{\sigma}'$ also has no non-trivial clone sets. Then:

$$f^{CC}(\boldsymbol{\sigma}) = f(\boldsymbol{\sigma}) = f(\boldsymbol{\sigma}') = f^{CC}(\boldsymbol{\sigma}')$$

where the first and last inequality follows from Condition 1 of Theorem 3 proven above, and the second inequality follows from the assumption that f satisfies ISDA$^{\text{ca}}$. If the root of T is a Q-node (with majority ranking σ) on the other hand, it must be an untied Q-node (*i.e.*, strictly more voters rank σ than its reverrse), otherwise we would have had $Sm(\boldsymbol{\sigma}) = A$, which contradicts the assumption that $a \notin Sm(\boldsymbol{\sigma})$. Since f satisfies ISDA$^{\text{ca}}$, we must have $f(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(B,T),B_2(B,T)\}} = \{B_1(B,T)\}$, otherwise removing $B_2(B,T)\}$, which is not in the Smith set of $\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(B,T),B_2(B,T)\}}$ would change the election result. Therefore, $f^{CC}(\boldsymbol{\sigma}) = B_1(A,T)$. Since a is not in the Smith set, this implies $K_\ell = \{a\} \neq B_1(B,T)$. Since the removal of a does not change the clone structure, the PQ tree of $\boldsymbol{\sigma}'$ (say T') is either a single leaf node corresponding to $B_1(B,T)$ (if $\ell = 2$) or is also a single Q-node with $\ell - 1$ children nodes that are all leaves and margin matrix $\sigma \setminus \{a\}$ (if $\ell > 2$). Since a did not come first in σ, this implies $f^{CC}(\boldsymbol{\sigma}') = B_1(A \setminus \{a\}, T') = B_1(A,T) = f^{CC}(\boldsymbol{\sigma})$. This finishes the base case.

Inductive: Assume that $f^{CC}(\boldsymbol{\sigma}) = f^{CC}(\boldsymbol{\sigma}')$ if the depth of T is $1, 2, \ldots k-1$. Fix a profile $\boldsymbol{\sigma}$ such that T has depth k. Since f^{CC} satisfies CC by Condition 2 proven above, we have $f^{CC}(\boldsymbol{\sigma}) = \Pi_{f^{CC}}(\boldsymbol{\sigma}, \mathcal{K})$ and $f^{CC}(\boldsymbol{\sigma}') = \Pi_{f^{CC}}(\boldsymbol{\sigma}', \mathcal{K}')$. Hence, it is sufficent to prove that $\Pi_{f^{CC}}(\boldsymbol{\sigma}, \mathcal{K}) = \Pi_{f^{CC}}(\boldsymbol{\sigma}', \mathcal{K}')$. Consider two cases:

1. If $|K_\ell| = 1$, then $K_\ell = \{a\}$ is a Smith-dominated candidate within $\boldsymbol{\sigma}^{\mathcal{K}}$. Moreover, $\mathcal{K}' = \{K_1, K_2, \ldots, K_{\ell-1}\}$. Since $\mathcal{K}$ correspond to the children of the root node of T, the PQ-tree of $\boldsymbol{\sigma}^{\mathcal{K}}$ (say $T^{\mathcal{K}}$) is either a single P-node or a Q-node. Since $\boldsymbol{\sigma}'^{\mathcal{K}'} = \boldsymbol{\sigma}^{\mathcal{K}} \setminus \{K_\ell\}$, it follows by the base case above that $f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}}) = f^{CC}(\boldsymbol{\sigma}'^{\mathcal{K}'})$. Then we have:

$$\Pi_{f^{CC}}(\boldsymbol{\sigma}, \mathcal{K}) = \bigcup_{K \in f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}})} f(\boldsymbol{\sigma}|_K) = \bigcup_{K \in f^{CC}(\boldsymbol{\sigma}'^{\mathcal{K}'})} f(\boldsymbol{\sigma}|_K) = \Pi_{f^{CC}}(\boldsymbol{\sigma}', \mathcal{K}')$$

   and we are done.

2. If $|K_\ell| > 1$: Then $\boldsymbol{\sigma}^{\mathcal{K}}$ and $\boldsymbol{\sigma}'^{\mathcal{K}'}$ are isomorphic, where the meta-candidate K_ℓ in $\boldsymbol{\sigma}^{\mathcal{K}}$ is replaced with the meta-candidate $K'_\ell = K_\ell \setminus \{a\}$ in $\boldsymbol{\sigma}'^{\mathcal{K}'}$. Consider two options:

   (2a) $K_\ell \notin f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}})$. Then, by neutrality, we have $f^{CC}(\boldsymbol{\sigma}'^{\mathcal{K}'}) = f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}})$, and we have

$$\Pi_{f^{CC}}(\boldsymbol{\sigma}, \mathcal{K}) = \bigcup_{K \in f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}})} f(\boldsymbol{\sigma}|_K) = \bigcup_{K \in f^{CC}(\boldsymbol{\sigma}'^{\mathcal{K}'})} f(\boldsymbol{\sigma}|_K) = \Pi_{f^{CC}}(\boldsymbol{\sigma}', \mathcal{K}').$$

   (2b) $K_\ell \in f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}})$. Then, by neutrality, we have $f^{CC}(\boldsymbol{\sigma}'^{\mathcal{K}'}) = f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}}) \setminus \{K_\ell\} \cup \{K'_\ell\}$. We argue that $Sm(\boldsymbol{\sigma}) \cap K_\ell \neq \emptyset$. Assume for the sake of contradiction that $Sm(\boldsymbol{\sigma}) \cap K_\ell = \emptyset$. Then $K_\ell \notin Sm(\boldsymbol{\sigma}^{\mathcal{K}})$. If the root of $T^{\mathcal{K}}$ is a Q-node, this contradicts $K_\ell \in f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}})$, since it cannot not be $B_1(\mathcal{K}, T^{\mathcal{K}})$, which is the only enqueued child node since f is ISDA$^{\text{ca}}$. If the root of $T^{\mathcal{K}}$ is a P-node, then by Condition 1 of Theorem 3, we have $f(\boldsymbol{\sigma}^{\mathcal{K}}) = f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}})$, so $K_\ell \in f(\boldsymbol{\sigma}^{\mathcal{K}})$ violates the assumption that f satisfies ISDA$^{\text{ca}}$, since by definition removing K_ℓ (which is not in $Sm(\boldsymbol{\sigma}^{\mathcal{K}})$) will change the outcome. Therefore, we must have $Sm(\boldsymbol{\sigma}) \cap K_\ell \neq \emptyset$. By Lemma 58, this implies $Sm(\boldsymbol{\sigma}) \subset K_\ell$, since $a \in K_\ell \setminus Sm(\boldsymbol{\sigma})$. Then, $a \notin Sm(\boldsymbol{\sigma}|_{K_\ell})$. Moreover, the PQ-tree of $\boldsymbol{\sigma}|_{K_\ell}$ has depth at most $k-1$. By the inductive hypothesis, this

implies that $f^{CC}(\boldsymbol{\sigma}|_{K_\ell}) = f^{CC}(\boldsymbol{\sigma}|_{K_\ell} \setminus \{a\}) = f^{CC}(\boldsymbol{\sigma}|_{K'_\ell})$. Therefore

$$\begin{aligned}\Pi_{f^{CC}}(\boldsymbol{\sigma}, \mathcal{K}) &= \left(\bigcup_{K \in f^{CC}(\boldsymbol{\sigma}^{\mathcal{K}}) \setminus \{K_\ell\}} f(\boldsymbol{\sigma}|_K) \right) \cup f^{CC}(\boldsymbol{\sigma}|_{K_\ell}) \\ &= \left(\bigcup_{K \in f^{CC}(\boldsymbol{\sigma}'^{\mathcal{K}'}) \setminus \{K'_\ell\}} f(\boldsymbol{\sigma}|_K) \right) \cup f^{CC}(\boldsymbol{\sigma}|_{K'_\ell}) \\ &= \bigcup_{K \in f^{CC}(\boldsymbol{\sigma}'^{\mathcal{K}'})} f(\boldsymbol{\sigma}|_K) = \Pi_{f^{CC}}(\boldsymbol{\sigma}', \mathcal{K}').\end{aligned}$$

In each case, we have shown that $\Pi_{f^{CC}}(\boldsymbol{\sigma}, \mathcal{K}) = \Pi_{f^{CC}}(\boldsymbol{\sigma}', \mathcal{K}')$, which, by Condition 2, implies $f^{CC}(\boldsymbol{\sigma}) = f^{CC}(\boldsymbol{\sigma}')$, thus completing the inductive case. □

Lemma 64. *If (neutral) f satisfies participation^{ca} (Def. 53), then f^{CC} satisfies participation^{ca}.*

Proof. Fix any profile $\boldsymbol{\sigma} \in \mathcal{L}(A)^n$ and any ranking $\sigma_{n+1} \in \mathcal{L}(A)$ such that $\mathcal{C}(\boldsymbol{\sigma}) = \mathcal{C}(\boldsymbol{\sigma} + \sigma_{n+1})$, implying that the PQ tree of both (say T and T', respectively) have the same structure. We denote $\boldsymbol{\sigma}' = \boldsymbol{\sigma} + \sigma_{n+1}$. Fix a node B in the PQ-tree that was dequeued by Algorithm 1 at some point when run on input $\boldsymbol{\sigma}$. Say $\mathcal{K} = \text{decomp}(B, T)$=decomp$(B, T')$ and that $\mathcal{K}^* \subseteq \mathcal{K}$ are the child nodes that were enqueued by the algorithm. We will show that if Algorithm 1 on input $\boldsymbol{\sigma}'$ ever dequeues B, then it will either enqueue $\max_{n+1}(\mathcal{K}^*)$ or a child node preferred by $\boldsymbol{\sigma}^{\mathcal{K}}_{n+1}$. Consider two cases:

1. B is a P-node. In that case, $\mathcal{K}^* = f(\boldsymbol{\sigma}^{\mathcal{K}})$ by construction of Algorithm 1. Similarly, if dequeued when run on input $\boldsymbol{\sigma}'$, Algorithm 1 will enqueue $f(\boldsymbol{\sigma}'^{\mathcal{K}})$. Since f satisfies clone-aware participation, we must have $\max_{n+1}(f(\boldsymbol{\sigma}'^{\mathcal{K}})) \succeq_{n+1} \max_{n+1}(f(\boldsymbol{\sigma}^{\mathcal{K}}))$, so the algorithm does indeed enqueue $\max_{n+1}(\mathcal{K}^*)$ or a child node preferred by $\boldsymbol{\sigma}^{\mathcal{K}}_{n+1}$.

2. B is a Q-node, with majority ranking σ^* over $\mathcal{K}$. Moreover, since $\mathcal{C}(\boldsymbol{\sigma}) = \mathcal{C}(\boldsymbol{\sigma}')$, $\sigma^{\mathcal{K}}_{n+1}$ must either be σ^* or its reverse. Consider three subcases:

   (2a) $f(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(B,T), B_2(B,T)\}}) = \{B_1(B,T)\}$. If $\sigma^{\mathcal{K}}_{n+1} = \sigma^*$, then $\boldsymbol{\sigma}'^{\mathcal{K}}|_{\{B_1(B,T'), B_2(B,T')\}}$ is simply $\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(B,T), B_2(B,T)\}}$ with an additional $(B_1(B,T) \succ B_2(B,T))$ vote. Since f satisfies participation^{ca}, we must have $f(\boldsymbol{\sigma}'^{\mathcal{K}}|_{\{B_1(B,T'), B_2(B,T')\}}) = \{B_1(B,T')\} = \{B_1(B,T)\}$. Thus $B_1(B,T)$ get enqueued on input $\boldsymbol{\sigma}'$, which is the top ranked candidate in $\sigma^{\mathcal{K}}_{n+1}$. If $\sigma^{\mathcal{K}}_{n+1}$ is the reverse of σ^*, on the other hand, the child node enqueued at B on input $\boldsymbol{\sigma}$ $(B_1(B,T))$ is the bottom ranked candidate in $\sigma^{\mathcal{K}}_{n+1}$, so it cannot possibly be ranked above the child node enqueued at B on input $\boldsymbol{\sigma}'$.

   (2b) $f(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(B,T), B_2(B,T)\}}) = \{B_2(B,T)\}$. If $\sigma^{\mathcal{K}}_{n+1} = \sigma^*$, the child node enqueued at B on input $\boldsymbol{\sigma}$ $(B_{|\mathcal{K}|}(B,T))$ is the bottom ranked candidate in $\sigma^{\mathcal{K}}_{n+1}$, so it cannot possibly be ranked above the child node enqueued at B on input $\boldsymbol{\sigma}'$. If $\sigma^{\mathcal{K}}_{n+1}$ is the reverse of σ^*, on the other hand, $\boldsymbol{\sigma}'^{\mathcal{K}}|_{\{B_1(B,T'), B_2(B,T')\}}$ is simply $\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(B,T), B_2(B,T)\}}$ with an additional $(B_2(B,T) \succ B_1(B,T))$ vote. By assumption f satisfies participation^{ca}; thus, we must have $f(\boldsymbol{\sigma}'^{\mathcal{K}}|_{\{B_1(B,T'), B_2(B,T')\}}) = \{B_2(B,T')\} = \{B_2(B,T)\}$. Thus $B_{|\mathcal{K}|}(B,T)$ get on input $\boldsymbol{\sigma}'$, which is the top ranked candidate in $\sigma^{\mathcal{K}}_{n+1}$.

   (2c) $f(\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(B,T), B_2(B,T)\}}) = \{B_1(B,T), B_2(B,T)\}$. If $\sigma^{\mathcal{K}}_{n+1} = \sigma^*$, $\boldsymbol{\sigma}'^{\mathcal{K}}|_{\{B_1(B,T'), B_2(B,T')\}}$ is simply $\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(B,T), B_2(B,T)\}}$ with an additional $(B_1(B,T) \succ B_2(B,T))$ vote. Since f satisfies participation^{ca}, we must have $B_1(B,T') \in f(\boldsymbol{\sigma}'^{\mathcal{K}}|_{\{B_1(B,T'), B_2(B,T')\}})$. Thus $B_1(B,T)$ is one of the child nodes that get enqueued on input $\boldsymbol{\sigma}'$, which is the top ranked candidate in $\sigma^{\mathcal{K}}_{n+1}$. If $\sigma^{\mathcal{K}}_{n+1}$ is the reverse of σ^*, on the other hand, $\boldsymbol{\sigma}'^{\mathcal{K}}|_{\{B_1(B,T'), B_2(B,T')\}}$

is simply $\boldsymbol{\sigma}^{\mathcal{K}}|_{\{B_1(B,T),B_2(B,T)\}}$ with an additional $(B_2(B,T) \succ B_1(B,T))$ vote. Since f satisfies participation$^{\text{ca}}$, we must have $B_2(B,T) \in f(\boldsymbol{\sigma}'^{\mathcal{K}}|_{\{B_1(B,T'),B_2(B,T')\}})$. Thus $B_{|\mathcal{K}|}(B,T)$ is one of the child nodes that get enqueued on input $\boldsymbol{\sigma}'$, which is the top ranked candidate in $\sigma^{\mathcal{K}}_{n+1}$.

In each case, we see that if Algorithm 1 is considering B when run on $\boldsymbol{\sigma}'$, it will either enqueue $\max_{n+1}(\mathcal{K}^*)$ or a child node strictly preferred by $\boldsymbol{\sigma}^{\mathcal{K}}_{n+1}$.

Say B is the root node (A), which is indeed dequeued by the algorithm when run on either input. If a $K \succ \max_{n+1}(\mathcal{K}^*)$ is enqueued (*i.e.*, a strictly preferred child node) when run on input σ', then we are done, since this implies $f^{CC}(\boldsymbol{\sigma}') \cap K \neq \emptyset$ and each element of K is preferred to all elements in $f^{CC}(\boldsymbol{\sigma})$ by σ_{n+1}. Otherwise, $\max_{n+1}(\mathcal{K}^*)$ must have been enqueued, and we can apply the same argument to that node, since it will be considered by the algorithm on both inputs. We repeat following the $\max_{n+1}(\mathcal{K}^*)$ on each step until we reach a strictly preferred child node that is enqueued, or we reach a leaf node, in which case $\max_{n+1}(f^{CC}(\boldsymbol{\sigma})) = \max_{n+1}(f^{CC}(\boldsymbol{\sigma}'))$. In either case, we have $\max_{n+1}(f^{CC}(\boldsymbol{\sigma}')) \succeq_{n+1} \max_{n+1}(f^{CC}(\boldsymbol{\sigma}'))$, proving f^{CC} satisfies participation$^{\text{ca}}$. □

Together Lemmata 57 and 60 to 64 prove Condition 4 of Theorem 3.

Condition 5. We analyze the running time of Algorithm 1: CC transformation for SCF f. First, we construct the PQ-tree $T = PQ(\boldsymbol{\sigma})$ for σ. By Lemma 11 (shown by Cornaz et al. [16]), this requires $O(nm^3)$ time. Next, whenever Algorithm 1 encounters a node B that is of type P-, it runs f on $\boldsymbol{\sigma}^{\mathcal{K}}$, where $\mathcal{K} = \text{decomp}(B,T)$. By definition of $\delta(T)$, we can upper bound the runtime of running $\boldsymbol{\sigma}^{\mathcal{K}}$ for each node B of type P- by $g(n,\delta(T))$. Hence, overall, this requires at most $|\mathcal{P}| \cdot g(n,\delta(T))$ runtime, where recall that $\mathcal{P}$ denotes the set of P-nodes in PQ-tree T. On the other hand, whenever Algorithm 1 encounters a node B that is of type Q-, it only runs f on the first two child nodes; *i.e.*, it runs f with at most two candidates. Again by definition of function g, each encounter of a Q-node in Algorithm 1 thus adds a running time of at most $g(n,2)$. Overall, this requires $|\mathcal{Q}| \cdot g(n,2)$ runtime, where $\mathcal{Q}$ denotes the set of Q-nodes in PQ-tree T.

Hence, the total runtime of Algorithm 1 is $O(nm^3)+|\mathcal{P}|\cdot g(n,\delta(T))+|\mathcal{Q}|\cdot g(n,2)$. By definition of $\delta(T)$, and excluding the trivial case where $m=1$, it follows that $\delta(T) \geq 2$, and thus $g(n,2) \leq g(n,\delta(T))$. Moreover, since all nodes of a PQ-tree are of either type P- or type Q-, it follows that $|\mathcal{P}| + |\mathcal{Q}| \leq m$, as the number of internal nodes in a tree with m leaves is bounded by m.

Therefore, the total running time of Algorithm 1 is upper bounded by $O(nm^3) + m \cdot g(n,\delta(T))$.

□

F On Section 5 (Obvious Independence of Clones)

In this section, we provide the proofs omitted from Section 5 of the main body, as well as a formal definition of extensive games and obviously-dominant strategies (in the restricted setting where each agent has a single information set).

F.1 Proof of Proposition 15

Given $\boldsymbol{\sigma}$ over candidates A with $|A| = m$, consider $d_{\boldsymbol{\sigma}} : B \times B \to [m] \cup \{0\}$ defined for each $a_i, a_j \in A$ as:

$$d_{\boldsymbol{\sigma}}(a_i,a_j) = \min_{\substack{K \subseteq A : a_i,a_j \in K, \\ K \text{ is a clone set w.r.t. } \boldsymbol{\sigma}}} |K| - 1$$

Proposition 15. *For any $\boldsymbol{\sigma}$, $d_{\boldsymbol{\sigma}}$ is a metric over the candidate set A.*

Proof. We prove $d_{\boldsymbol{\sigma}}$ satisfies all axioms of a metric:

- (Zero distance to self) For each $a \in A$, $\{a\}$ is a clone set, so $d(a,a) = |\{a\}| - 1 = 0$.
- (Positivity) If $a \neq b$, then any clone set K that contains both of them must have $|K| \geq 2$, so $d(a,b) \geq 2 - 1 = 1 > 0$.
- (Symmetry) Clearly, $d(a,b) = d(b,a)$.
- (Triangle inequality) Given any $a, b, c \in A$, say K_1 is the clone set that includes a, b with $|K_1| = d(a,b) + 1$ and K_2 is the clone set that includes b, c with $|K_2| = d(b,c) + 1$. Since $b \in K_1 \cap K_2$, we have $K_1 \cap K_2 \neq \emptyset$ so by Axiom (A1) by Elkind et al. [23], we have that $K_1 \cup K_2$ is a clone set. Notice $|K_1 \cup K_2| = |K_1| + |K_2| - |K_1 \cap K_2| \leq (d(a,b)+1)+(d(b,c)+1)-1 = d(a,b)+d(b,c)+1$. Since $a, c \in K_1 \cup K_2$, we have $d(a,c) \leq |K_1 \cup K_2| - 1 \leq d(a,b)+d(b,c)+1-1 = d(a,b)+d(b,c)$, satisfying triangle inequality.

□

F.2 Proof of Proposition 17

Next, we prove that in the strategic candidacy setting where the preferences of candidates are dictates by $d_{\boldsymbol{\sigma}}$, IoC rules not only achieve but strengthen candidate stability.

Proposition 17. *If f is IoC, then R is a dominant strategy in $\Gamma^f_{\boldsymbol{\sigma}}$ for all candidates.*

Proof. Given any $a \in A$, say $u_a(S)$ is the utility of this player in $\Gamma^f_{\boldsymbol{\sigma}}$ when exactly the candidates in $S \subseteq A$ play R, and all candidates in $A \setminus S$ play D, which is a decreasing function of $d_{\boldsymbol{\sigma}}(a, f(\boldsymbol{\sigma}|_S))$ (and minimized at $u_a(\emptyset)$). Fix any $a \in A$ and a pure action for every other candidate. Say S are the candidates among $A \setminus \{a\}$ that played R. To show that R is a dominant strategy for a, we would like to show $u_a(S \cup \{a\}) \geq u_a(S)$. Consider three cases:

1. Case 1: If $f(\boldsymbol{\sigma}|_{S\cup\{a\}}) = \{a\}$, then $u_a(S \cup \{a\}) > u_a(S)$ since $a \notin f(\boldsymbol{\sigma}|_S)$, so $d_{\boldsymbol{\sigma}}(a, f(\boldsymbol{\sigma}|_S)) > 0$, and we are done.
2. Case 2: $f(\boldsymbol{\sigma}|_{S\cup\{a\}}) = f(\boldsymbol{\sigma}|_S)$, then $u_a(S \cup \{a\}) = u_a(S)$ and we are done.
3. Case 3: $S = \emptyset$. Then $u_a(S)$ is the minimizer of u_a and $u_a(S \cup \{a\}) = u_a(\{a\})$ is the maximizer, so we are done.
4. Case 4: If Cases 1-3 are false, we must have $f(\boldsymbol{\sigma}|_{S\cup\{a\}}) = \{b\}$ and $f(\boldsymbol{\sigma}|_S) = \{c\}$ for some $b \neq a$ and $c \neq b$. Take any clone set $K \subseteq A$ with respect to A containing both a and c; we would like to show $b \in K$ (which will automatically apply $d_{\boldsymbol{\sigma}}(a,b) \leq d_{\boldsymbol{\sigma}}(a,c)$). Say $K' = K \cap (S \cup \{a\})$, which is a clone set in $\boldsymbol{\sigma}|_{S\cup\{a\}}$ by Definition 1. Since f is IoC, we have

$$K' \cap f(\boldsymbol{\sigma}|_{S\cup\{a\}}) \neq \emptyset \Leftrightarrow K' \setminus \{a\} \cap f(\boldsymbol{\sigma}|_S) \neq \emptyset.$$

   Since the right hand side is true (as $c \in K' \setminus \{a\} \cap f(\boldsymbol{\sigma}|_S)$), we must have $K' \cap f(\boldsymbol{\sigma}|_{S\cup\{a\}}) \neq \emptyset$, implying $b \in K' \subseteq K$ and therefore $d_{\boldsymbol{\sigma}}(a,b) \leq d_{\boldsymbol{\sigma}}(a,c)$ and $u_a(S \cup \{a\}) \geq u_a(S)$.

□

F.3 Definitions for extensive-form games and obviously dominant strategies

In this section, we introduce extensive-form games and obviously dominant strategies, which we use to argue that CC exposes the obviousness of IoC.

Definition 65. We can define an *extensive-form game* Γ as follows:

1. Γ is represented by a rooted tree structure. The set of all nodes in this tree is denoted by $\mathcal{H}$ with each edge of the tree representing a single game *action*. The game begins at the root, and each action traverses down the tree, until the game finishes at a leaf which we call a *terminal node*. The set of terminal nodes is denoted by $\mathcal{Z} \subset \mathcal{H}$, and the set of actions available at any nonterminal node $h \in \mathcal{H} \setminus \mathcal{Z}$ is denoted by A_h.
2. A finite set of strategic and chance players $|\mathcal{N} \cup \{c\}| = N + 1$ with $N \geq 1$. The set $\mathcal{N}$ contains the *strategic players*, and c stands for a *chance* "player" that models exogenous stochasticity. Each nonterminal node h is assigned to either a strategic player or the chance player, who chooses an action to take from A_h. We call the set of nodes assigned to Player i $\mathcal{H}_i$.
3. For each chance node $h \in \mathcal{H}_c$, a probability distribution $\mathbb{P}_c(\cdot \mid h)$ on A_h with which chance elects an action at h.
4. For each strategic player $i \in \mathcal{N}$, a (without loss of generality) nonnegative *utility (payoff)* function $u_i : \mathcal{Z} \to \mathbb{R}_{\geq 0}$ which returns what i receives when the game finishes at a terminal node. Player i aims to maximize that utility.
5. For each strategic player $i \in \mathcal{N}$, a partition $\mathcal{H}_i = \sqcup_{I \in \mathcal{I}_i} I$ of the nodes of i into information sets (*infosets*). Nodes of the same infoset are considered indistinguishable to the player at that infoset. For that, we also require $A_h = A_{h'}$ for $h, h' \in I$. This also makes action set A_I well-defined.

Strategies and utilities. Players can select a probability distribution—a *randomized action*—over the actions at an infoset. A (behavioral) *strategy* π_i of a player $i \in \mathcal{N}$ specifies a randomized action $\pi_i(\cdot \mid I) \in \Delta(A_I)$ at each infoset $I \in \mathcal{I}_i$. We say π_i is *pure* if it assigns probability 1 to a single action for each infoset. A (strategy) *profile* $\pi = (\pi_i)_{i \in \mathcal{N}}$ specifies a strategy for each player. We use the common notation $\pi_{-i} = (\pi_1, \ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_n)$. We denote the strategy set of Player i with S_i, and $S = \bigtimes_{i \in \mathcal{N}} S_i$.

We denote the reach probability of a node h' from another node h under a profile π as $\mathbb{P}(h' \mid \pi, h)$. It evaluates to 0 if $h \notin \text{hist}(h')$, and otherwise to the product of probabilities with which the actions on the path from h to h' are taken under π and chance. For any infoset, let I^{1st} refer to the nodes $h \in I$ for which I does not appear in $\text{seq}(h)$. Then the reach probability of I is $\mathbb{P}(I \mid \pi, h) \coloneqq \sum_{h' \in I^{\text{1st}}} \mathbb{P}(h' \mid \pi, h)$. We denote with $u_i(\pi \mid h) \coloneqq \sum_{z \in \mathcal{Z}} \mathbb{P}(z \mid \pi, h) \cdot u_i(z)$ the expected utility of Player i given that the game is at node h and the players are following profile π. Finally, we overload notation for the special case the game starts at root node h_0 by defining $\mathbb{P}(h \mid \pi) := \mathbb{P}(h \mid \pi, h_0)$ and $u_i(\pi) \coloneqq u_i(\pi \mid h_0)$.

We now introduce obviously dominant strategies. Since we will focus on games with no exogenous stochasticity (i.e., no chance nodes) and where every player will have a single infoset, our definition is a simplified version of the original definition by Li [44].

Definition 66 (Li 44, Obviously Dominant Strategy)**.** Given an EFG Γ with no chance nodes and a single infoset per player (i.e., $\mathcal{I}_j = \{I_j\}$ for each $j \in \mathcal{N}$) and a player $i \in \mathcal{N}$, an action $s \in A_{I_i}$ is obviously dominant if:

$$\forall s' \in I_i : \qquad \sup_{h \in I_i, \pi_{-i}} u_i((\pi_i^{s'}, \pi_{-i}) \mid h) \leq \inf_{h \in I_i, \pi_{-i}} u_i((\pi_i^{s}, \pi_{-i}) \mid h)$$

where π_i^s is the player i strategy that plays action s with probability 1.

Inutitively, an action s is obviously dominant for player a if for any other action s', *starting from when a must take an action*, the best possible outcome from s' is no better than the worst possible outcome from s. The $\sup$ / $\inf$ over $h \in I_i$ allows us to compare the best and worst possible for i given what she knows at the point where she must act (I_i), and the $\sup$ / $\inf$ over π_{-i} allows us to best and worst possible outcomes based on the strategies of all other players (again, given that I_i is reached), including those that have not acted yet.

For example, $\Gamma^f_{\boldsymbol{\sigma}}$ from the main body of the paper can be interpreted as an EFG where players act simultaneously. In this case, even if the f is IoC, running (R) is *not* an obviously dominant strategy, due to the uncertainty of the actions of every other candidate:

Example 67. *Consider $\Gamma^{STV}_{\boldsymbol{\sigma}}$, where $\boldsymbol{\sigma}$ is from Figure 1. For b, the* worst *outcome of running (R) is that every other candidate plays R too, making d the winner. The* best *outcome of dropping out (D), on the other hand, is for c to play R and d to play D, in which case c wins regardless of what a does. Since $2 = d_{\boldsymbol{\sigma}}(b, c) < d_{\boldsymbol{\sigma}}(b, a) = 4$, candidate b strictly prefers the latter outcome, showing that R is* not *an obviously-dominant strategy for her, even though it is a dominant strategy by Prop. 17.*

F.4 Proof of Theorem 4

Finally, we prove that in the process of implementing a rule f^{CC}, $\Lambda^f_{\boldsymbol{\sigma}}$ achieves obvious strategy-proofness for candidates.

Theorem 4. *For any neutral f, R is an obviously-dominant strategy in $\Lambda^f_{\boldsymbol{\sigma}}$ for all candidates.*

Proof. Take any candidate $a \in A$ and consider the point in $\Lambda^f_{\boldsymbol{\sigma}}$ where a must decide R or D. This happens when Algo. 1 is on the parent node of a, say B. If B is a Q-node, the worst possible outcome of playing R is a winning herself, which is her optimal outcome, and hence the best outcome of D cannot be any better. If is a P-node, then B is the smallest non-trivial clone set that contains a by Lemma 11 (in other words, the members of B are exactly a's second-favorite candidates after herself). Then the worst possible outcome of a running (R) is some other candidate $b \in B \setminus \{a\}$ winning (since Algo. 1 will move out of B only if all the candidates in B, including a, play D), whereas the best possible outcome of a dropping out (D) is, again, some other candidate $c \in B \setminus \{a\}$ winning. Since $d_{\boldsymbol{\sigma}}(a, b) = d_{\boldsymbol{\sigma}}(a, c)$ (both pairs are united by B as the smallest clone set), the latter outcome is no better than the former, proving that R is an obviously-dominant strategy for a. □

F.5 Example representations for $\Gamma^f_{\boldsymbol{\sigma}}$ and $\Lambda^f_{\boldsymbol{\sigma}}$

Figure 8 shows the tree representation of $\Gamma^{STV}_{\boldsymbol{\sigma}}$ from Example 19.

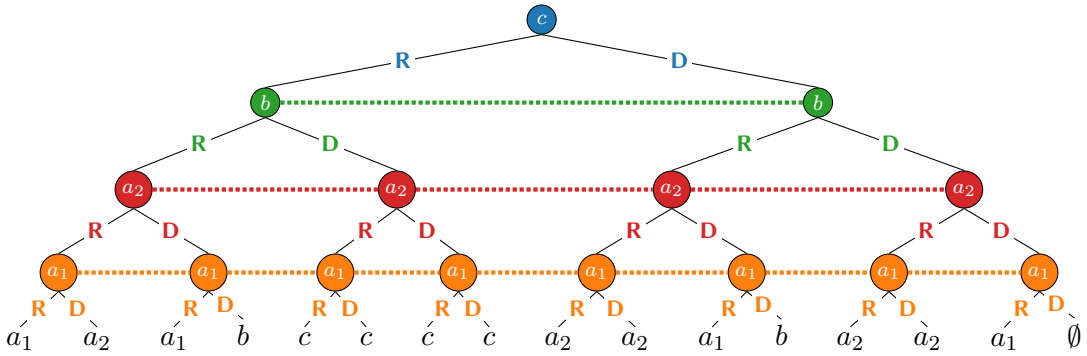

Figure 8: EFG representation of $\Gamma_{\boldsymbol{\sigma}}^{STV}$ for $\boldsymbol{\sigma}$ from Fig. 2. Terminals show the winner under that action profile. Information sets are joined by dotted lines. For a_1, the worst outcome of running is c winning, and the best outcome of dropping out is a_2 winning, so running is not an obviously dominant strategy for a_1.

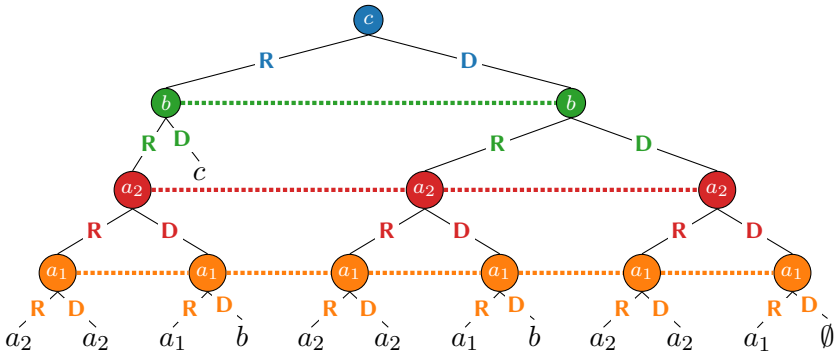

Figure 9: $\Lambda_{\boldsymbol{\sigma}}^{STV^{CC}}$, for $\boldsymbol{\sigma}$ from Fig. 2, the PQ-tree of which is in Fig. 4 (right). For a_1, best outcome of not running is a_2 winning, which is no better than the worst outcome of running, which is also a_2 winning. Therefore, running is an obviously dominant strategy for a_1. A similar analysis applies for all other candidates.