Proceedings of the 23rd International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Voting Compilation Revisited

Yann Chevaleyre[1] , **Jérôme Lang**[1*] , **Nicolas Maudet**[2]
[1]LAMSADE, CNRS, Université Paris-Dauphine, PSL
[2]LIP6, CNRS - Sorbonne Université
{yann.chevaleyre,jerome.lang}@lamsade.dauphine.fr, nicolas.maudet@lip6.fr

Abstract

Compiling a collection of votes (a profile) consists in compressing the information it contains in a minimal way, while still allowing to compute the winner after more votes are received. These additional votes can be understood temporally (when votes come in an asynchronous way) or spatially (when votes are gathered locally in polling stations, and their results published locally before being aggregated on a global level). Given a voting rule, two profiles are equivalent for this rule if for whichever profile we add to each of them, the winner in the two expanded profiles are the same. An equivalence relation between profiles corresponds to the set of information structures (called compilation structures) encoding equivalence classes. Some information structures, such as pairwise majority matrices, are known to be compilation structures of for some common voting rules, while some others, such as majority graphs, are not. We fully characterise the equivalence relations (or equivalently the information structures) that correspond to some voting rules. We list a number of interesting information structures and give known voting rules that correspond to them.

1 Introduction

Compiling a voting collection of votes (called a voting *profile*) consists in compressing the information it contains in a minimal way, while still allowing to compute the winner after more votes are received. There are at least two interpretations for such additional votes:

- *temporal*: votes come in an asynchronous way, and additional votes are those that haven't been cast yet.
- *spatial*: votes are gathered locally in polling stations, and their results published locally before being aggregated on a global level. Compiling local results is useful for distributed vote counting and vote verification (ACE Electoral Knowledge Network 2026; Filtser and Talmon 2019).

Vote compilation is based on the notion of *profile equivalence*: two profiles are equivalent for a voting rule if for whichever profile we add to each of them, the winner in the two expanded profiles will be the same. An equivalence relation between profiles thus corresponds to a set of information structures encoding equivalence classes.

*Corresponding author

Beyond additional votes (with either interpretation), there is another, more fundamental, motivation for studying voting compilation: it is a reasonable way of answering the question "which information does a given voting rule need?", which we develop below.

Identifying the minimal (or coarsest) information structure to be extracted from a profile needed to compute a given voting rule is an important issue in voting theory, that has been touched only superficially. The question appears informally in quite many places (see, as an example, Section 2.5 of (Zwicker 2016)), starting from Fishburn's classification of Condorcet rules in function of their informational requirements (Fishburn 1977).

As an example, the *majority graph* associated with profile P is the directed graph whose vertices are the candidates and that contains a containing an edge from x to y if x is preferred to y in a majority of votes. The *pairwise comparison matrix* (also called weighted majority graph) associated with P specifies, for each pair of candidates x, y, the number of votes preferring x to y. On the one hand, it looks quite intuitive to say that, for instance, the Copeland rule (where the winner maximizes the number of outgoing edges in the majority graph) needs 'only' the majority graph. On the other hand, on closer examination, formally stating something like "the majority graph is all we need to know (that is: the minimal information structure) for computing the Copeland winner" is much more complex than it looks at first glance.

Let us make things more precise. By *information structure* (IS) we mean some information content that can be extracted from the profile by a suitable *compilation function*. The majority graph and the pairwise comparison matrix, mentioned before, are such information structures. Information structures are naturally ordered by specificity: a structure S is a refinement of a structure S' if S' can be determined from S. For instance, the pairwise comparison matrix is a refinement of the majority graph. We say that an IS allows to compute a voting rule f simply if it contains enough information from the profile for the true winner to be determined (that is, the same winner would be determined from the full profile).

Clearly, the majority graph gives us enough information to determine the Copeland winner. But it is not the *minimal* information structure allowing to determine it. More generally, defining the minimal information structure needed to

determine a voting rule, as appealing as it seems, is useless: for almost all voting rules, the smallest information to be extracted from a profile P and that is sufficient for computing $f(P)$ is nothing but $f(P)$, that is, the winner itself.[1]

A way towards a meaningful definition consists at *contextualizing* the collective decision: at the time where P is known there is some uncertainty about which of the state of the world will materialize, and we look for the information to extract from P so that *when the state of the world is known*, we are able to declare a winner. One of the most relevant notions of states pertains to the flow of votes: there might be a few more votes coming in and the information to extract from P must be sufficient to find the winner once this uncertainty is revealed: this is precisely the realm of *compilation* (Chevaleyre et al. 2009; Xia and Conitzer 2010). This is the way we follow; in the conclusion we consider another possible type of uncertainty.

Therefore, *voting compilation is a way of answering the question "what is the minimal information structure we should know to compute a voting rule?"*. Let us come back for a moment to the Copeland rule. As said before, the Copeland rule is based on the majority graph, therefore we might expect that the compilation of a profile is precisely its majority graph. But this is not the case, and it is easy to see why: consider a profile P with two candidates a and b, and three votes: two $a \succ b$ and one $b \succ a$. The majority graph is $a \succ b$. Now, if two more votes $b \succ a$ come, what is the winner? If we know only the majority graph, we cannot know: it can be b, but it can also be a, because the majority graph can correspond to a profile with three votes $a \succ b$, which is indistinguishable from P. The compilation of the Copeland rule should map profiles to a more refined information structure, which is actually the pairwise comparison matrix (Chevaleyre et al. 2009).

We can make an even stronger statement : *the majority graph cannot be the compilation of any voting rule*. Intuitively, this fails because the majority graph of a union of profiles cannot be computed from their majority graphs. This leads us to an intriguing question: which information structures are the compilation of *some* voting rule? We say in that case that the information structure is *compilation implementable* and we answer the question by a characterization. After discussing related work in Section 2 and background on voting in Section 3, Section 4 we define compilation implementability and give the characterization. On a more practical side, in Section 5 we list a selected subset of interesting implementable information structures and in Section 6 we identify the information structures for several common voting rules. We conclude in Section 7.

A longer version of our article is available online at https://www.lamsade.dauphine.fr/%7Elang/papers/CLM26-long.pdf; it contains missing proofs and more examples.

[1]There are rules for which we need even less than $f(P)$, such as constant rules, and more generally rules that are not onto.

2 Related Work

Voting compilation was introduced in (Chevaleyre et al. 2009) and further studied in (Xia and Conitzer 2010). The focus was on finding the compilation complexity of a number of important voting rules, namely, the size (in terms of number of bits) of the minimal information structure. Here the focus is on the opposite direction: we do not start from voting rules, but from information structures and consider them as first-class citizens. Voting compilation is generalized to multiwinner rules (Karia and Lang 2024) and to varying sets of candidates (Chevaleyre et al. 2011). Generalized scoring rules (Xia and Conitzer 2009) use additive compilation functions, with a totally different purpose: votes are compiled to vectors of scores (not necessarily associated with alternatives) and the scores obtained by different votes are summed up. Peters et al. (2020) also make use of compilation functions for embedding voting rules into linear spaces, yet for another purpose (explainability).

Voting protocols and communication complexity of voting rules are studied in (Conitzer and Sandholm 2005) and a few subsequent papers. The compilation of a profile for a voting rule can be seen as a one-round communication protocol (Chevaleyre et al. 2009).

Informational requirements of voting rules are considered in (Sato 2009; Sato 2016), where plurality and plurality with runoff rule are identified as the rules, or more precisely the protocols (with possibly two rounds) that need the minimal information requirement among those that satisfy a number of desirable properties. This relates more closely to communication (or elicitation) protocols than to compilation.

We note that while the term compilation is reminiscent of *knowledge compilation* (Cadoli and Donini 1997), a well-established topic in KR, the purposes differ (minimize storage vs. improve the efficiency of computational tasks).

3 Notation and Background

We consider a fixed set X of $m \geq 2$ candidates. A profile is a collection of n rankings of candidates, for some number of voters $n \geq 1$: $P = (V_1, \ldots, V_n)$.[2] We denote by $\mathcal{P}(n)$ the set of all n-voter profiles and by $\mathcal{P} = \bigcup_{n \in \mathbf{N}} \mathcal{P}(n)$ the set of all profiles. If $P \in \mathcal{P}(n), Q \in \mathcal{P}(q)$ we denote by $P + Q$ the profile in $\mathcal{P}(n+q)$ obtained by concatenating P and Q. Given a ranking V and a profile P We denote by $\#(V, P)$ the number of votes V_i in P such that $V_i = V$. Given a ranking V and a candidate x, $rank(x, V) \in \{1, \ldots, m\}$ is the rank (also called position) of x in V (1 denoting the best rank). We also note $top(V)$ for the best candidate in V (that is, $rank(top(V), V) = 1$) and $ntop(x, P)$ for the number of times x appears in the top position in the profile P. We write votes as ordered lists, e.g., abc for $a \succ b \succ c$.

A resolute voting rule f maps every profile to a winner. As many rules are typically defined in their irresolute version (possibly returning tied winners), we make them reso-

[2]This view of voting rules defined on *ordinal* profiles, where each vote consists of a ranking over candidates, is the most standard view (Zwicker 2016). Our framework can be adapted to other classes of inputs such as approvals or evaluations, but we leave it for further work.

lute, as usual, by composing them with a tie-breaking priority relation $\triangleright$ over candidates.[3]

It is reasonable to focus on *anonymous* voting rules. Not only non-anonymous rules are non desirable from a social acceptability point of view, but compiling them is cumbersome, because compilation functions are based on quantifying over profiles of different size. If f is an anonymous rule, then a profile can simply be seen as a *voting situation*: for each ranking $\succ$ we just specify the number of voters who report it.

Two profiles P and Q are f-equivalent, denoted by $P \sim_f Q$, if for any profile R, $f(P+R) = f(Q+R)$ (Chevaleyre et al. 2009).

A *compilation function* σ is simply a function whose domain is $\mathcal{P}$. The co-domain of σ is called the *information space* associated with σ, and its elements are called *information structures*. Some notable examples of compilation functions map a profile to its majority graph, to its pairwise comparison matrix, to the number of voters who rank candidate x in position j for all x and j, to the number of voters ranking a x on top for each x, etc. Given a compilation function σ, two profiles P and Q are σ-equivalent, denoted by $P \sim_\sigma Q$, if $\sigma(P) = \sigma(Q)$.

For two compilation functions σ and σ', we say that σ' *refines* σ if there is a function h such that for every profile P, $\sigma(P) = h(\sigma'(P))$. By abuse of language we say that the codomain $\sigma(\mathcal{P})$ of σ refines the codomain $\sigma'(\mathcal{P})$ of σ. For instance, the pairwise comparison matrix refines the majority graph. The refinement relation is a lattice, with the most refined compilation function being the identity, and the coarsest being the function with a singleton codomain. Equivalently, σ refines σ' if $\sim_\sigma$ refines $\sim_{\sigma'}$, that is, every equivalence class for $\sim_{\sigma'}$ is the union of equivalence classes for $\sim_\sigma$.

A compilation function σ is compatible with f if $\sim_\sigma$ refines $\sim_f$: informally, if σ is compatible f then for every profile P there is enough information in $\sigma(P)$ to determine $f(P)$.[4] We are especially interested in coarsest compatible compilation functions σ for f, that verify $\sigma(P) = \sigma(Q)$ if and only if $P \sim_f Q$. Up to renaming, there is a unique such function σ_f, which by slight abuse of language we will call *the* compilation function for f.

As an example, consider the plurality rule plu: The plurality winner $plu(P)$ is the candidate maximizing $ntop(., P)$ (with tie-breaking if necessary). The two profiles $P = (abc, acb, bca, cba)$ and $Q = (bac, acb, cab, acb)$ are plu-equivalent. The compilation function σ_f is defined as $\sigma_f(P) = (ntop(x, P)|x \in X\}$. In particular, $\sigma(P) = \sigma(Q) = (a \mapsto 2, b \mapsto 1, c \mapsto 1)$.

Given an equivalence relation $\sim$, and a profile P, its equivalence class is denoted by $[P]$. Conversely, given an equivalence class C, we denote by P_C a profile such that $P_C \in C$ (that is, $[P_C] = C$).

[3] As in (Chevaleyre et al. 2009; Xia and Conitzer 2010) we focus on resolute rules for the sake of simplicity. Most of the setting and results can be extended to irresolute rules, albeit with some complications.

[4] Equivalently, σ is compatible with f if there exists a function ρ such that for any P and R, $f(P+R) = \rho(\sigma(P), R)$ (Chevaleyre et al. 2009).

4 Compilation Implementability

The focus of this section is on information structures, or equivalence relations, that correspond to the compilation of some voting rule. As said in Section 3, we can equivalently work on compilation functions σ or equivalence relations $\sim$. Clearly, for any equivalence relation $\sim$ over profiles there is a compilation function σ such that $\sim_\sigma = \sim$: the function σ that maps any profile P to its equivalence class w.r.t. $\sim$.

Existing work on voting compilation identified the equivalence classes associated with some specific rules. We address here the reverse question: which equivalence relations can correspond to some voting rules? We call such equivalence relations, or equivalently compilation functions, *compilation implementable*, or simply *implementable*. Can we characterize them? For those that are possible, are there interesting voting rules associated to them?

We start by a simple but important observation.

Observation 1. *For any voting rule f, if $P \sim_f Q$ and $P' \sim_f Q'$ then $P + P' \sim_f Q + Q'$.*

Proof. If $P \sim_f Q$ and $P' \sim_f Q'$ then for any R, $f(P+P'+R) = f(Q+P'+R)$ and $f(P'+Q+R) = f(Q'+Q+R)$, which translates into $P + P' \sim_f Q + P'$ and $P' + Q \sim_f Q'+Q$. By transitivity of $\sim_f$, we get $P+P' \sim_f Q+Q'$. □

We say that $\sim$ satisfies *decomposability* if for all P, P', Q, Q', if $P \sim Q$ and $P' \sim Q'$ then $P + P' \sim Q + Q'$. Observation 1 immediately gives a necessary condition for an equivalence relation to be compilation implementable.

Proposition 1. *$\sim$ is implementable only if it satisfies decomposability.*

This simple necessary condition already tells that some equivalence relations are not implementable. For example, the relation induced by the majority graph ($P \sim Q$ if P and Q have the same majority graph) is not implementable, as hinted above.[5] This explains why all Condorcet-consistent rules (based on the majority graph) have the pairwise comparison matrix as compilation function. As another example, $\sim$ defined by $P \sim Q$ if P and Q have the same plurality winner is not implementable either, and more generally, if $\sim$ is defined as $P \sim Q$ if and only if $f(P) = f(Q)$ for some voting rule f then $\sim$ is not implementable as soon as f is such that $f(P + Q)$ cannot be determined from $f(P)$ and $f(Q)$, which holds for almost all reasonable voting rules (we will see an exception in Subsection 6.2).

Observation 2. *The following are equivalent:*

1. *for all P and Q such that $P \not\sim Q$, there is a nonempty profile R such that $P + R \not\sim Q + R$.*
2. *for all P and Q such that $P \not\sim Q$, there is a vote V such that $P + V \not\sim Q + V$.*

[5] Take P containing three votes $a \succ b$, P' containing two votes $a \succ b$ and one vote $b \succ a$, and Q containing two votes $b \succ a$. Then $m(P) = m(P') = \{a \succ b\}$, but $m(P + Q) = \{a \succ b\} \neq m(P' + Q) = \{b \succ a\}$.

Proof. $(2 \Rightarrow 1)$ is obvious. For $(1 \Rightarrow 2)$, assume that 2 does not hold: then for every vote V, $P + V \sim Q + V$, and by induction on the number of votes, we get that for each profile R we have $P + R \sim Q + R$, therefore, 1 does not hold. □

Observation 3. *The following statements are equivalent, where σ is a compilation function, $\sim_\sigma$ the induced equivalence relation ($P \sim_\sigma Q$ if and only if $\sigma(P) = \sigma(Q)$) and $\mathcal{C}_\sigma = \cup_n \mathcal{C}_\sigma(n)$ the partition of profiles into equivalence classes with respect to $\sim_\sigma$.*

1. *$\sim_\sigma$ is decomposable.*
2. *for all P, Q, R, if $P \sim_\sigma Q$ then $P + R \sim_\sigma Q + R$.*
3. *for all P, Q, R, if $\sigma(P) = \sigma(Q)$ then $\sigma(P+R) = \sigma(Q+R)$.*
4. *for all P, P', Q, Q', if $\sigma(P) = \sigma(P')$ and $\sigma(Q) = \sigma(Q')$ then $\sigma(P + Q) = \sigma(P' + Q')$.*
5. *there exists a commutative operator $\oplus$ on $\mathcal{C}_\sigma(n)$ such that for all $C \in \mathcal{C}_\sigma(n)$, $C' \in \mathcal{C}_\sigma(q)$, $P \in C$, $Q \in C'$, we have $P + Q \in C \oplus C'$.*
6. *there exists a commutative operator $\oplus$ on $\mathcal{C}_\sigma(n)$ such that for all $C \in \mathcal{C}_\sigma(n)$, any $P \in C$, and any vote V, we have $P + V \in C \oplus [V]$.*

The straightforward proof is omitted (it can be found in the long version of the article).

Importantly, when decomposability is satisfied, we can abstract away from profiles and work on equivalence classes with respect to $\sim$. Instead of adding profiles, we add equivalence classes (5-6): if C, C' are two equivalence classes with n and n' voters respectively, $C \oplus C'$ is the equivalence class for $n + n'$ voters that corresponds to any $P + P'$ for $P \in C$ and $P' \in C'$. As an example, if equivalence classes correspond to pairwise majority matrices, $\oplus$ is simply the addition of the pairwise majority matrices.

From now on we focus on equivalence relations, and equivalently, compilation functions, that are implementable.

Is decomposability a sufficient condition for implementability? Not quite. To see why, consider this (weird) equivalence relation, where n is the number of voters in the considered profiles, and the number of candidates is $m = 2$ (the candidates are denoted a, b):

- if $n \leq 2$ then $P \sim Q$ if P and Q have the same pairwise majority matrix (which, since $m = 2$, means that they have the same number of votes for each of the two candidates).
- if $n \geq 3$ then $P \sim Q$ for all P and Q.

We can check that $\sim$ satisfies decomposability. But it is not compilation implementable. Suppose it is, that is, $\sim_f = \sim$ for some voting rule f, and let us look at what happens for $n = 2$. There are three equivalence relations (corresponding respectively to 0, 1, and 2 votes for a). Out of these three, a wins in k of them and b in the remaining $3 - k$, therefore, either a or b wins in more than one equivalence class. Without loss of generality, assume a wins in more than one class ($k \geq 2$). Therefore, there are two profiles P and Q such that $P \not\sim Q$, $f(P) = f(Q) = a$ (1). Moreover, for each $n \geq 3$, since there is only one equivalence class, the winner is the same in all n-voter profiles (it may vary with n) (2). Now, (1) and (2) imply that for any R, $f(P + R) = f(Q + R)$, which contradicts $P \not\sim Q$.

We will now give a reasonable *sufficient* condition for an equivalence relation to be implementable (Proposition 2). It will be subsumed by Theorem 1. However we state it and gives its proof, for two reasons: first, it is much easier to follow than the proof of Theorem 1, and helps understanding the latter; it makes it also easier to give a detailed example. Second, *in practice*, it is applicable to all reasonable voting compilation structures (those that are implementable but do not satisfy the sufficient condition of Proposition 2 are extremely weird). In other words, the value of Proposition 2 is practical while that of Theorem 1 is mostly theoretical.

Proposition 2. *$\sim$ is implementable if these two conditions hold:*

1. *$\sim$ is decomposable.*
2. *for all P and Q such that $P \not\sim Q$, there is a vote V such that $P + V \not\sim Q + V$.*

Proof. Assume Conditions 1 and 2 hold. We construct a voting rule f such that $\sim_f = \sim$. Let σ be the compilation function such that $\sigma(P)$ is the equivalence class of profile P with respect to $\sim$.

The following algorithm defines f for profiles of size 1 to n. It is equivalent to define f for all profiles or, more simply for all equivalence classes of $\mathcal{C}(n)$, for all n.

For each integer j, let $\mathcal{C}_\sigma(j)$ be the equivalence classes of $\mathcal{P}(j)$ with respect to $\sim$. The algorithm maintains a finite list $Fixed$ of equivalence classes, which are assigned to a winner. Because 1. holds, we know there exists $\oplus$ such that $f(P + P') = f(P_C)$, where $C = [P] \oplus [P']$.

```
Fixed ← ∅
for j ← 1 to n do
    for all C ∈ C_σ(j) \ Fixed do
        Assign C to some winner f(C); add C to Fixed
    end for
    for all C, C' ∈ C_σ(j) such that f(C) = f(C') do
        Find a class C_R (*) such that
          (a) C ⊕ C_R ≠ C' ⊕ C_R, and
          (b) C ⊕ C_R ∉ Fixed or C' ⊕ C_R ∉ Fixed.
        if C⊕C_R ∉ Fixed and C'⊕C_R ∉ Fixed then
            Assign f(C ⊕ C_R) and f(C' ⊕ C_R) to
              two different candidates
            Add C ⊕ C_R and C' ⊕ C_R to Fixed
        else  ▷ exactly one of C ⊕ C_R and C' ⊕ C_R is
              ▷ in Fixed, without of generality C ⊕ C_R
            Assign f(C' ⊕ C_R) such that
              f(C ⊕ C_R) ≠ f(C' ⊕ C_R)
            Add C' ⊕ C_R to Fixed
        end if
    end for
end for
```

We first prove that (*) for all $C, C' \in C \in \mathcal{C}_\sigma(j)$ such that $C \neq C'$ and $f(C) = f(C')$, there exists an equivalence class R such that $C \oplus C_R \neq C' \oplus C_R$ and at least one of $C \oplus C_R$ and $C' \oplus C_R$ is not in $Fixed$. We know from Condition

2 that there exists a vote V_1 such that $C \oplus [V_1] \neq C' \oplus [V_1]$. If at least one of $C \oplus [V_1]$ and $C' \oplus [V_1]$ is not fixed, then we take $C_R = [V_1]$ and we are done. If both of them are fixed, then there exists a vote V_2 such that $C + [V_1 + V_2] \neq C' + [V_1 + V_2]$. If at least one of $C + [V_1 + V_2]$ and $C' + [V_1 + V_2]$ is not fixed, then we take $C_R = [V_1 + V_2]$ and we are done. Otherwise, there exists V_3 such that $C + [V_1 + V_2 + V_3] \neq C' + [V_1 + V_2 + V_3]$, and so on. Now, at any step of the algorithm, $Fixed$ is finite, so this loop has to end up after at most k steps and we can take $C_R = [V_1 + V_2 + \ldots + V_k]$.

Now, we prove that $\sim_f = \sim$. For this we have to prove that for each $P, Q \in \mathcal{P}(j)$, (a) if $P \sim Q$, then $f(P + R) = f(Q + R)$ for all R, and (b) if $P \not\sim Q$, then $f(P + R) \neq f(Q + R)$ for some R. For (a): if $P \sim Q$, then $[P] = [Q]$ and $P + R \sim Q + R$ for all R, by hypothesis. Since f is defined by assigning a winner to each equivalence class (that is, implicitly, to all profiles in each equivalence class), we have $f(P + R) = f(Q + R)$ for all R. For (b): if $P \not\sim Q$, then the algorithm finds an C_R such that $[P] + C_R$ and $[Q] + C_R$ are not both fixed and assigns them to two different winners, therefore $f(P + R) \neq f(Q + R)$ for any $R \in C_R$. □

The weird equivalence defined above (to show that decomposability alone is not a sufficient condition for implementability) does not satisfy Condition 2.

Now, the sufficient condition of Proposition 2 is not necessary. To see why, consider all equivalence relations $\sim$ such that, for each j, $\mathcal{C}_\sigma(j)$ contains at most m equivalence classes and that for all $C, C' \in \mathcal{C}_\sigma(j)$, $C \neq C'$ implies $f(C) \neq f(C')$ — which is possible precisely because $|\mathcal{C}_\sigma(j)| \leq m$. Clearly, every such $\sim$ is implementable, although some of them do not satisfy Condition 2: for instance, $\mathcal{C}_\sigma(1)$ contains m equivalence classes (so that every profile is alone in its equivalence class) and for each $j \geq 2$, $\mathcal{C}_\sigma(j)$ contains only one class. $\sim$ fails Condition 2 for $n = 1$, but $\sim$ is implementable by a rule: for $n = 1$, the unique voter is a dictator, and for $n \geq 2$, the rule is constant.

Now we give the full characterisation.

For two pairs of equivalence classes (C, C') of $\mathcal{C}_\sigma(j)$, we say that (C, C') is a *problematic pair* if $C \neq C'$ and for any vote V, $C + [V] = C' + [V]$. The *problematic graph* $G_{\sim,n}$ is the graph whose vertices are all problematic classes of $\mathcal{C}(n)$ and containing an edge (C, C') if (C, C') is problematic.

The problematic relation on classes is a strict equivalence relation: it is irreflexive, symmetric, and satisfies full transitivity on distinct elements: if (C, C') and (C', C'') are both problematic and $C \neq C''$ then (C, C'') is problematic.

Therefore the problematic relation consists of disjoint cliques, and isolated nodes corresponding to classes that are never included in a problematic pair. An equivalence class for this relation is called a *problematic cluster*: $\Gamma \subseteq \mathcal{C}_\sigma(j)$ is a *problematic cluster* if (C, C') is problematic for all $C, C' \in \Gamma$, and for each $C \in \Gamma$ and $C'' \notin \Gamma$, (C, C'') is not problematic. An isolated node is also a problematic cluster.

Theorem 1. *$\sim$ is implementable if and only if these two conditions hold:*

1. *$\sim$ is decomposable.*
2. *for all n, every problematic cluster of $G_{\sim,n}$ is of size at most m.*

Proof. If 1 does not hold then we already know that $\sim$ is not implementable.

If 2 does not hold, then for some n there is a problematic cluster of size $> m$. Therefore, there is a pair $(C, C') \in \mathcal{C}_\sigma(n)$ such that $C \neq C'$, $f(C) = f(C')$ and $C + [V] = C' + [V]$ for every vote V. From Observation 2, this means that we should have $C = C'$, which contradicts $C \neq C'$.

Now, assume Conditions 1 and 2 hold. We show that there exists a voting rule f such that $\sim_f = \sim$.

For any fixed n, the following algorithm defines f for profiles of size 1 to n. Like the algorithm used in the proof of Proposition 2, it maintains a set $Fixed$ of classes that have already been assigned a winner, and repeatedly assigns some winner to every class of $\mathcal{C}_\sigma(j)$ for $j = 1, \ldots, n$, and also to a finite number of equivalence classes of $\mathcal{C}_\sigma(j)$ for $j > n$, to ensure the property that for each two classes $C \neq C'$ there is C_R (empty or not) such that $f(C \oplus C_R) \neq f([C' \oplus C_R])$.

Given a class C, we define $Poss(C)$ (relatively to a stage of the algorithm) as follows:

- if $C \in Fixed$, then it is assigned to a winner x and we set $Poss(C) = \{x\}$.
- if $C \notin Fixed$, let Γ be its problematic cluster (possibly a singleton). $Poss(C)$ is the set of all candidates in X that have not been assigned to a class of Γ, that is, $Poss(C) = \{x : f(C') \neq x \text{ for all } C' \in \Gamma \cap Fixed\}$. In particular, if C is an isolated class then $Poss(\Gamma) = X$.

```
1:  Fixed ← ∅
2:  for j ← 1, ..., n do
3:      Identify all problematic clusters including isolated
        classes) of C_σ(j)
4:      for all problematic clusters Γ of C_σ(j) do
5:          for all C ∈ Γ \ Fixed do
6:              Assign C to f(C) such that f(C') ≠ f(C)
7:                  holds for all C' ∈ Γ ∩ Fixed (**)
8:              Fixed ← Fixed ∪ {C}
9:          end for
10:     end for
11:     for all (C, C') such that f(C) = f(C') do
12:         Find a profile R such that
13:             |Poss(C ⊕ [R]) ∪ Poss(C' ⊕ [R])| ≥ 2 (***)
14:         Let NF = ({C ⊕ [R], C' ⊕ [R]} \ Fixed
15:         Assign winners to NF in such a way that
16:             f(C ⊕ [R]) ≠ f(C' ⊕ [R])
17:         Fixed ← Fixed ∪ {C}
18:     end for
19: end for
```

Property 1 The identification of problematic cluster is done explicitly by considering all pairs of classes (C, C'), check if (C, C'), is problematic by considering all possible votes V and check if $C \oplus [V] = C' \oplus [V]$ for all V, and apply transitive closure.

Invariant 1 A problematic cluster Γ never contains classes C and C' such that $C, C' \in Fixed$ and $f(C) = f(C')$. It is easily proven by induction. It is true initially. Classes of Γ are assigned a winner at steps 6 and 15. In both cases, the assignment is always made in such a way that no other class in Γ is assigned the same value as C. This is always possible because $|\Gamma| \leq m$.

Invariant 2 At any step of the algorithm, $Fixed$ is finite. This holds because at every stage, only a finite number classes are be added in $Fixed$.

Property 2 $Poss(C \oplus [R]) = \{x\}$ if and only if one of the two following conditions holds: (1) $C \oplus [R]$ is in $Fixed$ and assigned to x; (2) $C \oplus [R]$ belongs to a problematic cluster Γ which contains m classes all of which except C are in $Fixed$ and such that x is the only candidate to which none of these classes has been assigned. In the latter case, we will say that C is *implicitly fixed*. Thus, $|Poss(C \oplus [R]) \cup Poss(C' \oplus [R])| = 1$ if $C \oplus [R]$ and $C' \oplus [R]$ are assigned, explicitly or implicitly, to the same winner. This also means that if $|Poss(C \oplus [R]) \cup Poss(C' \oplus [R])| \geq 2$ then at least one of $C \oplus [R]$ and $C' \oplus [R]$ is not fixed (explicitly or implicitly) and can still be assigned to two different winners.

Invariant 3 At step 11, whenever $f(C) = f(C')$, there exists a class $[R]$ such that $|Poss(C \oplus [R]) \cup Poss(C' \oplus [R])| \geq 2$. First, $f(C) = f(C')$ implies that (C, C') is unproblematic (Invariant 1). By definition of nonproblematicity, there exists a vote V_1 such that $C + [V_1] \neq C' + [V_1]$. If $|Poss(C \oplus [V_1]) \cup Poss(C' \oplus [V_1])| \geq 2$, then we are done. Otherwise, $|Poss(C \oplus [V_1]) \cup Poss(C' \oplus [V_1])| = 1$ and by Property 1, $C \oplus [V_1]$ and $C' \oplus [V_1]$ are fixed (implicitly or explicitly) and assigned to the same winner. This implies that $(C \oplus [V_1], C' \oplus [V_1])$ is unproblematic (Invariant 1), therefore there exists a vote V_2 such that $C + [\{V_1, V_2\}] \neq C' + [\{V_1, V_2\}]$. If $|Poss(C \oplus [\{V_1, V_2\}]) \cup Poss(C' \oplus [\{V_1, V_2\}])| \geq 2$, then we are done, otherwise there exists a vote V_3, and so on. For every iteration where $C \oplus [\{V_1, \ldots, V_k\}]$ and $C' \oplus \{V_1, \ldots, V_k\}$ are fixed (implicitly or explicitly), $Fixed \cap \mathcal{C}_\sigma(|C| + k) \neq \emptyset$ (note if a class is implicitly fixed, then it belongs to a problematic cluster of cardinality at least 2 where all classes but one are in $Fixed$). Because $Fixed$ is finite (Invariant 2), there must be an integer k such that $Poss(C \oplus [\{V_1, \ldots, V_k\}] \cup Poss(C' \oplus \{V_1, \ldots, V_k\}) \geq 2$.

It only remains to prove that $\sim_f = \sim$. *This is done exactly as in the proof of Proposition 2.* □

5 A Zoo of Implementable Information Structures

We now move towards a more practical study of compilation functions. Our aim is to show the variety of information structures (or equivalence relations) that are implementable by common voting rules. In (Chevaleyre et al. 2009; Xia and Conitzer 2010), the following correspondences between voting rules and compilation were identified:

- plurality rule (plu): the compilation is the vector of plurality scores of candidates in P, that is, $\sigma_{plu}(P) = ntop(., P)$; equivalently $P \sim_{plu} Q$ if the plurality scores of every candidate x are the same in P and Q.
- Borda rule: the compilation is the vector of Borda scores of all candidates. This applies more generally to all positional scoring rules.
- Condorcet-consistent rules based on the majority graph or, more generally, on the pairwise majority matrix: the compilation is the pairwise majority matrix of P (*pmm*).
- plurality with runoff: the compilation is the weighted majority graph together with the plurality scores.
- Instant Runoff Voting (also called STV): the compilation gives, for all $A \subset X$ and $x \notin A$, the number of votes in P who rank x above all candidates in A. We call this information structure *best-from-subset* (*bfs*).

Here we list a number of *other* interesting implementable anonymous information structures.

anonymous is the number of voters reporting every possible ranking (also known as a *voting situation*).

positional is the matrix specifying, for each candidate x and integer $j \leq m$ (or $m - 1$), the number of votes in P ranking x in position j.

worst-from-subset (*wfs*) is the table specifying, for each $A \subset X$ and $x \notin A$, the number of votes in P who rank x below all candidates in A.

between is the table specifying, for each $A \subset X$ and $x, y \notin A$, the number of votes in P who rank x above all candidates in A, and all candidates in A above y.

omninomination is the (single) most prioritary (by tie-breaking) candidate ranked first in at least one vote of P.

The fact that all information structures of the above list are implementable follows directly from Proposition 2. Moreover, for all of them except **worst-from-subset** and **between**, we exhibit a suitable voting rule in Section 6. For these two information stuctures, we provide conjectures at the end of the discussion. All considered structures above are depicted on Figure 1, ordered by coarsening.

Notice that for most (probably all) connected pairs of structures in Figure 1, there are implementable compilation structures that are strictly inbetween. The argument uses tie-breaking refinements of a rule by another one: for instance, between positional and plurality scores, we can take the vector of all plurality scores and the vector of all two-approval scores (number of votes in which a candidate is ranked first or second).

We could of course find other implementable information structures, but it seems that the number of *natural* compilations is rather limited. And what's more, the number of compilation functions that correspond to rules that have been studied in the literature is even more limited.

6 Case Studies

In this section the aim is twofold: we show that some of the implementable information structures listed above correspond to known voting rules, and we also exhibit some

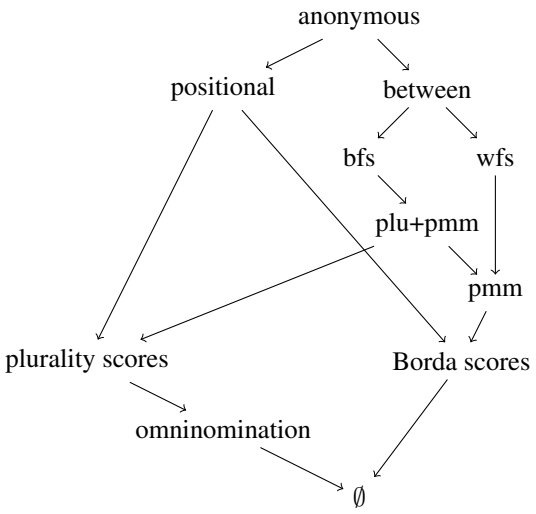

Figure 1: A selection of implementable structures, ordered by coarsening

important voting rules for which the associated compilation structure is none of the above. In the following proofs, the sufficiency part (showing that equivalent profiles for the compiled information structure are indeed equivalent for the voting rules) are relatively easy. The necessary parts are often much more involved, and some of them are deferred to the long version of our article.

6.1 Anonymous Compilation: the Modal Rule

The finest anonymous compilation function is the function that maps every ranking to its number of occurrences in the profile; the corresponding equivalence relation is $P \sim_{ano} Q$ if and only if for every vote V, $\#(V,P) = \#(V,Q)$.

The *modal* rule (Caragiannis, Procaccia, and Shah 2014) is defined as follows: given profile P, $V^*(P)$ is the vote that appears most often in P, and $modal(P) = top(V^*(P))$. (If more than one vote appears most often in P, we gather the tops of all tied votes and use tie-breaking priority.)

Proposition 3. *$P \sim_{mode} Q$ if and only if $P \sim_{ano} Q$.*

Proof. If $\#(V,P) = \#(V,Q)$ for all V, then for all R, $\#(V,P+R) = \#(V,Q+R)$ and therefore $modal(P+R) = modal(Q+R)$.

Conversely, assume that $P, Q \in \mathcal{P}(n)$ and $\#(V,P) \neq \#(V,Q)$ for some V. Without loss of generality, assume $\#(V,P) > \#(V,Q)$ (1). Let $top(P) = x$.

Case 1: there is a vote V' such that $top(V') = y \neq x$ and $\#(V',Q) > \#(V',P)$ (2). Let R consisting of $n+1+\#(V',P) - \#(V,P)$ votes V and n votes V'. We have

- $\#(V,P+R) = \#(V,P)+n+1+\#(V',P)-\#(V,P) = n+1+\#(V',P)$, and
- $\#(V',P+R) = \#(V',P)+n$
- for all $V'' \neq V,V'$, $\#(V'',P+R) = \#(V'',P) < n$ (because (1) implies $\#(V',Q) > 0$).

Therefore, $V^*(P+R) = V$ and $modal(P+R) = x$. Now,

- $\#(V,Q+R) = \#(V,Q)+n+1+\#(V',P)-\#(V,P) = n+1+\#(V',P)+(\#(V,Q)-\#(V,P)) \leq n+\#(V',P)$, because $\#(V,Q) < \#(V,P)$ (1).
- $\#(V',Q+R) = \#(V',Q)+n > \#(V',P)+n$ because of (2).
- for all $V'' \neq V,V'$, $\#(V'',Q+R) = \#(V'',Q) < n$ ((because (2) implies $\#(V',Q) > 0$).

Therefore, $V^*(Q+R) = V'$ and $modal(Q+R) = y \neq x$.

Case 2: for every vote V' such that $top(V') \neq x$ we have $\#(V',Q) \leq \#(V',P)$ (3), and there is a vote V^* such that $top(V^*) \neq x$ and $\#(V^*,Q) < \#(V^*,P)$ (4). From (3) and (4) and the pigeonhole principle, there is a vote $V^\bullet$ such that $top(V^\bullet) = x$ and $\#(V^\bullet,Q) > \#(V^\bullet,P)$. Then we apply the same line of reasoning as in Case 1, with V^* and $V^\bullet$ playing the roles of V and V'.

Case 3: for every vote V' such that $top(V') \neq x$ we have $\#(V',Q) = \#(V',P)$ (5).

If x does not have the topmost tie-breaking priority, take some $y \neq x$ such that $y \triangleright x$ and some $\hat{V}$ with $top(\hat{V}) = y$.

Let R consisting of $n + \#(\hat{V},P) + 1$ votes V and $n + \#(V,P)$ votes $\hat{V}$. We have $\#(V,P+R) = n+\#(\hat{V},P)+\#(V,P)+1$ and $\#(\hat{V},P+R) = n+\#(V,P)+\#(\hat{V},P)$, and for all $V' \neq V,\hat{V}$, $\#(V',P+R) = \#(V',P) \leq n$. Therefore, $\#(V,P+R) > \#(\hat{V},P+R)) > \#(V',P+R)$ for all $V' \neq V,\hat{V}$, and $modal(P+R) = x$.

We have $\#(V,Q+R) = n + \#(\hat{V},P) + \#(V,Q) + 1$. Because of (5) and $top(V') \neq x$, $\#(V',Q) = \#(V',P)$ and $\#(V,Q+R) = n+\#(\hat{V},Q)+\#(V,Q)+1$. Now, $\#(\hat{V},Q+R) = \#(\hat{V},Q) + \#(V,P) + n > \#(\hat{V},Q) + \#(V,Q)$ because of (1). Therefore, $\#(\hat{V},Q+R) \geq \#(V,Q+R)$. Moreover, for all $V' \neq V,\hat{V}$, $\#(V',Q+R) = \#(V',Q) \leq n < \#(\hat{V},Q+R)$. Since $y \triangleright x$, we have $modal(Q+R) = y \neq modal(P+R)$.

If x has the topmost tie-breaking priority, the same construction works with $n + \#(\hat{V},P)$ votes V in R instead of $n + \#(\hat{V},P) + 1$.

□

6.2 Winner-Based Compilation: Omninomination

We have seen that the modal rule uses maximal information over all anonymous rules. At the other extremity, the minimal information needed by an anonymous and onto rule consists *of the current winner*: this is necessary because one must be able to recover the current winner from the compilation (which corresponds to adding $R = \emptyset$). (Of course, one can do even better for rules that are not onto, with constant rules at the extreme case.) Are there natural rules whose compilation consists of the current winner? We answer positively below.

We define the omninomination rule $Omni_\triangleright$, where $\triangleright$ is a tie-breaking priority: for a profile P, $Omni_\triangleright(P)$ is the most prioritary candidate, w.r.t., $\triangleright$, among those ranked first in at least one vote of P. It is a resolute version of the (irresolute) omninomination rule (Gärdenfors 1976).

Proposition 4. *$P \sim_{Omni_\triangleright} Q$ if and only if $Omni_\triangleright(P) = Omni_\triangleright(Q)$.*

Proof. Let us define $\triangleright(x, y) = x$ if $x \triangleright y$ and y otherwise. If $P \sim_{Omni_\triangleright} Q$ then for any R, $Omni_\triangleright(P + R) = \triangleright(Omni_\triangleright(P), Omni_\triangleright(R)) = \triangleright(Omni_\triangleright(Q), Omni_\triangleright(R)) = Omni_\triangleright(Q + R)$. Now, assume $P \not\sim_{Omni_\triangleright} Q$, that is, $Omni_\triangleright(P) \neq Omni_\triangleright(Q)$. Without loss of generality, $Omni_\triangleright(P) \triangleright Omni_\triangleright(Q)$. Take $R = Q$. We have $Omni_\triangleright(P + Q) = Omni_\triangleright(P)$ and $Omni_\triangleright(Q + Q) = Omni_\triangleright(Q) \neq Omni_\triangleright(P)$. □

6.3 Full Positional Compilation: Lexicographic Plurality

The *lexicographic plurality refinement (LPR)* rule (see, e.g., (Postlewaite and Schmeidler 1986; Goldsmith et al. 2014) is defined as follows. Given profile P, candidate x and rank $i \in \{1, \ldots, m\}$, recall that $pos(x, i, P)$ is the number of voters in P in which x is ranked in position i. We call $(pos(x, i, p))_{i=1\ldots m}$ the positional vector of x w.r.t. P, and $Pos(P) = (pos(x, i, P) \mid x \in X, i \leq m)$ the positional matrix of P. The LPR winner is the plurality winner if there is only one plurality winner, otherwise the candidate among plurality winners most frequently ranked in position 2 if there is only one such candidate, and so on. The priority relation is used only if the tied candidates in Y all have the same positional vector. Formally:

$i \leftarrow 1$
$Y \leftarrow X$
repeat
  $Y \leftarrow \{\, y \in Y \mid y$ maximizes $pos(y, i, P) \,\}$
  $i \leftarrow i + 1$
until $|Y| = 1$ **or** $i = m$
return the most priority candidate in Y

Proposition 5. *$P \sim_{LPR} Q$ if and only if $Pos(P) = Pos(Q)$.*

Proof. As the definition of $LPR(P)$ is defined only from $Pos(P)$, if $Pos(P) = Pos(Q)$ then $LPR(P) = LPR(Q)$.

Assume $Pos(P) \neq Pos(Q)$. Let k be the smallest integer for which there is some x such that $pos(x, k, P) \neq pos(x, k, Q)$. Because $Pos(P) \neq Pos(Q)$, k is defined and $k \leq m - 1$ ($k \neq m$ because $\sum_{j=1}^{n} pos(x, j, P) = \sum_{j=1}^{n} pos(x, j, Q) = n$). Without loss of generality, assume $pos(x, k, P) > pos(x, k, Q)$. By the pigeonhole principle, there is a y such that $pos(y, k, Q) > pos(x, k, P)$. We assume without loss of generality that x has priority over y.

We construct profile R as follows. We define $P^{x \leftrightarrow y}$ as the profile with n votes, one for each vote of P with the positions of x and y exchanged; R^x the profile with $2n + 1$ votes with x ranked first and y last (other candidates ranked in arbitrary order); and R^y the profile with $2n+1$ votes with y ranked first and x in position $k+1$ (other candidates ranked in arbitrary order). R is the $5n + 2$-votes profile defined as $R = P^{x \leftrightarrow y} + R^x + R^y$. We observe that

1. $pos(x, 1, P + R) = pos(y, 1, P + R) = pos(x, 1, P) + pos(y, 1, P) + 2n + 1$.
2. if $k > 1$ then $pos(x, 1, Q + R) = pos(x, 1, Q) + pos(y, 1, P) + 2n + 1 = pos(x, 1, P) + pos(y, 1, P) + 2n + 1 = pos(x, 1, P) + pos(y, 1, Q) + 2n + 1 = pos(y, 1, Q + R)$.
3. for each $z \neq x, y$, $pos(x, 1, P + R) \leq 2n$.
4. for each $z \neq x, y$, $pos(x, 1, Q + R) \leq 2n$.
5. for each $2 \leq j \leq k$, $pos(x, j, P + R) = pos(y, j, P + R) = pos(x, j, P) + pos(y, j, P)$
6. for each $2 \leq j < k$: $pos(x, j, Q + R) = pos(x, j, Q) + pos(y, j, P) = pos(x, j, P) + pos(y, j, P)$; and $pos(y, j, Q + R) = pos(y, j, Q) + pos(x, j, P) = pos(y, j, P) + pos(x, j, P) = pos(x, j, Q + R)$.
7. $pos(x, k, Q + R) = pos(x, k, Q) + pos(y, k, P)$ and $pos(y, k, Q + R) = pos(y, k, Q) + pos(y, k, P)$. Because $pos(x, k, Q) < pos(x, k, P)$ and $pos(y, k, P) < pos(y, k, Q)$, we have $pos(y, k, Q + R) - pos(x, k, Q + R) = pos(y, k, Q) + pos(x, k, P) - pos(x, k, Q) - pos(y, k, P) = (pos(y, k, Q) - pos(y, k, P)) + (pos(x, k, P) - pos(x, k, Q)) > 0$.
8. if $k < m - 1$ then $pos(x, k + 1, P + R) \geq 2n + 1$
9. if $k < m - 1$ then $pos(y, k + 1, P + R) \leq 2n$
10. if $k = m - 1$ then $pos(x, m, P + R) = pos(y, m, P + R) = pos(x, 1, m) + pos(y, 1, m) + 2n + 1$

We distinguish three cases:

- Case 1: $k = 1$. From 1 and 3, x and y are tied with the highest plurality score, therefore $LPR(P + R) \in \{x, y\}$. Now, from 8 and 9 we have $LPR(P + R) = x$. Finally, from 4 and 7, we have $LPR(Q + R) = y$.
- Case 2: $2 \leq k \leq m-1$. From 1 and 3 we have $LPR(P + R) \in \{x, y\}$ as in Case 1. Then from 5, 8 and 9 we have $LPR(P+R) = x$. From 2 and 4 we have $LPR(Q+R) \in \{x, y\}$. From 6 and 7 we have $LPR(Q + R) = y$.
- Case 3: $k = m-1$. From 1 and 3 we have $LPR(P+R) \in \{x, y\}$. From 5 and 10, we have $pos(x, j, P + R) = pos(y, j, P + R)$ for all j and we have to use the tie-breaking priority to conclude $LPR(P + R) = x$. From 6 and 7 we have $LPR(Q + R) = y$.

In all cases, we have $LPR(P + R) \neq LPR(Q + R)$. □

Example 1. *$P = (abcde, bcdea, dcabe, cdbae, deabc)$, $Q = (bcade, cdbae, abcde, dcbea, decba)$. We check that $k = 3$ and we can take $x = a$ and $y = b$. R consists in the five votes $(bacde, acdeb, dcbae, cdabe, debac)$ plus 11 votes $a \times \times \times b$ and 11 votes $b \times \times a \times$. The following table partly describes the position matrices (what is omitted is useless) for $P + R$ (left) and $Q + R$ (right):*

	1	2	3	4	5
a	13	1	3	14	1
b	13	1	3	3	12
c, d, e	≤ 10				

	1	2	3
a	13	1	2
b	13	1	4
c, d, e	≤ 10		

We check that $LPR(P + R) = a$ and $LPR(Q + R) = b$.

6.4 Bucklin

The *Bucklin* rule is defined as follows: given a profile P and $x \in X$, $BS(x, P)$ (the *Bucklin score* of candidate x for profile P) is the smallest integer k such that a majority of votes V_i in P such that $rank(V_i, x) \leq k$. Then we take $k^*(P) = \min_{x \in X} BS(x, P)$, and the Bucklin winner is then the $k^*(P)$-approval winner, that is, the candidate maximizing the number of votes in P ranking it in positions 1 to $k^*(P)$. While this means that the Bucklin rule can be determined from the positional matrix, Proposition 6 shows that only its upper half (up to one unit) is sufficient and necessary.

Proposition 6. *$P \sim_{Bucklin} Q$ if and only if for all $x \in X$, and $k \leq \frac{m}{2} + 1$, we have $pos(x, k, P) = pos(x, k, Q)$.*

For sufficiency, we first remark that for any m-candidate profile there always exists a candidate x with Bucklin score at most $\frac{m}{2} + 1$. This implies that $k^*(P + R) \leq \frac{m}{2} + 1$ and $k^*(Q + R) \leq \frac{m}{2} + 1$, and since $pos(x, k, P) = pos(x, k, Q)$ for all $k \leq \frac{m}{2} + 1$, we have $k^*(P + R) = k^*(P + R)$, and $Bucklin(P + R) = Bucklin(Q + R)$.

The proof of necessity is rather long, and as this rather uncommon compilation structure is not central in our study, the proof is postponed to the long version of our article.

6.5 Minimax Rank

We now give one rule for which the compilation structure is more 'exotic' : the *minimax rank rule* (Congar and Merlin 2012). Given a profile $P = (V_1, \ldots, V_n)$ and a candidate x, the minimax rank of x in P is $MaxR(x, P) = \max_{i=1,\ldots,n} rank(x, V_i)$, and the $MMR(P)$ is the most prioritary candidate in $\operatorname{argmin}_{x \in X} MaxR(x, P)$.

Proposition 7. *$P \sim_{MMR} Q$ if and only if the following two conditions hold:*

1. *$MMR(P) = MMR(Q) = x$.*
2. *for all $y \neq x$, $MaxR(y, P) = MaxR(y, Q)$.*

Before we give the proof, we note that what we need to know is *almost* $(MaxR(., P), x \in X)$ but not quite; we don't need to know the maximum rank of the winner. To see that, consider $P = (abcde, edcba)$ and $Q = (acbde, ecdba)$, and lexicographic tie-breaking. We have $MaxR(x, P) = MaxR(x, Q) \geq 4$ for all $x \neq c$, and $MaxR(c, P) = 3 \neq MaxR(c, Q) = 2$. The winner is c in both profiles. Suppose we add a profile R. If $MaxR(c, R) \leq 3$ then we don't need to know the exact value of $MaxR(c, R)$ to conclude that c is still the winner in $P + R$ and $Q + R$. If $MaxR(c, R) \geq 4$ then we know the values of $MaxR(c, P + R)$ and $MaxR(c, Q + R)$ (and also the values of $MaxR(., P + R)$ and $MaxR(., Q + R)$ for other candidates) so we are able to determine the winner.

Proof sketch. The left-to-right direction is a tedious case analysis. We postpone it to the supplementary material.

For the right-to-left direction: first, if $MMR(P) \neq MMR(Q)$ then obviously $P \not\sim_{MMR} Q$. If $MMR(P) = MMR(Q) = x$ and there is some $y \neq x$ such that $MaxR(P, y) \neq MaxR(Q, y)$, then without loss of generality, $MaxR(P, y) = \varphi$ and $MaxR(Q, y) = \psi > \varphi$. Because $MMR(P) = MMR(Q) = x$, we have $\psi > \varphi \geq MaxR(x, P), MaxR(x, Q)$.

If x has priority over y then we take a vote V with y in first position and x in position $\varphi + 1 \leq \psi$. We have $MaxR(x, P + V) = MaxR(x, Q + V) = \varphi + 1$, $MaxR(y, P+V) = \varphi$ and $MaxR(y, Q+V) = \psi \geq \varphi+1$, therefore $MMR(P + R) = y$ and $MMR(Q + R) = x$.

If y has priority over x then we take a vote V with y in first position and x in position φ. We have $MaxR(x, P + V) = MaxR(x, P + V) = \varphi$, $MaxR(y, P + V) = \varphi$ and $MaxR(y, P + V) \geq \varphi + 1$, therefore $MMR(P + R) = y$ and $MMR(Q + R) = x$. □

We end this section by mentioning two other rules that we looked at: Young and Dodgson. Given profile P, the Young score of x is the largest subprofile P' (obtained by removing the minimal number of voters from P) for which x is a Condorcet winner ($+\infty$ if this is not possible). The Young winner is the candidate with smallest Young score. The Dodgson rule is similar except that instead of removing a minimum number of voters we perform a minimum number of swaps of adjacent candidates in votes of P. More details can be found in (Caragiannis, Hemaspaandra, and Hemaspaandra 2016).

We conjecture that the *worst-from-subset* and *between* information structures correspond respectively to the Young and Dodgson rules. In both cases, the sufficiency proof is easy, but for the necessity proof we have spent a considerable amount of time and energy trying to prove or disprove it, without any success so far.

7 Discussion

7.1 Summary of Contributions

On the conceptual side, our characterization of implementable information structures (or equivalently, equivalence relations) helps understanding better the informational requirements of voting rules and provides a way of classifying them according to the amount of information they need.

On the more practical side, we have identified the compilation functions of common voting rules, spanning over a number of interesting and natural information structures that were not known to correspond to common voting rules before. We have thus identified a rule with maximal anonymous compilation (the modal rule), an onto rule with minimal anonymous compilation (the omninomination rule), one whose compilation is the rank distribution induced from the profile (lexicographic plurality). In addition, we have identified other common rules with somewhat 'exotic' compilation structures (*e.g* minimax rank), and stated a conjecture about the Young and Dodgson rules.

7.2 Further Work

Beyond Dodgson and Young, there are only a few common single-winner voting rules whose corresponding information structure has not been identified yet. We can think of 'hybrid' rules such as Nanson and Baldwin, but not many more.

We may also think of extending our results to multiwinner extension of the rules we considered. For many rules,

the compilation structure remains the same for the immediate multiwinner extension (e.g., best k Borda scores, last remaining k candidates with STV); however, for the multiwinner extension of Bucklin and minmaxrank, the compilation structure is richer. Finally, many multiwinner voting rules are not of the "best-k" type, and these need to be studied separately (see (Karia and Lang 2024) for some preliminary results).

The focus of our paper was on the identification of compiled information structures associated with voting rules. We did not address their *compilation complexity* (which is equal to the logarithm of the number of equivalence classes of the compiled information structure); we leave it for follow-up work.

7.3 Contextual Equivalence

We recall that one of the primary motivations for voting compilation is to give a meaningful definition of what we *need to know* from a profile for being able to apply a voting rule. We argued that it is nontrivial to give a perfect definition, and as far as we know, compilation seems the closest to do the job. Still, one could think of at least another definitions. Recall our introductory discussion about contextualizing (that is, uncertainty about which state of the world will materialize). Instead of bearing on unknown votes, another meaningful type of uncertainty could be the candidates running from the election: X is the set of all candidates that may run, and the actual set of candidates running will be $C \subseteq X$. For a profile P, and $C \subseteq X$, let $P^{\downarrow C}$ is the restriction of P to the candidates in C. Then P and Q are f-equivalent if for any $C \subseteq X$, $f(P^{\downarrow C}) = f(Q^{\downarrow C})$.[6]

This leads to totally different compilation structures: for instance, the majority graph is enough to compile any rule based on the majority graph, but it is not necessary, because if we have a Condorcet winner, then this information is enough; for plurality, we need the whole anonymous profile. Given that the notions are so different, we don't expect our characterisation results extend to this setting.

Contextual equivalence appears in other settings, especially in the knowledge representation community. Recall that in our setting, two profiles P and Q are equivalent for a voting rule if for any profile R, $P + R$ and $Q + R$ have the same winner. Now, two logic programs P and Q are strongly equivalent if for any logic program R, $P \cup R$ and $Q \cup R$ have the same answer sets ((Lifschitz, Pearce, and Valverde 2001; Turner 2001), as well as (Eiter, Fink, and Woltran 2007) for a recent overview). This notion of strong equivalence has been generalized to many other knowledge representation formalisms (Baumann and Strass 2022) and in particular to logic programs with preferences (Faber, Tompits, and Woltran 2008; Faber, Truszczynski, and Woltran 2013). This connection is certainly worth investigating.[7]

[6]We could also think of considering *adding* candidates, but this leads to many difficulties (Chevaleyre et al. 2011).

[7]We thank the metareviewer for pointing out this connection to us.

Acknowledgments

This work has been supported by a grant from the French State managed by the National Research Agency (ANR) through the France 2030 program, with the reference ANR-23-IACL-0008 (PR[AI]RIE-PSAI), and through the CONDORCET project with reference ANR-24-EXMA-0001. We thank all reviewers (and the metareviewer) for their useful comments.

AI Declaration

The authors have not employed any Generative AI tools.

References

ACE Electoral Knowledge Network. 2026. Section: Vote counting / consolidating voting results. https://aceproject.org/main/english/vc/vcb05.htm.

Baumann, R., and Strass, H. 2022. An abstract, logical approach to characterizing strong equivalence in non-monotonic knowledge representation formalisms. *Artif. Intell.* 305:103680.

Cadoli, M., and Donini, F. M. 1997. A survey on knowledge compilation. *AI Communications* 10(3-4):137–150.

Caragiannis, I.; Hemaspaandra, E.; and Hemaspaandra, L. A. 2016. Dodgson's rule and young's rule. In Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A. D., eds., *Handbook of Computational Social Choice*. Cambridge University Press. 103–126.

Caragiannis, I.; Procaccia, A. D.; and Shah, N. 2014. Modal ranking: A uniquely robust voting rule. In Brodley, C. E., and Stone, P., eds., *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, 616–622. AAAI Press.

Chevaleyre, Y.; Lang, J.; Maudet, N.; and Ravilly-Abadie, G. 2009. Compiling the votes of a subelectorate. In Boutilier, C., ed., *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, 97–102.

Chevaleyre, Y.; Lang, J.; Maudet, N.; and Monnot, J. 2011. Compilation and communication protocols for voting rules with a dynamic set of candidates. In Apt, K. R., ed., *Proceedings of the 13th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-2011), Groningen, The Netherlands, July 12-14, 2011*, 153–160. ACM.

Congar, R., and Merlin, V. 2012. A characterization of the maximin rule in the context of voting. *Theory and Decision* 72:131—-147.

Conitzer, V., and Sandholm, T. 2005. Communication complexity of common voting rules. In Riedl, J.; Kearns, M. J.; and Reiter, M. K., eds., *Proceedings 6th ACM Conference on Electronic Commerce (EC-2005), Vancouver, BC, Canada, June 5-8, 2005*, 78–87. ACM.

Eiter, T.; Fink, M.; and Woltran, S. 2007. Semantical characterizations and complexity of equivalences in answer set programming. *ACM Trans. Comput. Log.* 8(3):17.

Faber, W.; Tompits, H.; and Woltran, S. 2008. Notions of strong equivalence for logic programs with ordered disjunction. In Brewka, G., and Lang, J., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, September 16-19, 2008*, 433–443. AAAI Press.

Faber, W.; Truszczynski, M.; and Woltran, S. 2013. Strong equivalence of qualitative optimization problems. *J. Artif. Intell. Res.* 47:351–391.

Filtser, A., and Talmon, N. 2019. Distributed monitoring of election winners. *Artificial Intelligence* 276:79–104.

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

Gärdenfors, P. 1976. Manipulation of social choice functions. *Journal of Economic Theory* 13(2):217–228.

Goldsmith, J.; Lang, J.; Mattei, N.; and Perny, P. 2014. Voting with rank dependent scoring rules. In Brodley, C. E., and Stone, P., eds., *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, 698–704. AAAI Press.

Karia, N., and Lang, J. 2024. Compiling the votes of a subelectorate for multi-winner voting rules. In Freeman, R., and Mattei, N., eds., *Algorithmic Decision Theory - 8th International Conference, ADT 2024, New Brunswick, NJ, USA, October 14-16, 2024, Proceedings*, volume 15248 of *Lecture Notes in Computer Science*, 18–32. Springer.

Lifschitz, V.; Pearce, D.; and Valverde, A. 2001. Strongly equivalent logic programs. *ACM Trans. Comput. Log.* 2(4):526–541.

Peters, D.; Procaccia, A. D.; Psomas, A.; and Zhou, Z. 2020. Explainable voting. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Postlewaite, A., and Schmeidler, D. 1986. Strategic behaviour and a notion of ex ante efficiency in a voting model. *Social Choice and Welfare* 3(1):37–49.

Sato, S. 2009. Informational requirements of social choice rules. *Math. Soc. Sci.* 57:188–198.

Sato, S. 2016. Informational requirements of social choice rules to avoid the condorcet loser: A characterization of the plurality with a runoff. *Math. Soc. Sci.* 79:11–19.

Turner, H. 2001. Strong equivalence for logic programs and default theories (made easy). In Eiter, T.; Faber, W.; and Truszczynski, M., eds., *Logic Programming and Nonmonotonic Reasoning, 6th International Conference, LPNMR 2001, Vienna, Austria, September 17-19, 2001, Proceedings*, Lecture Notes in Computer Science, 81–92. Springer.

Xia, L., and Conitzer, V. 2009. Finite local consistency characterizes generalized scoring rules. In Boutilier, C., ed., *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, 336–341.

Xia, L., and Conitzer, V. 2010. Compilation complexity of common voting rules. In Fox, M., and Poole, D., eds., *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 915–920. AAAI Press.

Zwicker, W. S. 2016. Introduction to the theory of voting. In Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A. D., eds., *Handbook of Computational Social Choice*. Cambridge University Press. 23–56.