Verified Construction of Fair Voting Rules

Karsten Diekhoff, Michael Kirsten[0000–0001–9816–1504], and Jonas Krämer

Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
`kirsten@kit.edu`, `{karsten.diekhoff,jonas.kraemer}@student.kit.edu`

Abstract. Voting rules aggregate multiple individual preferences in order to make collective decisions. Commonly, these mechanisms are expected to respect a multitude of different fairness and reliability properties, e.g., to ensure that each voter's ballot accounts for the same proportion of the elected alternatives, or that a voter cannot change the election outcome in her favor by insincerely filling out her ballot. However, no voting rule is fair in all respects, and trade-off attempts between such properties often bring out inconsistencies, which makes the construction of arguably practical and fair voting rules non-trivial and error-prone. In this paper, we present a formal and systematic approach for the flexible and verified construction of voting rules from composable core modules to respect such properties by construction. Formal composition rules guarantee resulting properties from properties of the individual components, which are of generic nature to be reused for various voting rules. We provide a prototypical logic-based implementation with proofs for a selected set of structures and composition rules within the theorem prover Isabelle/HOL. The approach can be readily extended in order to support many voting rules from the literature by extending the set of basic modules and composition rules. We exemplarily construct the well-known voting rule *sequential majority comparison* (SMC) from simple generic modules, and automatically produce a formal proof that SMC satisfies the fairness property monotonicity. Monotonicity is a well-known social-choice property that is easily violated by voting rules in practice.

Keywords: Social choice · Higher-order logic · Modular verification.

1 Introduction

In an election, voters cast ballots to express their individual preferences about eligible alternatives. From these individual preferences, a collective decision, i.e., a set of winning alternatives, is determined using a *voting rule.* Throughout the literature and in practice, there are many different voting rules each of which exhibit different behaviors and properties. Depending on the specific applications and regulations, voting rules are devised for a variety of different design goals towards carefully selected behaviors and properties. Imagine, for instance, one situation where a village wants to elect a local council, or another one where a group of friends wants to choose a destination to go on vacation based on each of the friend's preferences. In the former case, the village might prefer to be represented by a larger council rather than having only few representatives who are elected by a majority, but are strongly disliked by everybody else. For the latter, it is clearly undesirable to choose multiple destinations, but rather settle for one so that the group can spend the vacation together.

Indeed, there is no general rule which caters for every requirement, and any voting rule shows paradoxical behavior for some voting situation [1]. Therefore, an approach to analyze voting rules for their behavior by clear-cut formal properties, the so-called *axiomatic method*, has emerged. The axiomatic method advocates the use of rules that provide rigorous guarantees (which we call *properties*), and compares them based on guarantees that they do, or do not, satisfy. These properties capture very different requirements of fairness or reliability, e.g., principles that each vote is counted equally, that the electees proportionally reflect the voters' preferences, or that they are preferred by a majority of the voters. Devising voting rules towards such properties is generally cumbersome as their trade-off is inherently difficult and error-prone. Attempting to prove properties for specific voting rules often exhibits design errors, but is cumbersome as well [4]. As of yet, there exists no general formal approach to systematically devise voting rules towards formal properties without being either error-prone or extremely cumbersome.

Contribution. In this paper, we present a formal systematic approach for the flexible and verified construction of voting rules from compact composable modules with guaranteed formal properties. Indeed, when taking an abstract view, many voting rules share similar structures, e.g., aggregating the individual votes by calculating the sum or some other aggregator function. Based on this observation, our approach enables flexible, intuitive and verified construction of interesting voting rules from a small number of compositional structures. These structures exhibit precise and general interfaces such that their scope may easily be extended with further modules. We devise a general component type as well as special types, e.g., for aggregation functions, and compositional structures, e.g., for sequential, parallel and loop composition. The resulting properties, e.g., common social choice properties from the literature, are guaranteed from composing modules with given individual properties by rigorous composition rules.

We demonstrate the logic-based application with proofs for a selected set of composition structures and rules, and composable modules within the theorem prover Isabelle/HOL [13]. Thereby, the approach is amenable both for external scrutiny as compositions are rigorously and compactly defined, and for an integration in larger automatic voting rule design or verification frameworks. As case study, we define composition rules for the common social choice property *monotonicity*, and demonstrate a formal correct-by-construction verification of the rule *sequential majority comparison* (SMC). The construction produces a proof that SMC fulfills the monotonicity property using a set of basic modules.

Outline. The rest of this paper is structured as follows: Section 2 introduces formal concepts and definitions from social choice theory for our construction approach. We present the core framework in Section 3 and demonstrate the verified construction framework within Isabelle/HOL in Section 4. In Section 5, we apply our approach to the case study of constructing the monotone voting rule sequential majority comparison, and give an overview of related work to our approach in Section 6. We finally conclude and discuss future work in Section 7.

2 Concepts and Definitions from Social Choice Theory

We consider a fixed finite set $\mathcal{A}$ of eligible *alternatives* and a finite (possibly ordered) set $\mathscr{K}$ of *voters* (with cardinality k). In an election, each voter i casts a *ballot* $\succsim_i \in \mathcal{L}(\mathcal{A})$, which is a linear order[1], ranking the alternatives $\mathcal{A}$ according to i's preference. We collect all votes in a *profile*, i.e., a sequence $\underset{\sim}{\succsim} = (\succsim_1, \ldots, \succsim_k)$ of k ballots. Given the set $\mathcal{L}(\mathcal{A})$ of linear orders on $\mathcal{A}$, $\mathcal{L}(\mathcal{A})^k$ defines the set of all profiles on $\mathcal{A}$ of length k, i.e., for all voters. Hence, we have $\mathcal{L}(\mathcal{A})^+ = \bigcup_{k \in \mathbb{N}^+} \mathcal{L}(\mathcal{A})^k$, the set of all finite, nonempty profiles on $\mathcal{A}$, i.e., the input domain for a voting rule. Voting rules (see Definition 1) elect a nonempty subset $\mathcal{C}(\mathcal{A})$ of the alternatives as (possibly tied) winners, given $\mathcal{C}(X)$ denotes the set of all nonempty subsets of a set X.

Definition 1 (Voting Rule). *Given a finite set of alternatives $\mathcal{A}$, a* voting rule *f maps each possible profile $\underset{\sim}{\succsim} \in \mathcal{L}(\mathcal{A})^+$ to a nonempty set of winning alternatives in $\mathcal{C}(\mathcal{A})$:*

$$f : \mathcal{L}(\mathcal{A})^+ \to \mathcal{C}(\mathcal{A}).$$

In practice and in literature, a multitude of voting rules are in use. A common example is the function that returns all alternatives that are ranked at first position by a plurality of the voters, hence called *plurality voting*. Another common kind of voting rules assigns values for every ballot to each alternative according to her position occupied on the ballot, and elects the alternatives with the maximal score, i.e., the sum of all such values for her. Such rules are called scoring rules, e.g., the *Borda rule*, where the value of an alternative on a ballot is the number of alternatives ranked below her on that ballot.

Social Choice Properties. Within social choice theory, the axiomatic method has established a number of general fairness and reliability properties called (*axiomatic*) *social choice properties*. They formally capture intuitively desirable or in other ways useful properties to compare, evaluate, or characterize voting rules. Such properties are applicable in a general way as they are defined on abstract voting rules only with respect to profiles and returned sets of winning alternatives. For the sake of simplicity, the examples illustrated in this paper only address properties of universal nature, i.e., they require that all mappings of a given voting rule belong to some set of admissible ways, as formally described by the property of interest, for associating sets of winners to profiles. Besides properties which *functionally* limit the possible sets of winning alternatives for

[1] A linear order is a transitive, complete, and antisymmetric relation.

any one given profile, properties may also *relationally* limit combinations (of finite arity) of mappings, e.g., certain (hypothetical) changes of a profile may only lead to certain changes of the winning alternatives. Relational properties capture a voter's considerations such as how certain ways of (not) filling out her ballot may or may not affect the chances of winning for some alternatives.

Within this paper, we use *Condorcet consistency* and *monotonicity* as running examples. The functional property *Condorcet consistency* (see Definition 3) requires that if there is an alternative w that is the *Condorcet winner* (see Definition 2), the rule elects w as unique winner. A Condorcet winner is an alternative that wins every pairwise majority comparison against all other alternatives, i.e., for any other alternative, there is a majority of voters who rank the Condorcet winner higher than that alternative.

Definition 2 (Condorcet Winner). *For a set of alternatives $\mathcal{A}$ and a profile $\succsim \in \mathcal{L}(\mathcal{A})^+$, an alternative $w \in \mathcal{A}$ is a* Condorcet winner *iff the following holds:*

$$\forall a \in \mathcal{A} \setminus \{w\} : |\{i \in \mathscr{K} : a \succsim_i w\}| < |\{i \in \mathscr{K} : w \succsim_i a\}|.$$

Note that, if a Condorcet winner exists, it is unique by the above definition.

Definition 3 (Condorcet Consistency). *For a set of alternatives $\mathcal{A}$, a voting rule f is* Condorcet consistent *iff for every profile $\succsim \in \mathcal{L}(\mathcal{A})^+$ and (if existing) the respective Condorcet winner $w \in \mathcal{A}$, the following holds:*

$$w \textit{ is } \text{Condorcet winner } \textit{for} \succsim \quad \Rightarrow \quad f(\succsim) = \{w\}.$$

Note here that for profiles for which no Condorcet winner exists, the property imposes no requirements on the election outcome. The relational property *monotonicity* expresses that if a voter were to change her vote in favor of some other alternative, the outcome could never change to the disadvantage of that alternative. Monotone voting rules are resistant to some forms of strategic manipulation where a voter could make their preferred alternative the (unique) winner by misrepresenting her actual preferences and assigning a higher rank to another alternative on her ballot. A voting rule is monotone (see Definition 5) iff for any two profiles $\succsim$ and $\succsim'$ which are identical except for one alternative a that is ranked higher in $\succsim'$ (while preserving all remaining pairwise-relative rankings), the election of a for $\succsim$ always implies her election for $\succsim'$. We define this "ranking higher" as *lifting* an alternative (see Definition 4).

Definition 4 (Lifting). *For a set of alternatives $\mathcal{A}$ and two profiles $\succsim, \succsim' \in \mathcal{L}(A)^k$, $\succsim'$ is obtained from $\succsim$ by* lifting *an alternative $a \in \mathcal{A}$ iff there exists a ballot $i \in [1, k]$ such that $\succsim_i \neq \succsim'_i$ and for each such i the following holds:*

i. There exists some alternative $x \in \mathcal{A}$ such that $x \succsim_i a$ and $a \succsim'_i x$, and
ii. we have $y \succsim_i z \Leftrightarrow y \succsim'_i z$ for all other alternatives $y, z \in \mathcal{A} \setminus \{a\}$.

We may thus define the monotonicity property as follows.

Definition 5 (Monotonicity). *For a set of alternatives $\mathcal{A}$ and an alternative $a \in \mathcal{A}$, a voting rule f is* monotone *iff for all profiles $\succsim, \succsim' \in \mathcal{L}(\mathcal{A})^+$ where $\succsim'$ is obtained from lifting a in $\succsim$, the following implication holds:*

$$a \in f(\succsim) \Rightarrow a \in f(\succsim').$$

3 Composable Modules and Compositional Structures

The verified construction approach consists of two structural and two semantic concepts, namely (i) *component types* that specify structural interfaces wherein components can be implemented, and (ii) *compositional structures* that specify structural contracts which combine components to create new components that are again composable. Moreover, semantic aspects for constructing concrete voting rules are addressed by (iii) *composition rules* that define semantic rules which compositions can contractually depend on, i.e., if components fulfill a rule's requirements, the composition guarantees the rule's semantics, as well as (iv) *composable modules* that define concrete semantics of either directly implemented or constructed modules from which other modules can be composed using the composition rules. In the following, we give details on component types and compositional structures for composing voting rules based on the ideas in [9].

3.1 Electoral Modules

The structural foundation of our approach are *electoral modules*, a generalization of voting rules as in Definition 1. We define electoral modules (see Definition 6) so that they act as the principal component type (cf. (i)) within our framework. In contrast to a voting rule, an electoral module does not need to make final decisions for all the alternatives, i.e., partition[2] them (only) into winning and losing alternatives, but can instead defer the decision for some or all of them to other modules. Hence, electoral modules partition the received (possibly empty) set of alternatives $A \subseteq \mathcal{A}$ into *elected*, *rejected* and *deferred* alternatives. In particular, any of those sets, e.g., the set of winning (elected) alternatives, may also be left empty, as long as they collectively still hold all the received alternatives. Just like a voting rule, an electoral module also receives a profile which holds the voters' preferences, which, unlike a voting rule, consider only the (sub-)set of alternatives that the module receives. We take this into account by the following definition of our input domain $\mathcal{D}^{\mathcal{A}}_{\text{mod}}$:

$$\mathcal{D}^{\mathcal{A}}_{\text{mod}} := \{(A, \succsim) \mid A \subseteq \mathcal{A}, \succsim\, \in \mathcal{L}(A)^+\}$$

$\mathcal{D}^{\mathcal{A}}_{\text{mod}}$ contains all subsets of $\mathcal{A}$ paired with matching profiles. We can hence define electoral modules as follows:

Definition 6 (Electoral Module). *For eligible alternatives $\mathcal{A}$ and a (sub-)set $A \subseteq \mathcal{A}$, we define an* electoral module *as a function m with*

$$m : \mathcal{D}^{\mathcal{A}}_{\text{mod}} \to \mathcal{P}(\mathcal{A})^3.$$

The function m maps a set of alternatives with a matching profile to the set-triple (e, r, d) of sets of elected (e), rejected (r), *and* deferred (d) *alternatives such that*

$$(A, \succsim) \in \mathcal{D}^{\mathcal{A}}_{\text{mod}} \Rightarrow (m(A, \succsim) = (e, r, d) \textit{ partitions } A).$$

[2] We say that a sequence of sets $s_1, \ldots, s_n$ partitions a set S if and only if S equals the union $\bigcup_{i \in [1,n]} s_i$ over all sets s_i for $i \in [1, n]$ and all their pairwise intersections are empty, i.e., $\forall i \neq j \in [1, n] : s_i \cap s_j = \emptyset$.

In the following, we denote the set of electoral modules by $\mathcal{M}_{\mathcal{A}}$, as well as $m_e(A, \succsim)$, $m_r(A, \succsim)$, and $m_d(A, \succsim)$ for the elected, rejected and deferred alternatives, respectively, of an electoral module m for $(A, \succsim)$.

Moreover, we can easily translate voting rules to electoral modules by returning a triple of empty sets in case the module receives an empty set. Otherwise, we return an empty deferred set, an elected set with exactly the winning alternatives, and a rejected set with the complement of the winning alternatives (we remove the alternatives which are not contained in the received set of alternatives). Note that, as a consequence, social choice properties can also be easily translated in order to conform to electoral modules.

3.2 Sequential Composition

Sequential composition (see Definition 7) is a compositional structure (cf. (ii)) for composing two electoral modules m, n into a new electoral module $(m \triangleright n)$ such that the second module n only decides on alternatives which m defers and cannot reduce the set of alternatives already elected or rejected by m. In this composition, n receives only m's deferred alternatives $m_d(A, \succsim)$ and a profile $\succsim_{|(m_d(A,\succsim))}$ which only addresses alternatives contained in $m_d(A, \succsim)$.

Definition 7 (Sequential Composition). *For any set of alternatives $\mathcal{A}$ and a (sub-)set $A \subseteq \mathcal{A}$, electoral modules $m, n \in \mathcal{M}_{\mathcal{A}}$ and input $(A, \succsim) \in \mathcal{D}^{\mathcal{A}}_{\text{mod}}$, we define the sequential composition function $(\triangleright) : \mathcal{M}^2_{\mathcal{A}} \to \mathcal{M}_{\mathcal{A}}$ as*

$$\begin{aligned}(m \triangleright n)(A, \succsim) := \quad & (m_e(A, \succsim) \cup n_e(m_d(A, \succsim), \succsim_{|(m_d(A,\succsim))}), \\ & m_r(A, \succsim) \cup n_r(m_d(A, \succsim), \succsim_{|(m_d(A,\succsim))}), \\ & n_d(m_d(A, \succsim), \succsim_{|(m_d(A,\succsim))}))\end{aligned}$$

3.3 Revision Composition

Mostly for convenience, we define a revision composition (see Definition 8) for situations in which we want to revise the alternatives already elected by a prior module, e.g., for enabling sequential composition with a tie-breaking module. For an electoral module m, the revision composition removes m's elected alternatives and attaches them to the previous deferred alternatives, while the rejected alternatives are kept unchanged. Whereas this composition can also be achieved by parallel composition, this dedicated structure turns out to be beneficial in our implementation due to its frequent uses.

Definition 8 (Revision Composition). *For any set of alternatives $\mathcal{A}$ and a (sub-)set $A \subseteq \mathcal{A}$, electoral module $m \in \mathcal{M}_{\mathcal{A}}$, and input $(A, \succsim) \in \mathcal{D}^{\mathcal{A}}_{\text{mod}}$, we define the revision composition $(\downarrow) : \mathcal{M}_{\mathcal{A}} \to \mathcal{M}_{\mathcal{A}}$ as*

$$(m\!\downarrow)(A, \succsim) := (\emptyset,\ m_r(A, \succsim),\ m_e(A, \succsim) \cup m_d(A, \succsim)).$$

3.4 Parallel Composition

The parallel composition (see Definition 11) lets two electoral modules make two independent decisions for the given set of alternatives. Their two decisions are then aggregated by an *aggregator* (see Definition 9), which is another component type that combines two set-triples of elected, rejected and deferred alternatives (as well as the set of alternatives) into a single such triple (we define the input domain $\mathcal{D}_{\text{agg}}^{\mathcal{A}}$ accordingly).

Definition 9 (Aggregator). *For a set of alternatives $\mathcal{A}$, a (sub-)set $A \subseteq \mathcal{A}$ and input $(A, p_1, p_2) \in \mathcal{D}_{\text{agg}}^{\mathcal{A}}$, an aggregator is a function*

$$agg : \mathcal{D}_{\text{agg}}^{\mathcal{A}} \to \mathcal{P}(\mathcal{A})^3 \text{ such that } agg(A, p_1, p_2) \text{ partitions } A.$$

A useful instance of such an aggregator is the max-aggregator $agg_{\max}$:

Definition 10 (Max-Aggregator). *Given two set-triples (e_1, r_1, d_1), (e_2, r_2, d_2) of elected (e), rejected (r) and deferred (d) alternatives, $agg_{\max}$ picks, for each alternative a and the sets containing a, the superior one of the two sets (assuming the order* $\mathrm{e} > \mathrm{d} > \mathrm{r}$*).*

$$\begin{aligned} agg_{\max}((e_1, r_1, d_1),\ (e_2, r_2, d_2)) = \\ (e_1 \cup e_2, (r_1 \cup r_2) \setminus (e_1 \cup e_2 \cup d_1 \cup d_2), (d_1 \cup d_2) \setminus (e_1 \cup e_2)) \end{aligned}$$

Based on the notion of aggregators, we can now define the parallel composition as a function mapping two electoral modules m, n and an aggregator $agg \in \mathcal{G}_{\mathcal{A}}$ (the set of all aggregators) to a new electoral module $(m \ ||_{agg}\ n)$:

Definition 11 (Parallel Composition). *For a set of alternatives A and a (sub-)set $A \subseteq \mathcal{A}$, electoral modules m, n, and an aggregator agg we define the parallel composition $(||) : (\mathcal{M}_{\mathcal{A}} \times \mathcal{G}_{\mathcal{A}} \times \mathcal{M}_{\mathcal{A}}) \to \mathcal{M}_{\mathcal{A}}$ as*

$$(m \ ||_{agg}\ n)(A, \succsim) := agg(A,\ m(A, \succsim),\ n(A, \succsim)).$$

3.5 Loop Composition

Based on sequential composition (Section 3.2) for electoral modules, we define the more general loop composition for sequential compositions of dynamic length. A loop composition $(m \circlearrowleft_t)$ repeatedly composes an electoral module m sequentially with itself until either a fixed point is reached or a *termination condition* t is satisfied. Within our framework, termination conditions, technically another component type, are boolean predicates on set-triples such that they are suitable for electoral modules. The full definition can be found within the Isabelle/HOL theories provided with this paper.

3.6 A Simple Example

As a simple example, we illustrate the construction of a voting rule using structures from above. Consider the well-known *Baldwin's rule*, which is a voting rule based on sequential elimination [2]. The rule repeatedly eliminates the alternative with the lowest Borda score (see Section 2) until only one alternative remains.

As basic modules (cf. (iv)), we use (a) a module that computes the Borda scores, rejects the alternative with the lowest such score, and defers the rest, as well as (b) a module that attaches all deferred alternatives to the elected set.

Moreover, we choose a termination condition such that the loop of interest stops when the set of (deferred) alternatives has reached size one.

Therefore, Baldwin's rule can be obtained by

1. composing (a) by a loop structure with above mentioned termination condition, and
2. sequentially composing the loop composition with (b).

Moreover, loop composition can be directly used for many voting rules of a category called *tournament solutions*. Tournament solutions typically consist of multiple rounds, in each comparing a pair of alternatives based on their profile rankings, and the winner of a comparison advances to the next round.

4 Verified Modular Construction Framework

In the following, we describe how we model the concepts defined in Section 3 within a modular proof framework for the verified construction of voting rules.

4.1 Isabelle and Higher-Order Logic (HOL)

We implemented and proved our logical concepts within the interactive theorem prover Isabelle/HOL [13]. The Isabelle/HOL system provides a generic infrastructure for implementing deductive systems in higher-order logics and enabling to write tactics for human-readable and machine-checked proofs to show that the deductive conclusions are indeed correct. We decided to use Isabelle/HOL, because higher-order logic (HOL) allows to define very expressive, rigorous and general theorems. By this means, a theorem is –once proven correct within Isabelle– re-checked and confirmed by Isabelle/HOL within a few seconds every time the theorem is loaded. Proofs within Isabelle/HOL are based on the employed theories at the core of the Isabelle system. We made use of the possibility to define very general theorems to be reused for the construction of various voting rules and sorts of composition. Moreover, the framework allows for easy application and extension potentially within a larger framework for the automatic discovery and construction of voting rules, provided that the voting rule of interest can be composed from the given compositional rules and composable modules using the given composition structures and component types.

The definitions and theorems within our framework are mostly self-contained, i.e., for the most part they only rely on basic set theory as well as the theories of finite lists, relations, and order relations for defining the profiles and linear orders used within our notion of profiles and modules as seen in Listing 1. Therein, we introduce a handy type abbreviation (Line 1) for profiles which are lists of relations and therefrom define profiles on alternatives (Lines 3 to 4) based on the theory of order relations, and moreover finite profiles (Lines 5 to 6) which we use in a number of structures and concepts. Moreover, type abbreviations for results of electoral modules (Line 8), i.e., set-triples, are introduced, and electoral modules (Line 10) as defined in Section 3. We capture the partitioning with the two functions to express disjointness of the three sets in an electoral module result (Lines 20 to 21) and that their union yields the set of alternatives of the input (Lines 17 to 18). Finally, at the end of Listing 1, we can essentially define electoral modules on finite profiles and partitionings of the alternatives (Lines 23 to 25). We did not require any additional theories besides the ones provided off-the-shelf with the Isabelle system.

```
1  type_synonym 'a Profile = "('a rel) list"
2
3  definition profile_on :: "'a set ⇒ 'a Profile ⇒ bool" where
4    "profile_on A p ≡ (∀ b ∈ (set p). linear_order_on A b)"
5  abbreviation finite_profile :: "'a set ⇒ 'a Profile ⇒ bool" where
6  "finite_profile A p ≡ finite A ∧ profile_on A p"
7
8  type_synonym 'a Result = "'a set * 'a set * 'a set"
9
10 type_synonym 'a Electoral_module = "'a set ⇒ 'a Profile ⇒ 'a Result"
11
12 fun disjoint :: "'a Result ⇒ bool" where "disjoint (e, r, d) =
13   ((e ∩ r = {}) ∧
14    (e ∩ d = {}) ∧
15    (r ∩ d = {}))"
16
17 fun unify_to :: "'a set ⇒'a Result ⇒ bool"
18   where "unify_to A (e, r, d) ↔ (e ∪ r ∪ d = A)"
19
20 definition partition_of :: "'a set ⇒ 'a Result ⇒ bool" where
21   "partition_of A result  ≡ disjoint result ∧ unify_to A result"
22
23 definition electoral_module :: " 'a Electoral_module ⇒ bool"
24   where "electoral_module m ≡
25   ∀A p. finite_profile A p → partition_of A (m A p)"
```

Listing 1. Central Isabelle/HOL definitions for electoral modules.

As of yet, our verified construction framework comprises concepts and proofs for 18 composition rules with ten reusable auxiliary properties and eight properties which translate directly to common social choice properties from the literature. Thereof, we implemented the auxiliary properties and the monotonicity property with respective proofs within the Isabelle/HOL framework.

4.2 Verified Construction based on Composition Rules

From devising composition rules and properties as described in the beginning of this section together with the component types and structures as described in Section 3, our framework now only requires a small set of basic components in order to construct interesting voting rules for the desired social choice properties which have been defined as properties and included in the rules for composing electoral modules. The power of our approach lies both in the generality of the composition rules and compositional structures such that various voting rules may be constructed for various properties, and in the reduction of complexity such that compositions for complex social choice properties can be defined by predominantly local composition rules in a step-by-step manner.

In general, the verified construction using composition rules works as follows: When we want to obtain a voting rule with a set of properties p from some basic components c and d which satisfy sets of properties p_c and p_d respectively, we might make use of a compositional structure X which guarantees that a composed module $m_c X m_d$ satisfies the properties p. Hence, we can get a desired voting rule by instantiating m_c and m_d by c and d respectively, which gives us the induced voting rule f_{cXd}. Note that, when we specify a set of target properties p, any voting rule induced by our framework (if a suitable one can be induced) from a set of basic components and compositional structures, necessarily comes with an Isabelle proof which establishes the validity of p for the induced voting rule. By design, these proofs are short and can in most cases be automatically inferred. Hence, given the soundness of the Isabelle/HOL theorem prover, we obtain a formal proof that the resulting voting rule indeed satisfies the required properties without the need to re-check the obtained rule.

Example. One such example using structures from Section 3 and properties defined in this section is that p consists of the property Condorcet consistency, X is the sequential composition, p_c also consists of Condorcet consistency, and p_d is empty. Thus, we have no requirements for properties of any component d, since sequential composition cannot revoke any alternatives that are already elected. If a Condorcet winner exists, this alternative is already elected by the first module, and if not, Condorcet consistency trivially holds. On its own, this composition rule might not be very sensible, but may be used in combination with other rules to preserve Condorcet consistency of composed voting rules. A voting rule from the literature which is constructed in such a manner is *Black's rule*. Black's rule is a sequential composition of (a component which induces) the Condorcet rule and (a component which induces) the Borda rule.

5 Case Study

As a case study for demonstrating the applicability of our approach to existing voting rules and the merits of composition, we constructed the voting rule *sequential majority comparison* (SMC) from the literature (e.g., from Brandt et al. [5]), thereby producing a compositional proof that the rule is monotone.

Sequential Majority Comparison (SMC). The voting rule of sequential majority comparison, also known as sequential pairwise majority, is simple enough for understanding, but still complex enough such that it demonstrates interesting properties such as monotonicity. Essentially, SMC fixes some (potentially arbitrary) order on all alternatives and then consecutively performs pairwise majority elections. We start by doing pairwise comparisons of the first and the second alternative, then compare the winner of this pairwise comparison to the third alternative, whose winner is then compared to the fourth alternative, and so on. SMC belongs to a category of voting rules called *tournament solutions*, for which we outline a possible construction pattern in the following.

Verified Construction of Tournament Solution. As indicated in Section 3, loop composition appears sensible for *tournament solutions*, as a list of alternatives is processed by multiple rounds, whereof in each, the previously chosen alternative is compared to the next alternative on the list regarding their rankings in the profile, and the winner of a comparison advances to the next round.

To compare alternatives, we use any electoral module m which elects one alternative and rejects the rest (for example via plurality voting). To limit comparisons to two alternatives, we use the electoral module $pass^2_>$, which defers the two alternatives ranked highest in some fixed order $>$ and rejects the rest. Similarly, $drop^2_>$ rejects these two alternatives and defers the rest.

We can now describe a single comparison in our tournament as

$$c = (pass^2_> \triangleright m) \;||_{agg_{\max}}\; drop^2_>$$

The first part of the parallel composition elects the winner of the current comparison and rejects all other alternatives. The second part defers all alternatives which are not currently being compared and therefore stay in the tournament.

The termination condition $t_{|d|=0}$ is satisfied iff the set of deferred alternatives passed to it is empty. Then we describe a single round of our tournament as

$$r = (c \circlearrowleft_{t_{|d|=0}}) \downarrow$$

Now, for the case of sequential majority comparison (SMC), we proceed as follows.

Verified Construction of SMC. Every single comparison elects a single alternative to advance to the next round and rejects the other. As long as alternatives are left, the next c compares the next two alternatives. If there ever is only a single alternative left, it advances to the next round automatically. At the end of the round, we need to revise to defer all winners to the next round instead of electing them.

Let m_{elect} be the electoral module which elects all alternatives passed to it and $t_{|d|=1}$ the termination condition that is satisfied when exactly one alternative is deferred. We can now define the whole tournament as

$$t = (r \circlearrowleft_{t_{|d|=1}}) \triangleright m_{\text{elect}}.$$

t repeats single rounds as long as there is more than one alternative left and then elects the single survivor.

Implementing the Construction of SMC in Isabelle/HOL. After having described a general pattern for the verified construction of tournament solutions and sequential majority comparison, we give only structural information on the implemented construction proofs for SMC as the details are rather lengthy, but instead refer the reader to the Isabelle/HOL proofs.

We can construct SMC by combining six different basic components by using all of our composition structures, i.e., the sequential, parallel, loop, and revision structure, and thereby produce a proof that SMC is a monotone voting rule.

```
definition SMC :: "'a rel ⇒ 'a Electoral_module" where
  "SMC x ≡ let a = Max_aggregator; t = Defer_eq_condition 1 in
    ((((Pass_module 2 x) ▷ ((Plurality_module↓) ▷ (Pass_module 1 x))) ||ₐ
      (Drop_module 2 x)) ↻ₜ) ▷ Elect_module"

theorem SMC_sound:
  assumes order: "linear_order x"
  shows "electoral_module (SMC x)"

theorem SMC_monotone:
  assumes order: "linear_order x"
  shows "monotone (SMC x)"
```

Listing 2. The modular construction of SMC in Isabelle/HOL.

The high-level modular construction can be seen in Listing 2, where SMC stands for sequential majority comparison composed of a number of simple components. Each component, the largest of which is an electoral module inducing plurality voting (see Section 2), consists of not more than three lines in higher-order logic and we provided proofs within our Isabelle/HOL framework for easy reuse and modification for similar voting rules.

Moreover, Listing 2 shows the simplicity of the abstract proof obligations both that SMC is again an electoral module and satisfies the monotonicity property. Both tasks are proven fully modularly and are hence a direct result of SMC's composition, and is apt for an automated integration within a potential future logic-based synthesis tool. We omit the proofs at this point, but they are available for download[3] and can be inspected and re-played for inspection and automatically checked using Isabelle/HOL. The full proof comprises 26 compositions using a set of six basic components within the theorem prover Isabelle/HOL.

6 Related Work

We base the core component type in our verified construction framework on the electoral modules from the unified description of electoral systems in [7]. Therein, Grilli di Cortona et al. devise a complex component structure for describing hierarchical electoral systems with a focus on proportional voting rules including notions for electoral districts and concepts of proportionality. Note, however, that the component type within this work is already quite different from the structures in [7]. In the current state, essentially, both concepts only share the concept of reducing and partitioning the set of alternatives.

General informal advice on voting rule design is given by Taagepera [15]. Moreover, a first approach for composing voting rules in a limited setting is given by Narodytska et al. [12] that is readily expressible by our structures. Other work designs voting rules less modularly for statistically guaranteeing social choice properties by machine learning [17]. Prior modular approaches also target verification [10, 16] or declarative combinations of voting rules [6], but ignore the social choice or fairness properties targeted by our work.

We have defined our compositional approach within Isabelle/HOL [13], a theorem prover for higher-order logic. Isabelle/HOL provides interactive theorem proving for rigorous systems design. Further work on computer-aided verification of social choice properties for voting rules using HOL4 has been done by Dawson et al. [8]. More light-weight approaches with some loss of generality, but the merit of generating counterexamples for failing properties has been devised by Beckert et al. [3] and Kirsten and Cailloux [11]. Therein, techniques for relational verification of more involved social choice properties have been applied. Another interesting approach has been followed by Pattinson and Schürmann [14], where voting rules are directly encoded into HOL rules within tactical theorem provers.

7 Conclusion

Within this work, we introduced an approach to systematically construct voting rules from compact composable modules to satisfy formal social choice properties. We devised composition rules for a selection of common social choice properties, such as monotonicity or Condorcet consistency, as well as for reusable auxiliary properties. By design, these composition rules give formal guarantees, in the form of an Isabelle proof based on the properties satisfied by the component properties, that a constructed voting rule fulfills the social choice property of interest as long as its components satisfy specific properties, which we have proved within Isabelle/HOL for the scope of our case study.

[3] https://github.com/VeriVote/verifiedVotingRuleConstruction/

Currently, the construction capabilities of our framework are not fully automatic, as in order to construct a voting rule with a specific property, the concrete assembly structure (in the form of a chain of composition structures and core modules) needs to be given. However, this can already be easily turned into a (simple) synthesis tool by a simple Prolog program. This program holds (only) a collection of which core components satisfy (as proven in Isabelle/HOL within the framework presented in this paper) which individual properties, together with the composition rules, i.e., how the composition of components requires and establishes formal properties). As a result, this program comes up with (albeit simple) proposals for concrete compositions in order to obtain a voting rule which (provenly) satisfies the requested properties, with the proofs provided by our framework within Isabelle/HOL.

Our approach is applicable to the construction of a wide range of voting rules which use sequential or parallel modular structures, notably voting rules with tie-breakers, elimination procedures, or tournament structures. This includes well-known rules such as instant-runoff voting, Nanson's method, or sequential majority comparison (SMC). We constructed SMC from simple components, which we presume to be reusable for the construction of further rules, and automatically by construction produced a proof that SMC satisfies monotonicity from basic formal proofs for the structures, compositions and components which we compositionally constructed. This case study and all required definitions were implemented and verified with the theorem prover Isabelle/HOL. Finally, our approach can be safely extended with additional modules, compositional structures, and rules, for integration into voting rule design or verification frameworks.

Outlook. So far, composition is realized mostly by transferring sets of deferred alternatives between modules. We also intend to inspect the more involved modular structures already incorporated in some more complicated voting rules, in order to achieve a more flexible notion of composition. This, however, also involves making more detailed assumptions on how exactly information is passed between modules, which might come with a loss of generality. This however, seems to be necessary for voting rules such as Single-Transferable Vote (STV), which are not composable for sensible social choice properties with our strong notion of locality in composition rules.

Moreover, it would be interesting to make use of the code generation functionality of Isabelle/HOL in order to, besides the abstract specifications of the components and composition structures, produce actual executable program code for the constructed voting rules. Furthermore, we envision an automatic synthesis tool built on top of the provided framework such that construction can be provided fully automatically. This could be done by, e.g., a Prolog program as described above, together with SMT or Horn solvers to manage larger and more complex compositions.

References

1. Arrow, K.J.: Social Choice and Individual Values. Yale University Press, third edn. (2012)
2. Baldwin, J.M.: The technique of the nanson preferential majority system of election. Transactions and Proceedings of the Royal Society of Victoria **39** (1926)
3. Beckert, B., Bormer, T., Kirsten, M., Neuber, T., Ulbrich, M.: Automated verification for functional and relational properties of voting rules. In: Sixth International Workshop on Computational Social Choice (COMSOC 2016) (2016)
4. Beckert, B., Goré, R., Schürmann, C.: Analysing vote counting algorithms via logic. In: Hutchison, D., et al. (eds.) Automated Deduction – CADE-24. LNCS, vol. 7898. Springer (2013)
5. Brandt, F., Conitzer, V., Endriss, U., Lang, J., Procaccia, A.D.: Handbook of Computational Social Choice. Cambridge University Press (2016)
6. Charwat, G.: Democratix: A declarative approach to winner determination. In: ADT. LNCS, vol. 9346. Springer (2015)
7. Grilli di Cortona, P., Manzi, C., Pennisi, A., Ricca, F., Simeone, B.: Evaluation and optimization of electoral systems. SIAM (1999)
8. Dawson, J.E., Goré, R., Meumann, T.: Machine-checked reasoning about complex voting schemes using higher-order logic. In: Haenni, R., et al. (eds.) 5th International Conference on E-Voting and Identity (VoteID 2015). LNCS, vol. 9269. Springer (2015)
9. Diekhoff, K., Kirsten, M., Krämer, J.: Formal property-oriented design of voting rules using composable modules. In: Venable, K., Pekec, S. (eds.) 6th International Conference on Algorithmic Decision Theory (ADT 2019). LNAI (2019)
10. Ghale, M.K., Goré, R., Pattinson, D., Tiwari, M.: Modular formalisation and verification of STV algorithms. In: Third International Joint Conference on Electronic Voting (E-Vote-ID 2018) (2018)
11. Kirsten, M., Cailloux, O.: Towards automatic argumentation about voting rules. In: Bringay, S., Mattioli, J. (eds.) 4ème conférence sur les Applications Pratiques de l'Intelligence Artificielle (APIA 2018), Nancy, France, July 2-6, 2018 (2018)
12. Narodytska, N., Walsh, T., Xia, L.: Combining voting rules together. In: De Raedt, L., et al. (eds.) 20th European Conference on Artificial Intelligence (ECAI 2012). vol. 242. IOS Press (2012)
13. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL - A Proof Assistant for Higher-Order Logic, LNCS, vol. 2283. Springer (2002)
14. Pattinson, D., Schürmann, C.: Vote counting as mathematical proof. In: Pfahringer, B., Renz, J. (eds.) 28th Australasian Joint Conference on Advances in Artificial Intelligence (AI 2015). LNCS, vol. 9457. Springer (2015)
15. Taagepera, R.: Designing electoral rules and waiting for an electoral system to evolve. The Architecture of Democracy: Institutional Design, Conflict Management, and Democracy in the Late Twentieth Century (2002)
16. Verity, F., Pattinson, D.: Formally verified invariants of vote counting schemes. In: Australasian Computer Science Week Multiconference (ACSW 2017). ACM (2017)
17. Xia, L.: Designing social choice mechanisms using machine learning. In: Gini, M.L., et al. (eds.) International conference on Autonomous Agents and Multi-Agent Systems (AAMAS '13). IFAAMAS (2013)