On the Specification and Verification of Voting Schemes

Bernhard Beckert[1], Rajeev Goré[2], and Carsten Schürmann[3]

[1] Karlsruhe Institute of Technology beckert@kit.edu
[2] The Australian National University Rajeev.Gore@anu.edu.au
[3] IT University of Copenhagen carsten@itu.dk

Abstract. The ability to count ballots by computers allows us to design new voting schemes that are arguably fairer than existing schemes designed for hand-counting. We argue that formal methods can and should be used to ensure that such schemes behave as intended and are conform to the desired democratic properties. Specifically, we define two semantic criteria for single transferable vote (STV) schemes, formulated in first-order logic, and show how bounded model-checking can be used to test whether these criteria are met. As a case study, we then analyse an existing voting scheme for electing the board of trustees for a major international conference and discuss its deficiencies.

1 Introduction

The goal of any social choice function is to compute an "optimal" choice from a given set of preferences. Voting schemes in elections are a prime example of such choice functions as they compute a seat distribution from a set of preferences recorded on ballots. By voting scheme we mean a concrete description of a method for counting the ballots and computing which candidates are elected – as opposed to an actual computer implementation of such a scheme or a scheme describing the process of casting votes via computer. The difficulty in designing preferential voting schemes is that the optimisation criteria are not only multi-dimensional, but multi-dimensional on more than one level. On one level, we want to satisfy each voter, so each voter is a dimension. On a higher level, there are desirable global criteria such as "majority rule" and "minority protection" that are at least partly inconsistent with each other. It is well-known that "optimising" such theoretical voting schemes along one dimension may cause them to become "sub-optimal" along another.

This observation is not new and voting specialists have proposed a series of mathematical criteria [3] that can be used to compare various voting schemes with one another. A classic example is the notion of a Condorcet winner, defined as the candidate who wins against *each* other candidate in a one-on-one contest. Such a winner exists provided that there is no cycle in the one-to-one contest relation. A voting scheme is said to satisfy the Condorcet criterion if the Condorcet winner is guaranteed to be elected when such a winner exists. Another is

the *monotonicity criterion* which requires that a candidate who wins a contest will also win if the ballots were changed uniformly to rank that candidate higher.

In practice, theoretical voting schemes are often simplified in many ways when used in real-world elections, typically to reduce their complexity to allow counting by hand. Such practical schemes may not satisfy general properties such as the Condorcet criterion simply because it is intractable to compute the Condorcet winner by hand, but they may satisfy some weaker version of "optimality" that is specific to that particular scheme. It may even happen that one among the optimal winners is chosen at random [2] (as allowed by the Australian Capital Territory's Hare-Clark Method) or that someone other than the optimal winner is elected.

Voting schemes also evolve over time – for national elections in the large, and local elections, union elections, share holder elections, and board of trustee elections in the small. Incremental changes to the electoral system, the tallying process and the related algorithms challenge the common understanding about what the voting scheme actually does. For example, since 1969 some local elections in New Zealand adopted Meeks' method [7], which is a voting scheme for preferential voting that uses fractional weightings in its computations and is too complex to count by hand. This also required an adjustment of understanding about who will now be elected. In general, it is often not clear whether changes to the electoral system improve or worsen the overall quality of a voting scheme with regard to the various dimensions of optimisation. Changes to the electoral system in Germany, for example, have created paradoxical situations where more votes for a party translate into fewer seats and fewer votes into more seats, and have prompted Germany's Supreme Court to intervene repeatedly (see, e.g., [6]).

Many jurisdictions are now using computers to count ballots according to traditional voting schemes. Using computers to count ballots opens up the possibility to use voting schemes which really are optimised along multiple dimensions, while retaining global *desiderata* such as the Condorcet criterion. The inherent complexity of counting ballots according to such schemes means that it may no longer be possible to "verify" the result by hand-counting, even when the number of ballots is small. It is therefore important to imbue these schemes with the trust accorded to existing schemes. Note that our focus is on trust in the voting scheme, not trust in the computer-based process for casting votes.

One way to engender trust in such complex yet "fairer" voting schemes is to specify the *desiderata* when the scheme is being designed, and then formally check that the scheme meets these criteria before proposing changes to the legislation to enact the scheme. Such formal analyses could contribute significant unbiased information into the political discussions that typically involve such legislative changes and also assure voters that the changes will not create paradoxical situations as described above.

Formal analysis, however, is only practicable when we possess formal specifications of the voting scheme. We argue that it is important to give declarative specifications of the properties of a voting scheme for two reasons: (1) For understanding their properties and how they change during the evolution process,

so that improving a scheme in one aspect does not by accident introduce flaws w.r.t. other aspects. (2) For checking the correctness of the scheme from both an algorithmic and implementation perspective. We also argue that general criteria are not sufficient and criteria are needed that are tailor-made for specific (classes of) voting schemes.

The properties in question are difficult to state, to formalise, to understand. to analyse, and to describe declaratively (as opposed to algorithmically) because: the final voting scheme may have to compromise between the conflicting demands of multiple individual desirable properties; the voting scheme may evolve and we may have to revisit these desiderata; even when the properties can be made mathematically precise, the resulting mathematical statement cannot serve as a specification if the electoral law defines a voting scheme that does not (always) compute the optimal solution.

Contributions Here, we show how seemingly innocuous revisions to a voting scheme can have serious implications on the desired properties of the system. As a running example, we use the preferential voting schemes single transferable vote (STV) that is used in large national elections world-wide, but also for smaller professional elections.

In Section 2, we define two tailor-made criteria to establish the desired properties of the voting scheme. Both criteria are formulated using first-order logic and are amenable for bounded model checking, which is the tool of choice for our formal analysis (Section 3). Subsequently, we discuss (Section 4) a particularly interesting variant of the Single Transferrable Vote Algorithm (CADE-STV) for the board of trustees of the International Conference on Automated Deduction (CADE). We explain its oddities and differences to standard STV, and give a historical account of the conception and the stepwise refinement of the algorithm. This paper extends our system description of a bounded model checking system for analysing voting schemes and its application to CADE-STV [1].

Related work Voting schemes have been investigated by social choice theorists for many decades. These tend to be mathematical analyses which prove various (relative) properties of different voting schemes: see [11]. Such work tends to concentrate on what we have referred to as theoretical schemes and is often couched in terms of a formal theorem and its proof in natural language.

There is also a significant body of research on various properties of vote-casting schemes, particular security properties [13].

There does not appear to be much existing work on the formal analysis of voting schemes using methods and tools from the computer aided verification and automated deduction communities in our sense, although there is some existing work on the formal analysis of actual implementations of such schemes [9, 8, 5].

2 Semantic Criteria for Analysing Voting Schemes

We focus on preferential voting schemes. Each vote consists of a partial linear order on candidates. Suppose that $\mathtt{C}$ candidates, numbered $1, 2, \ldots, \mathtt{C}$, are competing for $\mathtt{S} > 0$ vacant seats in an election. Furthermore, assume that $\mathtt{V} \geq 1$

votes have been cast and are collected in a ballot box. It is commonly agreed that for $k \leq \mathtt{C}$, a vote $[c_1, c_2, \ldots, c_k]$ ranks a subset of the candidates in decreasing order of preference so that each $c_i \in \{1, 2, \ldots, \mathtt{C}\}$ and $c_i \neq c_j$ for $i \neq j$.

2.1 Basic Criteria

Many criteria that voting schemes preferably should satisfy have been proposed (for an overview see [3]). Below, we describe a few important examples. Note that, even though these basic criteria seem obvious and indispensable for voting schemes on first sight, they are in fact not always satisfied by each reasonable voting scheme. Most real-word voting schemes violate at least some basic criteria for some possible ballot box input.

An "obvious" and widely used criterion is the *majority criterion*, which states that, if a candidate c is ranked first by a majority of voters, then c must be elected. This is indeed satisfied by all reasonable preferential voting schemes that use votes ranking candidates. However, the majority criterion can be violated by preferential voting schemes where voters can attach a numerical preference to candidates instead of just ranking them (Borda count scheme).

Another "obvious" criterion is the *monotonicity criterion* [14]. Assume that there are two ballot boxes B and B' where B' results from B by raising the preference for a candidate c in one or more of the votes and leaving the votes otherwise unchanged (i.e., a vote of the form $[c_1, \ldots, c_{i-1}, c, c_{i+1}, \ldots, c_k]$ is replaced by $[c_1, \ldots, c_{j-1}, c, c_j, \ldots, c_{i-1}, c_{i+1}, \ldots, c_k]$ $(j < i)$. The monotonicity criterion states that, if c is elected using the ballot box B, then c must also be elected using B'. Surprisingly, some real-world voting schemes – including STV – do not satisfy monotonicity [14].

A further simple criterion is the *fill-all-seats criterion*, which states that all available seats are filled provided that there are sufficient candidates, i.e., $\mathtt{C} \geq \mathtt{S}$. In practice, this criterion is often used in a restricted form, e.g., candidates can be elected only if they reach a certain minimal quota.

2.2 Criteria Characterising the Election Result

The majority criterion fully describes the election result for the simple case of a single seat and a candidate with a majority of first preferences. But we desire criteria characterising the "right" result in increasingly complex situations.

An example is the *Condorcet criterion*. A candidate c is a Condorcet winner if c wins a one-to-one comparison against all other candidates, i.e., for all $c' \neq c$ there are more voters preferring c over c' than there are voters preferring c' over c. The *Condorcet criterion* states that a Condorcet winner c must be elected if there is one. And, as long as there are open seats and there are Condorcet winners among the remaining candidates, these must also be elected.

Note that Condorcet winners do not exist for all ballot boxes, so the Condorcet criterion does not specify the election result for all situations (but only for those where a clear winner exists). Moreover, it is well known that STV (which we use as a case study) does not satisfy the Condorcet criterion.

2.3 Tailor-made Criteria for Preferential Voting Schemes

As stated previously, many more voting scheme criteria have been developed and are described in the literature. So, as a first approach to specifying and analysing a particular voting scheme, one could select some of these to characterise the scheme's properties. For a detailed analysis, however, that is not sufficient. General criteria cannot distinguish between variants of the same voting scheme (or the number of available general criteria would have to be very high). Moreover, there is a trade-off between two goals when defining voting scheme criteria:

Coverage For as many different ballot boxes as possible, the criterion should apply and restrict the number of possible election results.

Restrictiveness The number of possible election results for each ballot box should be restricted as much as possible.

For example, the majority and Condorcet criteria are very restrictive (they specify exactly one winner), but they do not have good coverage (they only apply if there is a clear winner). The fill-all-seats criterion, on the the other hand, has full coverage (it restricts the possible outcome for all ballot boxes), but it is not very restrictive.

Ideally, one would like to have an axiomatically defined criterion that allows exactly one result for every possible ballot box, i.e., has full coverage and is fully restrictive. But for many voting schemes used in practice, such criteria do not exist. In these cases, we rely on tailor-made criteria that strike a compromise between coverage and restrictiveness. For example, for our analysis of preferential voting, we have devised two tailor-made criteria that capture the essence of *preferential* voting (Criterion 2) with *proportional representation* (Criterion 1) and are applicable to our case study STV:

(1) There must be enough votes for each elected candidate.
(2) If the preferences of *all* voters w.r.t. two particular candidates are consistent, then that collective preference is not contradicted by the election result.

The first criterion only considers number of votes and ignores preferences, while the second criterion only considers preferences and ignores number of votes. This separation of the two dimensions (number of votes and preferences) is the key to finding strong criteria that can be described declaratively.

The two criteria compromise in different ways on the two goals of generality and restrictiveness: Criterion 1 has full coverage. It applies to all ballot-boxes without being too restrictive (as the order of preferences is not considered). Criterion 2 has lower coverage. It only applies if the voters' preferences are not contradictory. In that case, however, it is rather restrictive as only a small number of election results are permissible.

Criterion 2 is a weaker version of the the Condorcet criterion that, in contrast to Condorcet, is satisfied by STV. It assumes a preference to be collective if *all* voters agree (or at least not disagree), while the Condorcet criterion assumes a preference to be collective if it is supported by a *majority* of voters.

Criterion 1: Enough Votes for each Elected Candidate This criterion captures that the votes can be partitioned with an assignment of exactly one class in the partition to each elected candidate such that, if $\mathtt{Q}$ is the quota, then:

1. There are exactly $\mathtt{Q}$ votes in each class that supports an elected candidate.
2. For each vote in a class that supports a candidate, that candidate occurs somewhere among the preferences of the supporting vote.

In the second condition above, the actual order of preferences is not taken into consideration. Thus, this is a weak property that can be satisfied by a wide variety of STV variants. But it is strict in that each vote counts only once.

Example 1. Assume there are four candidates A, B, C, D for two vacant seats, the votes to be counted are $[A, B, D], [A, B, D], [A, B, D], [D, C], [C, D]$, and the quota is $\mathtt{Q} = 2$. The election result $[A, D]$ satisfies Criterion 1 using the partition $\{[A, B, D], [A, B, D]\}$, $\{[C, D], [D, C]\}$, $\{[A, B, D]\}$. The election result $[B, D]$ violates the majority criterion (as A despite its majority of first preferences is not elected). Nevertheless it satisfies Criterion 1 choosing the same partition as above (because the ordering of A and B is not considered), which shows that the criterion compromises on restrictiveness. But, the result $[A, B]$, which contradicts proportional representation, is not supported by this or any other partition (which shows that this criterion is indeed related to the requirement of proportional representation).

Formalisation. To formalise the criteria, we use first-order logic over the theories of natural numbers and arrays with the following notation in addition to the notation defined previously:

b: is the ballot box, where $b[i, j] \in \{1, \ldots, \mathtt{C}\}$ is the number of the candidate that is ranked by vote i in the j^{th} place. Thus i's preference is $[b[i, 1], b[i, 2], \ldots]$. If vote i ranks only $k \leq \mathtt{C}$ candidates, then $b[i, j] = 0$ for $k < j \leq \mathtt{C}$.

r: is the result, where $r[i]$ is the ith candidate that is elected ($1 \leq i \leq \mathtt{S}$). If less than $\mathtt{S}$ candidates are elected, then $r[i] = 0$ for the empty seats.

Our criterion is formalised by a formula ϕ in which all the above (free) variables occur. We also use an existentially quantified variable a of type array that represents the partition and the assignment of classes in the partition to elected candidates as follows:

$a[i] = k$ if the i^{th} vote supports the k^{th} elected candidate $r[k]$. If the i^{th} vote does not support any elected candidate, then $a[i] = 0$.

Then, the formula $\phi = \exists a(\phi_1 \wedge \ldots \wedge \phi_4)$ is the existentially quantified conjunction:

$$\forall i\big(1 \leq i \leq \mathtt{V} \rightarrow 0 \leq a[i] \leq \mathtt{S}\big) \tag{ϕ_1}$$

$$\forall i\big(1 \leq i \leq \mathtt{V} \rightarrow (a[i] \neq 0 \rightarrow r[a[i]] \neq 0)\big) \tag{ϕ_2}$$

$$\forall i\big((1 \leq i \leq \mathtt{V} \wedge a[i] \neq 0) \rightarrow \exists j(1 \leq j \leq \mathtt{C} \wedge b[i,j] = r[a[i]])\big) \tag{ϕ_3}$$

$$\begin{aligned}&\forall k\big((1 \leq k \leq \mathtt{S} \wedge r[k] \neq 0) \rightarrow \\ &\quad \exists \mathit{count}\big(\mathit{count}[0] = 0 \wedge \\ &\qquad \forall i(1 \leq i \leq \mathtt{V} \rightarrow (a[i] = k \rightarrow \mathit{count}[i] = \mathit{count}[i-1] + 1) \wedge \\ &\qquad\qquad (a[i] \neq k \rightarrow \mathit{count}[i] = \mathit{count}[i-1])) \wedge \\ &\qquad \mathit{count}[\mathtt{V}] = \mathtt{Q}\big)\big)\end{aligned} \tag{ϕ_4}$$

Formulae ϕ_1 and ϕ_2 express well-formedness of the partition. Formula ϕ_3 expresses that only votes can support a candidate in which that candidate is somewhere ranked. Formula ϕ_4 expresses that each class supporting a particular elected candidate has exactly $\mathtt{Q}$ elements. To formalise this, we use an array *count* such that $\mathit{count}[i]$ is the number of supporters among votes $1, \ldots, i$ that support the k^{th} elected candidate.

Note, that this criterion assumes all seats to be filled and has to be relaxed if a voting scheme does not satisfy the fill-all-seats criterion or there are not enough candidates that can reach the quota.

2.4 Criterion 2: Election Result Consistent with Preferences

The idea of our second criterion is that, if there are two candidates a, b such that in the union of all votes' preferences there is an argument for ranking a over b but no argument for ranking b over a (i.e., a and b are not part of a cycle of preferences), then b must not be ranked higher than a in the election result.

Formalisation. That there is an argument for ranking a over b means that there are candidates $a = c[0], \ldots, c[k] = b$ and there are votes $v[1], \ldots, v[k]$ such that $v[i]$ prefers $c[i-1]$ over $c[i]$ $(1 \leq i \leq k)$.

That vote $v[i]$ prefers candidate c_1 over candidate c_2 can be formalised by:

$$\begin{aligned}\phi(v, i, c_1, c_2) \;=\; &\exists j(1 \leq j \leq \mathtt{C} \wedge b[v[i], j] = c_1 \wedge \\ &\quad \forall j'(1 \leq j' < j \rightarrow b[v[i], j'] \neq c_2))\end{aligned}$$

The first line of the above formula says that voter $v[i]$ gives the preference j to candidate c_1. The second line says that v does not give a higher preference $j' < j$ to c_2, i.e., gives c_2 lower preference or no preference at all.

Now, we can formalise that there is an argument for ranking a over b by:

$$\begin{aligned}\Phi(a, b) \;=\; &\exists v \exists c \exists k(a = c[0] \wedge b = c[k] \wedge \\ &\quad \forall i(1 \leq i \leq k \rightarrow (1 \leq v[i] \leq \mathtt{V} \wedge 1 \leq c[i] \leq \mathtt{C} \wedge \\ &\qquad \phi(v, i, c[i-1], c[i]))))\end{aligned}$$

In a similar way as with ϕ, we can formalise the fact that the voting result gives a higher ranking to candidate c_1 than to candidate c_2 as follows:

$$\begin{aligned}\psi(c_1, c_2) \;=\; & \exists j(1 \leq j \leq \mathtt{C} \wedge r[j] = c_1 \wedge \\ & \quad \forall j'(1 \leq j' < j \rightarrow r[j'] \neq c_2))\end{aligned}$$

Then, using the formulas Φ and ψ the criterion can be formalised as follows:

$$\forall a \forall b\big((1 \leq a \leq \mathtt{C} \wedge 1 \leq b \leq \mathtt{C} \wedge a \neq b \wedge \Phi(a, b) \wedge \neg\Phi(b, a)) \;\rightarrow\; \neg\psi(b, a)\big)$$

2.5 Determinism

Another important criterion for voting schemes is *determinism*. Voting schemes can contain various non-determinisms that occur when candidates have the same number of votes or preferences. While that may not be a problem on an abstract level, for concrete elections it is important to clearly specify how these are to be resolved. Otherwise, choices by the election officials (or their computers) when counting the ballots could influence the election result, which is clearly undesirable.

3 Bounded Model Checking for Analysing Voting Schemes

In this section we discuss a technique for verifying that a voting scheme satisfies any of the aforementioned semantic criteria. This technique is called *bounded model checking*. It is well understood, and its application to voting schemes has been discussed in an earlier paper [1]. A bounded model checker examines an (arbitrarily small or large) finite state space of ballot boxes and tries to check if the provided semantic criteria hold for each box. If a model check run does not find a bad state, we have established that the criteria are satisfied, which by itself is not a proof but indicates the absence of programming bugs and conceptual problems. If the model checker finds a bad state, it is possible to extract a counter example for future inspection.

Besides a logical formulation of the criteria, the bounded model checking requires a formal description of the voting scheme, i.e. an implementation of the voting scheme in programming languages whose semantics is clearly defined. Fragments of programming languages with a clear mathematical foundation are preferred to capture the essence of the voting algorithm. In our earlier work we have shown that linear logic is adequate to express voting schemes, and that proof search within linear logic is tantamount to bounded model checking.

4 Case Study: Variants of the Single Transferable Vote Scheme

Single transferable vote (STV) is a preferential voting scheme [15] for multi-member constituencies aiming to achieve proportional representation according to the voters' preferences.

4.1 The Standard Version of STV

There are many versions of STV, but most are an extension or variant of the standard version that is shown in Figure 1.

For input and output of the algorithm, we use the same notation and encoding as in Section 2. There are `V` voters electing `S` of `C` candidates, and:

- b: is the input ballot box, where $b[i, j]$ is the number of the candidate that is ranked by vote i in the jth place. If the vote does not rank all candidates, then $b[i, j] = 0$ for the empty places.
- r: is the output election result, where $r[i]$ is the ith candidate that is elected ($1 \leq i \leq \mathtt{S}$). If less than `S` candidates are elected, then $r[i] = 0$ for the empty seats.

We assume the input for the algorithm to satisfy the following conditions (which are pre-conditions for running the standard STV algorithm): (1) $\mathtt{C} \geq \mathtt{S}$, (2) $\mathtt{V} \geq 1$. and (3) votes are linear orders of a subset of the candidates, i.e., for all $1 \leq i \leq \mathtt{V}$ and all $1 \leq j, j' \leq \mathtt{C}$:

- $0 \leq b[i, j] \leq \mathtt{C}$,
- if $b[i, j] \neq 0$ and $j \neq j'$ then $b[i, j] \neq b[i, j']$,
- if $b[i, j] = 0$ then $b[i, j'] = 0$ for all $j' \geq j$.

The initialisation part of the STV algorithm in particular computes a quota necessary to obtain a seat (line 5). Different definitions of quotas are used in practice, and the most common is the Droop quota $\mathtt{Q} = \lfloor \mathtt{V}/(\mathtt{S}+1) \rfloor + 1$.

To determine the election result, STV uses an iterative process, which repeats the following two steps until either a winner is found for every seat or the number of remaining candidates equals the number of open seats (lines 10–33).

1. If no candidate reaches the quota of first-preference votes, a candidate with a minimal number of first-preference votes is eliminated and that candidate is deleted from all ballots (lines 17–19).
2. Otherwise one of the candidates with `Q` or more first-preference votes is chosen (line 23) and declared elected (line 24). Of the first-preference votes for that candidate, `Q` are chosen and erased (lines 26–29). These are the votes that are considered to have been "used up". If the candidate has more than `Q` votes, the surplus votes remain in the ballot box. Finally, the elected candidate is deleted from all ballots still in the box.

The procedure for deleting a candidate c (lines 40–47 works by searching for the candidate in each vote and, if c is found to have preference j, then the candidate with preference $j + 1$ moves to preference j, the candidate with preference $j + 2$ moves to preference $j + 1$, and so on.

When the main loop of the standard STV algorithm as shown in Figure 1 terminates, either (a) all seats are filled, or (b) the number `cc` of remaining candidates is equal to the number of open seats. In case (b), a further step is needed to distribute some or all of the remaining candidates to the equal

Standard Version of STV

```
1  // Initialisation
2  r  := [0, ..., 0];            // no one elected yet
3  e  := 1;                      // e is the next seat to be filled
4  cc := C;                      // cc is the number of (continuing) candidates
5  Q  := ⌊V/(S + 1)⌋ + 1;        // Droop quota

7  // Main loop: While not all seats filled and
8  //            there are more continuing candidates than open seats
9  // In each iteration one candidate is elected or one candidate eliminated
10 while (e ≤ S) ∧ (cc > S − e + 1) do
11     // QuotaReached is the set of candidates for which the number of
12     // first-preference votes reaches or exceeds the quota Q
13     QuotaReached := {c | 1 ≤ c ≤ C ∧ #{v | 1 ≤ v ≤ V ∧ b[v, 1] = c} ≥ Q};
14     if QuotaReached = ∅ then
15         // no one has reached the quota,
16         // eliminate a weakest candidate by deletion from the ballot box
17         Weakest := {c | 1 ≤ c ≤ C ∧ #{v | 1 ≤ v ≤ V ∧ b[v, 1] = c} is minimal};
18         choose c ∈ Weakest;
19         delete(c);
20     else
21         // one or more candidates have reached the quota,
22         // elect one of them
23         choose c ∈ QuotaReached;
24         r[e] := c;  // put c in the next free seat
25         e := e + 1; // increase the number e of the next seat to be filled
26         do Q times                                  // Q of the votes that
27             choose i ∈ {i | 1 ≤ i ≤ V ∧ b[i, 1] = c}; // give c top preference
28             for j = 1 to C do b[i, j] := 0; od     // get erased
29         od
30         delete(c);   // delete c from the ballot box
31     fi
32     cc := cc − 1;    // in any case we have one less continuing candidate
33 od

35 // Fill the empty seats
36 if e < S then
37     fill the remaining seats r[e, . . . , S] with the remaining cc candidates

39 // procedure for deleting candidate c from votes in b
40 procedure delete(c) begin
41     for i = 1 to V do for j = 1 to C do
42         if b[i, j] = c then
43             for k = j to C − 1 do b[i, k] := b[i, k + 1] od;
44             b[i, C] := 0;
45         fi
46     od od
47 end
```

Standard Version of STV

Fig. 1. The standard STV algorithm

number of remaining seats. The default is to fill all the remaining seats with the remaining candidates (line 37). Alternatively, one may continue the main STV loop to see if the further candidates get elected (which may leave seats open).

Example 2. We consider the same situation as in Example 1, i.e., there are four candidates A, B, C, D for two vacant seats, and the votes to be counted are $[A, B, D], [A, B, D], [A, B, D], [D, C], [C, D]$. The Droop quota in this case is $\mathtt{Q} = \lfloor 5/(2 + 1) \rfloor + 1 = 2$.

In the first iteration of the main loop, candidate A meets the quota and is hence elected. Two of the votes $[A, B, D]$ are erased, the third is a surplus vote. It is transformed into $[B, D]$ by deleting A from the ballots.

In the second iteration no candidate reaches the quota, thus the weakest of the remaining candidates B, C, D is eliminated – which one depends on the kind of tie-breaker used as all three have exactly one first-preference vote at that point. (1) If the tie-break eliminates B, the aforementioned transformed vote $[B, D]$ will be transformed again and will become a vote for D, so that D will be elected in the next iteration. (2) If the tie-break eliminates C, the vote $[C, D]$ will be transformed into a vote for D, and thus D will be elected. (3) If the tie-break eliminates D, then C will be elected, analogously, in the next iteration. In summary, the algorithm reports either $[A, D]$ or $[A, C]$ as the election result but not, for example, $[A, B]$ or $[B, D]$. If the number of second-preference votes is used as a tie-breaker, then B is eliminated first (case 1 above).

The standard STV algorithm has three choice points that are sources of non-determinism. These are resolved in different ways by different variants of STV:

1. Who is eliminated if several candidates have the same minimal number of first preferences (line 18)?
2. Who is elected if several candidates have reached the quota (line 23)?
3. How are the votes chosen that are deleted when an elected candidate has more than quote votes (line 27)?

Choice points (1) and (2) are typically handled – to some extent at least – by defining various kinds of tie-break rules. They can also be handled by declaring all weakest candidates eliminated resp. declaring all strongest candidates elected. That, however, is not always possible (there may not be enough open seats). And it can affect the election result in unexpected ways.

Choice point (3) can be eliminated using the notion of fractional votes. Instead of erasing a fraction of the votes that needs to be chosen, the same fraction of each vote is erased and the remaining fraction remains in the ballot box. This is done in many versions of STV used in real-world elections.

The above considerations illustrate that the STV algorithm as presented in this section is not only one but an entire family of vote counting algorithms. There are a number of parameters to play with: the quota, the choice of tie-breakers, placement of candidates once there are as many free seats as remaining candidates.

There are further options that – we argue in Section 4.2 – lead to election systems that can no longer be considered part of the STV family.

4.2 The CADE-STV Election Scheme

The bylaws of the Conference on Automated Deduction (CADE) specify an algorithm for counting the ballots cast for the election of members to its Board of Trustees [4]. The intention of the bylaws is to design a voting algorithm that

takes the voters' preferences into account. The algorithm has been implemented in Java and used by several CADE Presidents and Secretaries in elections for the CADE Board of Trustees. It has later on also been used by TABLEAUX Steering Committee Presidents, including one of the authors, for the election of members to the TABLEAUX Steering Committee.

Pseudo code for the CADE-STV scheme is included in the CADE bylaws [4], which makes it an interesting target for formal analysis. CADE-STV differs from the standard version of STV as shown in Figure 1 in several ways:

Quota Instead of the Droop quota, CADE-STV uses a quota of 50% of the votes – independently of the number of seats to be filled. That is, line 5 in Fig. 1 is changed to "`Q := ⌊V/2⌋ + 1`".

Empty seats CADE-STV does not fill seats that remain open at the end of the main loop, i.e., lines 36–37 are removed.

Restart Each time a candidates c reaches the quota `Q` of first-preference votes and gets elected, the election for the next seat restarts with the original ballot box – with the only exception that the elected candidate c is deleted. Thus, (a) the `Q` votes used to elect c are not erased but are only changed by deleting c, and (b) weak candidates that have been eliminated are "resurrected" and take part in the election again. That is, (a) the code for erasing votes (lines 26–29) is removed and (b) replaced by code for resurrecting the eliminated candidates.

4.3 Effects of the Differences between CADE-STV and Standard STV

Effects of Restart To illustrate the effect of the restart mechanism in CADE-STV on the election result, we consider an example:

Example 3. Let us run CADE-STV on Example 1. First, we compute the majority quota `Q` $= 3$. In the first iteration, A has three first preferences, which means that A is the majority winner and is seated. Since CADE-STV uses restart, A's votes are not deleted but are redistributed at the end of the first iteration. Now the ballot box contains $[B, D], [B, D], [B, D], [D, C], [C, D]$. Following the algorithm, we observe that now B is the majority candidate with 3 first preference votes and is seated. The election is over, and the election result is $[A, B]$ (which is different from the possible results $[A, D]$ or $[A, C]$ of standard STV).

Running our bounded model checker for analysing STV schemes that we have described in [1] on CADE-STV confirms that the election results computed by CADE-STV do not always satisfy Criterion 1, which is closely related to proportional representation (see Sect. 2.3). Indeed, our bounded model checker finds smaller counter examples than the one shown in Example 3, but these are not as illustrative.

The effect of the differences between standard STV and CADE-STV is further clarified by the following theorem and its corollary: in certain cases, there is no proportional representation in the election results computed by CADE-STV. See also Example 4 below.

Theorem 1. *If a majority of voters vote in exactly the same way* $[c_1, \ldots, c_k]$*, then CADE-STV will elect the candidates preferred by that majority in order of the majority's preference.*

Proof. Since a majority of voters choose c_1 as their first preference, no other candidate can meet the "majority quota". Thus c_1 is elected in the first round. When redistributing the ballots, each of the majority of ballots with c_1 as first preference have c_2 as second preference. All become first preferences for c_2. Thus candidate c_2 is guaranteed to have a majority of first preferences and is elected in round two, and so on until all vacancies are filled. □

Corollary 1. *If the electorate consists of two diametrically opposed camps that vote for their candidates only, in some fixed order, then the camp with a majority will always get their candidates elected and the camp with a minority will never get their candidate elected.*

Standard STV does not use the restart mechanism and so it will elect the first ranked candidate of the majority, but will then reuse only the surplus votes and not all votes as done by CADE-STV. Thus the second preference from the majority is not necessarily the second person elected. Consequently, majorities do not rule outright in standard STV.

Effects of High Quota and No Filling of Empty Seats No matter how many candidates there are and how many seats need to be filled, a candidate can only be seated by CADE-STV if he or she accumulates more than 50% of the votes. Any candidate with less than 50% of the vote is defeated. Thus, CADE-STV obviously violates the fill-all-seats criterion. But because of the high quota it also prevents proportional representation as candidates supported by a large minority can neither be elected via reaching the quota nor via filling seats left empty at the end of the main loop.

In fact, if the high quota of 50% and no filling of empty seats were the only changes w.r.t. standard STV, only a single candidate could be elected because more than 50% of the votes would be used up by electing that candidate. CADE-STV requires the restart mechanism to elect further candidates.

Example 4. Assume that there are 100 seats and two parties nominating candidates $A_1, \ldots, A_{100}$ and $B_1, \ldots, B_{100}$, respectively. Further assume that there are 51% of A-voters and 49% of B-voters. All A-voters vote $[A_1, \ldots, A_{100}]$ and all B-voters vote $[B_1, \ldots, B_{100}]$. Standard STV elects $A_1, \ldots, A_{51}, B_1, \ldots, B_{49}$, i.e., the result is a perfect proportional representation.

With a quota of 50% and no filling of empty seats, only A_1 gets elected and then nothing further happens, which is clearly undesirable. But CADE-STV uses, in addition, the restart mechanism. Therefore, like standard STV, it fills all seats. The result is different, however, because the votes used to elect $A_1, \ldots, A_{51}$ do not get erased. CADE-STV produces the election result $[A_1, ..., A_{100}]$.

The above example again shows that the majority can rule with CADE-STV and there is no proportional representation in that case (Corollary 1).

4.4 Observations on the History of CADE-STV

We discuss the history of the CADE-STV scheme because it illustrates the problem of evolving an election scheme without using formally specified semantic criteria and a formal definition of the input to the scheme. It is publicly known that there were lots of discussions among the CADE Trustees over a long period of evolving CADE-STV. But we do not know what the non-public deliberations actually where. The following is based on our interpretation of the publicly available material.

The Violation of Proportional Representation The CADE-STV voting scheme is the result of a long discussion among the board of trustees that took place in the years 1994–1996. David A. Plaisted published various concerns about the existing voting scheme which can be found on his homepage [12].

One of Plaisted's concerns was that a minority supporting candidates standing for re-election could re-elect these candidates against the wishes of the majority as that majority is not sufficiently coordinated in its behaviour to elect alternative candidates [12]:

> *Of course, one of the main purposes of a democratic scheme is to permit the membership to vote a change in the leadership if there is a need for this. However, the new bylaws make this more difficult in several ways. The problem is that those who are unsatisfied with the scheme will tend to split their votes among many candidates (unless they are so disgusted as to put the trustee candidates at the very bottom of the list), but those who are satisfied will tend to vote for the trustee nominees. This means that the trustee nominees tend to be elected even if only a minority is happy with the scheme.*

We believe that because of Plaisted's concerns the board introduced the high 50% quota and did not include a mechanism for filling seats that remain empty. On first sight, this seems good because it solves the problem illustrated in Plaisted's scenario. But, as explained above, this deviation from the standard STV setup not only violates the fill-all-seats criterion but also the goal of proportional representation (see Example 4). Thus, the CADE-STV scheme protects the majority at the expense of the minority.

Also, as explained above, if the high quota and the remaining empty seats were the only changes, only a single candidate could be elected. So, in effect, one was forced to change the algorithm further. The result was that the restart mechanism was added to the algorithm, that reuses the original ballot box for each seat and does not erase votes (because then more candidates can be elected, see Example 4).

There would have been a different solution than using a restart that would have solved Plaisted's problem without restricting proportional representation as much: One could have used Standard STV with an additional rule that – before the main algorithm is started – anybody who does not appear (with arbitrary preference) on at least 50% of the votes is immediately eliminated.

Example 5. Using the same input ballots as in Example 4, the algorithm would then elect $[A_1, ..., A_{51}]$, which still suppresses the B minority, but at least gives the A party only those seats that are proportional to the A votes.

Well-formedness and Interpretation of Input Apparently, during some CADE elections, there was some confusion about the meaning of not listing a candidate at all on a ballot and how that should be translated into input for the CADE-STV voting scheme.

The instruction was given to the voters that not listing a candidate is the same as giving that candidate the lowest possible preference. But that is not the correct interpretation. It is easy to see that for both standard STV and CADE-STV, there is a difference between giving a candidate the lowest possible preference and not listing the candidate at all. For example, if there are candidates A, B, C, then $[A, B]$ is different from $[A, B, C]$. When candidates A and B get eliminated, $[A, B, C]$ turns into a vote for C and may help to elect C, which $[A, B]$ does not. One could transform a ballot of the form $[A, B]$ into an input vote $[A, B, C]$ (and, thus, make them equal by definition). But that only works if a single candidate is missing from the ballot. If more are missing, they would have to be put in the same spot on the ballot, which is not possible. Indeed, CADE-STV does not work correctly if input votes contain candidates with equal preference, i.e., if the pre-condition that a vote is a partial linear order is violated. As that pre-condition was never clearly specified, fixing the problem in CADE-STV was a lengthy process that took several years.

This shows that not formalising the pre-conditions which the input must satisfy is problematic. Besides the possibility of errors or unintended behaviour of the algorithm, it is important that the voters understand how their ballot is transformed into input for the algorithm.

5 Conclusion

We have discussed semantic criteria for desired properties of voting schemes. And our case study demonstrates the importance of such criteria both for formal analysis of voting schemes and their evolution and the development process. Semantic criteria need to be explicitly stated. A discussion of voting schemes using anecdotal descriptions of individual voting scenarios is not a good basis for making electoral laws.

In future work, we plan to implement more efficient analysis tools based on SMT solvers for checking that criteria are satisfied. This will allow to investigate larger classes of voting schemes and to use more complex criteria. We also plan to extend our analysis to criteria that measure the quality of election results based on difference measures [10] in addition to yes/no criteria.

References

1. Beckert, B., Goré, R., Schürmann, C.: Analysing vote counting algorithms via logic. And its application to the cade election system. In: Proceedings, 24th International

Conference on Automated Deduction (CADE), Lake Placid, NY, USA. LNCS, Springer (2013)
2. Brams, S., Sanver, R.: Voter sovereignty and election outcomes. Retrieved from http://www.nyu.edu/gsas/dept/politics/faculty/brams/sovereignty.pdf (2003), accessed 21 March 2013
3. Brandt, F., Conitzer, V., Endriss, U.: Computational social choice. In: Weiss, G. (ed.) Multiagent Systems. MIT Press (2012), forthcoming. Available at http://www.illc.uva.nl/~ulle/pubs/files/BrandtEtAlMAS2012.pdf
4. CADE Inc.: CADE Bylaws (effective Nov. 1, 1996; amended July/August 2000). Retrieved from http://www.cadeinc.org/Bylaws.html, accessed 20 Jan 2013
5. Cochran, D.: Formal Specification and Analysis of Danish and Irish Ballot Counting Algorithms. Ph.D. thesis, ITU (2012)
6. Court, F.C.: Provisions of the federal electoral act from which the effect of negative voting weight emerges unconstitutional. Press release no. 68/2008 (2008)
7. Hill, I.D., Wichmann, B.A., Woodall, D.R.: Single transferable vote by Meek's method. Computer Journal 30(3) (1987)
8. J R. Kiniry, D.C., Tierney, P.E.: Verification-centric realization of electronic vote counting. Tech. rep., School of Computer Science and Informatics, University College Dublin (2007)
9. M McGaley, J.P.G.: Electronic voting: A safety critical system. Tech. Rep. NUIM-CS-TR2003-02, Department of Computer Science, National University of Ireland, Maynooth (March 2003)
10. Meskanen, T., Nurmi, H.: Closeness counts in social choice. In: Braham, M., Steffen, F. (eds.) Power, Freedom, and Voting. Springer (2008)
11. Pacuit, E.: Voting methods. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Winter 2012 edn. (2012)
12. Plaisted, D.A.: A consideration of the new cade bylaws. Retrieved from http://www.cs.unc.edu/Research/mi/consideration.html, accessed 22 Mar 2013
13. Sun, Y., Zhang, C., Pang, J., Alcalde, B., Mauw, S.: A trust-augmented voting scheme for collaborative privacy management. Journal of Computer Security 20(4), 437–459 (2012)
14. Wikipedia: Monotonicity criterion. Retrieved from http://en.wikipedia.org/wiki/Monotonicity_criterion, accessed 21 March 2013
15. Wikipedia: Single transferable vote. Retrieved from http://en.wikipedia.org/wiki/Single_transferable_vote, accessed 20 Jan 2013