Counting Votes with Multisets*

Bart Jacobs
Radboud University
Nijmegen, The Netherlands
`bart@cs.ru.nl`

Michael Johnson
Richard Buckland
University of New South Wales
Sydney, Australia
`michael.johnson1@unsw.edu.au`
`richardb@unsw.edu.au`

A multiset is a 'set' in which elements may occur multiple times. These structures are ideal for expressing the outcome of an election, for instance of the form 60 'yes' and 40 'no'. Moreover, multisets are a useful datatype in vote counting algorithms. This will be illustrated in three different forms of vote counting, known as: 'instant-runoff', 'De Borda', and 'single transferrable vote'. The relevant abstract properties of multisets are: (1) they form a (free) commutative monoid, and (2) they form a functor, and (3) also a monad. This paper illustrates how such categorical properties can be put to good use in deriving and expressing election outcomes. The emphasis is not on the (elementary) category theory involved, but on its application in voting systems.

1 Introduction

Counting votes seems obvious. Suppose there are two candidates A and B, and one hundred people have voted, precisely once, either A or B. Then we can arrive at an outcome of, say, 72 votes for A and 28 votes for B. This paper looks at the datatypes that are appropriate for counting votes and for reporting the vote outcome. For instance, at the end of the voting period one may have a *list* of length 100, containing 72 A's and 28 B's, in some order. The vote outcome, we claim, is best described as a *multiset*. In this case it will be written as $72|A\rangle + 28|B\rangle$. In such a multiset the order of the elements is irrelevant, only their multiplicity counts. These multiplicities are written as numbers before 'ket' symbols $|\cdot\rangle$, with the candidates written inside the ket.

What we have described is a very simple election, with only two candidates, and only one vote per candidate. There may be more candidates or multiple options, and voters may express preferences, in the form of a list of candidates, say in descending order of preference. The term 'preferendum' is sometimes used for such elections. There may be good reasons to organise a preferendum, with multiple weighted options, instead of a referendum, with only two options. A referendum may be hijacked to express discontent on other matters — as perhaps happened in the case of Brexit. A preferendum, in contrast, invites a more nuanced stance, and seems less susceptible to external influence. These political considerations are interesting, but are not the topic of this paper, and only serve as motivation.

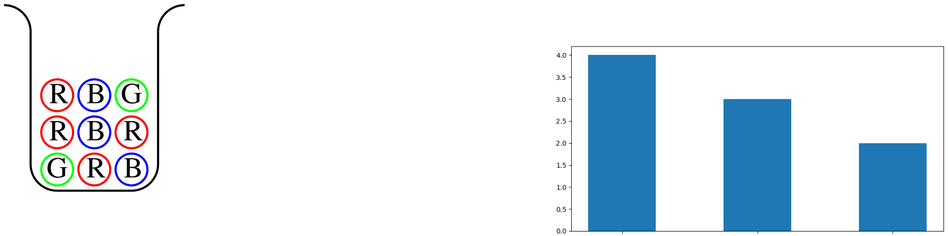
It turns out that it is not trivial to describe, explain, or implement these vote counting mechanisms with multiple weighted options. Things become even more difficult when surplus votes (after reaching a threshold) are transferred, in some form. The two main points of this paper are the following.

1. Multisets are the appropriate datatype for vote counting and reporting.
2. Some elementary techniques from category theory capture the operations on multisets that are relevant in vote couting algorithms (like instant-runoff, De Borda, single transferable votes). In particular, the fact that multisets form a functor, a monad, and carry a (free) monoid structure turns out to be remarkably useful.

---

*Version of March 30, 2026

To appear in EPTCS.



$$4|R\rangle + 3|B\rangle + 2|G\rangle$$

Figure 1: An urn with four red, three blue and two green balls, described as multiset on the left, with the corresponding bar chart on the right.

The paper aims to demonstrate these points, mainly via examples of different types of elections, with different forms of vote counting. The (partly implicit) point that the paper aims to make is that describing these vote counting algorithms in terms of multisets makes them ‘abstract’ so that they can be described in a few lines. One can debate whether abstractness contributes to proper understanding, but mathematicians generally think so. And many computer scientists think so too, since it makes it easier for them to produce provably correct implementations. Again largely implicit, is the motivation that vote counting should happen in a transparant and correct manner.

The category theory used in this paper is relatively elementary, but it is extremely useful. The contribution of this paper is thus not categorical, but lies in recognising and exploiting the relevant categorical structure in these applications, especially for multisets.

This paper explains different vote count methods mainly via examples, with relatively small numbers. The calculations can, in principle, be checked by hand. For this paper we have used Python scripts, from (an updated version of) the EfProb library [5].

The paper devotes quite a bit of space to explaining multisets and their categorical structure. These explanations are aimed at non-category theorists, to make the material accessible outside the circle of (categorical) experts. Readers who wish to learn more category theory are referred to the extensive literature [1, 2, 10, 13, 11, 14, 4, 12]. The next two sections contain a gentle introduction to the datatypes of multisets and lists. Lists are ubiquitous in computer science and mathematics, but multisets seem to be an orphaned, under-valued datatype. Section 4 describes the ‘functoriality’ of multisets and puts this property to good use in describing so-called instant-runoff voting to determine a winner from lists of preferences. The fact that multisets form a monad is used in Section 5 for De Borda vote counting, with different weights. Finally, Section 6 describes the mechanism of single transferable votes for multi-member electorates. As will be discussed in those sections, these different types of elections are actually used, at various places in the world. Notably Australia has a rich tradition of (subtle) vote counting. The paper ends with a number of conclusions.

2 A first look at multisets

A set is a collection of elements. For instance, one may have a set of colours $S = \{R, B, G\}$, for red, blue and green. In a set, elements occur at most once. There is the set of natural numbers $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ in which is each number occurs precisely once. The elements in a set are not ordered. We can also write $\mathbb{N} = \{1, 0, 3, 2, 5, 4, \ldots\}$.

There are situations where one wishes to be able to talk about multiple occurrences of elements.

Figure 1 shows an urn / vase containing coloured balls. One sees that the urn contains nine balls in total, of which four are red, three blue and two green. This is a typical example of a multiset. We shall use special 'ket' notation and write this urn / bag / multiset as:

$$4|R\rangle + 3|B\rangle + 2|G\rangle \qquad \text{expressing} \qquad \begin{cases} \text{4 red} \\ \text{3 blue} \\ \text{2 green.} \end{cases}$$

The verticle bar $|$ and the right angle $\rangle$ together form what it is called a ket $|\cdot\rangle$. Inside such a ket we write an element, like a color *R* / *B* / *G*. Before the ket we write how often that element occurs in a multiset. These kets have no mathematical meaning but are used as notation to separate the elements (inside) from their multiplicities (upfront).

The important thing to note is that the urn in Figure 1 may also be seen as a *ballot box*, containing nine votes, namely four for option *R*, three for option *B*, and two for option *G*. It is at this stage unclear — and irrelevant — what these three options are. The important point is that options that voters express typically occur multiple times, and thus can be described by multisets. The bar chart on the right in Figure 1 tabulates the votes and clearly shows that option *R* has most votes.

Convention 2.1 *Several conventions apply when we write multisets via kets.*

1. *The order in which the items are listed does not matter. Thus we have equalities of multisets:*

$$4|R\rangle + 3|B\rangle + 2|G\rangle \;=\; 2|G\rangle + 3|B\rangle + 4|R\rangle \;=\; 3|B\rangle + 2|G\rangle + 4|R\rangle.$$

2. *Zero occurrences are typically ommitted. If we write Y for yellow, then:*

$$0|Y\rangle + 4|R\rangle + 3|B\rangle + 2|G\rangle \;=\; 4|R\rangle + 3|B\rangle + 2|G\rangle.$$

We may write the term $0|Y\rangle$ to emphasise that there are no (zero) yellow balls (or votes) in the urn, but usually this is not so relevant.

3. *The multiplicities of multiply occurring kets with the same element are added, as in:*

$$2|R\rangle + 2|G\rangle + 3|B\rangle + 2|R\rangle \;=\; 4|R\rangle + 3|B\rangle + 2|G\rangle. \tag{1}$$

This makes sense, both in terms of urns and ballot boxes.

This third equation will turn out to be very useful.

We are used to adding numbers, like in $2+3=5$. One can also add multisets. This can be represented by pooring the contents of two urns / ballot boxes into a new urn, see Figure 2. We write this addition of multisets also as $+$. This may be confusing at first, but is actually convenient. For instance the adding of multisets in Figure 2 can be written as a mathematical equation:

$$\Big(4|R\rangle + 3|B\rangle + 2|G\rangle\Big) + \Big(1|R\rangle + 2|B\rangle + 3|G\rangle\Big) \;=\; 5|R\rangle + 5|B\rangle + 5|G\rangle.$$

In fact, this equation can be seen as a consequence of Convention 2.1 (3), in the style of Equation (1).

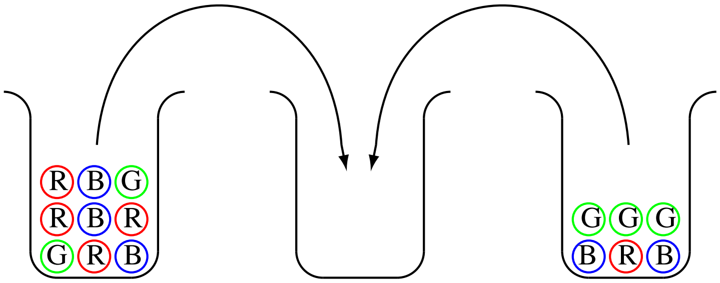


Figure 2: Addition of multisets can be represented via a join of contents of two urns / ballot boxes. The resulting urn in the middle will contain the sum of the two outer multisets, namely: $5|R\rangle + 5|B\rangle + 5|G\rangle$.

Notation 2.2 *For a set X, we write $\mathcal{N}(X)$ for the set of multisets, written as $n_1|x_1\rangle + \cdots + n_K|x_K\rangle$, with elements $x_i \in X$ with natural numbers as multiplicities $n_i \in \mathbb{N}$.*

For a such a multiset $\varphi = \sum_i n_i|x_i\rangle \in \mathcal{N}(X)$ we write $\|\varphi\| \in \mathbb{N}$ for the size of the multiset, given by the sum of multiplicities:

$$\|\varphi\| = \Big\| \textstyle\sum_i n_i|x_i\rangle \Big\| = \textstyle\sum_i n_i.$$

For two multisets $\varphi, \psi \in \mathcal{N}(X)$ we write $\varphi + \psi \in \mathcal{N}(X)$ for the sum of multisets, obtained by addition of multiplicties, basically as in (1)*. This addition is comutative, satisfying $\varphi + \psi = \psi + \varphi$ for all multisets $\varphi, \psi \in \mathcal{N}(X)$.*

Writing $\mathbf{0} \in \mathcal{N}(X)$ for the empty / zero multisets, with multiplicity zero for each element from X, we get $\varphi = \mathbf{0} + \varphi = \varphi + \mathbf{0}$.

In Figure 2, the two urns on the left and right have size 9 and 6. Their sum has size 15. Indeed, in general one has $\|\varphi + \psi\| = \|\varphi\| + \|\psi\|$. The general, mathematical statement is that $\mathcal{N}(X)$ is a (free) commutative monoid, and that the size map $\| - \| \colon \mathcal{N}(X) \to \mathbb{N}$ is a homomorphism of monoids, that preserves addition (including zero). For (much) more information, see [9].

Now that we know about addition of multisets, we can already compute outcomes of elections. We illustrate this in a simple preferendum.

Example 2.3 *Consider an election with a set of four options $V = \{a, b, c, d\}$. Voters can express a vote of the form (x, y, z), where $x \in V$ is their first choice that counts three times, $y \in V$ is their second choice that counts twice, and z is their third choice that counts only once. In a valid vote (x, y, z) these choices x, y, z should all be different. There are in this simple example seven votes, of the form:*

$$(d,a,c) \quad (a,b,d) \quad (c,a,b) \quad (d,a,c) \quad (c,b,a) \quad (a,b,c) \quad (d,b,c).$$

We can now calculate the vote outcome as a multiset over V, that is, as an element of $\mathcal{N}(V)$. We proceed by turning each vote (x, y, z) into a multiset $3|x\rangle + 2|y\rangle + 1|z\rangle$. Then we can add all these multisets. This yields:

$$\begin{aligned}&\Big(3|d\rangle + 2|a\rangle + 1|c\rangle\Big) + \Big(3|a\rangle + 2|b\rangle + 1|d\rangle\Big) + \Big(3|c\rangle + 2|a\rangle + 1|b\rangle\Big) + \Big(3|d\rangle + 2|a\rangle + 1|c\rangle\Big) \\ &\qquad + \Big(3|c\rangle + 2|b\rangle + 1|a\rangle\Big) + \Big(3|a\rangle + 2|b\rangle + 1|c\rangle\Big) + \Big(3|d\rangle + 2|b\rangle + 1|c\rangle\Big) \\ &= 13|a\rangle + 9|b\rangle + 10|c\rangle + 10|d\rangle.\end{aligned}$$

Thus, option a is most popular. This is an instance of a De Borda count [3, 6, 7], which we shall describe more systematically in Section 5. The calculations require some care when done by hand, but when you use a programming language (or package) that supports multisets, the counting can be done systematically, at a high level of abstraction.

3 From lists to multisets

One can imagine a situation where people have to write their vote, say Y for yes and N for no, on a piece of paper and then hand it over to some trusted person who counts the votes. This person ends up with a list / pile of papers. The outcome can then be extracted by turning the list into a multiset. We call this process *accumulation* and use a function *acc* to do this. This could look as follows.

$$acc\Big(\langle Y,Y,N,Y,N,N,N,Y,Y,Y\rangle\Big) \;=\; 6|Y\rangle + 4|N\rangle. \tag{2}$$

Using lists instead of multisets as datatype for votes is not such a good idea, mainly because it may leak information and thus compromise confidentiality. Indeed, if someone has registered the order in which people have voted, then seeing the election outcome as a list of votes, makes it possible to reconstruct individual votes.

An additional advantage of using multisets is that it is easy to see the outcome, as on the right in (2). Seeing the list does not immediately tell us the conclusion.

There is some mathematical structure at work that is nice to make explicit.

Notation 3.1 *For a set X we write $\mathcal{L}(X)$ for the set of (finite) lists with elements from the set X. A typical element of $\mathcal{L}(X)$ is of the form $\langle x_1,\ldots,x_n\rangle$, with elements $x_i \in X$. The order of elements does matter in a list. Elements may occur multiple times in a list.*

Given two lists $\ell_1,\ell_2 \in \mathcal{L}(X)$ we write $\ell_1 +\!\!+\, \ell_2 \in \mathcal{L}(X)$ for their concatenation. It is simply obtained by gluiing them together, in order, as in:

$$\langle x_1,\ldots,x_n\rangle +\!\!+ \langle y_1,\ldots,y_m\rangle \;=\; \langle x_1,\ldots,x_n,y_1,\ldots,y_m\rangle.$$

Together with the empty list $\langle\rangle \in \mathcal{L}(X)$ this turns the set $\mathcal{L}(X)$ into a monoid — but not a commutative one.

We write $\mathrm{acc}\colon \mathcal{L}(X) \to \mathcal{N}(X)$ for the accumulation function that turns a list into a multiset by counting occurrences and forgetting the order. This may be written as:

$$\mathrm{acc}\Big(\langle x_1,\ldots,x_n\rangle\Big) \;=\; 1|x_1\rangle + \cdots + 1|x_n\rangle.$$

This is a convenient and clever definition, since the multiplicities of multiply occurring elements are automatically added by Convention 2.1 (3).

A basic fact is that accumulation maps concatenations to sums, as in:

$$acc\big(\ell_1 +\!\!+\, \ell_2\big) \;=\; acc\big(\ell_1\big) + acc\big(\ell_2\big) \qquad \text{and} \qquad acc\big(\langle\rangle\big) \;=\; \mathbf{0}. \tag{3}$$

Formulated more abstractly, the function $acc\colon \mathcal{L}(X) \to \mathcal{N}(X)$ is a homomorphism of monoids.

This is directly relevant for voting. Consider a polling place with three ballot boxes BB_1, BB_1, BB_3 in which lists ℓ_1, ℓ_2, ℓ_3 of votes are collected, and counted at the end of the day:

$$\boxed{\text{voters}} \left.\begin{array}{l} \xrightarrow{\ell_1} BB_1 \\ \xrightarrow{\ell_2} BB_2 \\ \xrightarrow{\ell_3} BB_3 \end{array}\right\} \xrightarrow{\text{vote counting}} \; acc\big(\ell_1\big) + acc\big(\ell_2\big) + acc\big(\ell_3\big).$$

The sum of the three accumulated lists on the right-hand-side is the outcome of the election. The sum $+$ is a sum of multisets. Notice that by using this sum $+$ of multisets, we assume that each ballot box is counted separately. One could also first put the piles of votes together via concatenation $+\!\!+$ and then count them all via accumulation. This gives as election outcome $acc\big(\ell_1 +\!\!+\, \ell_2 +\!\!+\, \ell_3\big)$. Via (3) we can be sure that the outcome is the same.

4 Functoriality of multisets and instant-runoff voting

Instant-runoff voting is a vote count mechanism that is used[1] to elect members of the Australian House of Representatives and the National Parliament of Papua New Guinea. It is also used to elect the head of state in India, Ireland, and Sri Lanka. The vote counting happens in several rounds, in which one candidate with the least highest-preference-votes, is eleminated, but where the other votes (with this eliminated candidate first) are still included, see below for details.

This vote counting process is non-trivial, both in complexity and in amount of work. It turns out that an abstract description can be given in terms of multisets, using that multisets form a *functor*. This means that the multiset operation $\mathcal{N}$ does not only apply to sets, like in Notation 2.2, but also to functions between them. This property is also called *functoriality*.

Definition 4.1 *Let X, Y be two arbitrary sets, with a function $f\colon X \to Y$ between them. In this situation we can form the two sets $\mathcal{L}(X)$ and $\mathcal{L}(Y)$ of lists over X and over Y, and also the sets $\mathcal{N}(X)$ and $\mathcal{N}(Y)$ of multisets over X and over Y.*

1. *One can define a function between these two sets of lists, written as $\mathcal{L}(f)\colon \mathcal{L}(X) \to \mathcal{L}(Y)$, namely:*
$$\mathcal{L}(f)\Big(\langle x_1, \ldots, x_n\rangle\Big) = \langle f(x_1), \ldots, f(x_n)\rangle. \tag{4}$$
2. *We can also define a function between the two sets of multisets, written as $\mathcal{N}(f)\colon \mathcal{N}(X) \to \mathcal{N}(Y)$, via:*
$$\mathcal{N}(f)\Big(\textstyle\sum_i n_i\big|x_i\big\rangle\Big) = \textstyle\sum_i n_i\big|f(x_i)\big\rangle. \tag{5}$$

The function $\mathcal{L}(f)$ in (4) performs what functional programmers call map-list. It simply applies the function f to all the elements of a list. The function $\mathcal{N}(f)$ in (5) works similarly, but the sum property of Convention 2.1 (3) may kick in, when different elements in X are mapped by the function f to the same element in Y. We shall exploit this property below.

In terms of urns filled with coloured balls, the function $\mathcal{N}(f)$ can be seen as a repainting of the balls in the urn, where the function $f\colon X \to Y$ maps one set of colours to another set. For instance, let's write Y for yellow and P for purple, and consider a repainting function $f\colon \{R,B,G\} \to \{Y,P\}$, given by $f(R) = f(G) = Y$ and $f(B) = P$. Then we can repaint the (contents) of the urn in Figure 1 via the function $\mathcal{N}(f)$, as elaborated in:

$$\begin{aligned}
\mathcal{N}(f)\Big(4\big|R\big\rangle + 3\big|B\big\rangle + 2\big|G\big\rangle\Big) &\overset{(5)}{=} 4\big|f(R)\big\rangle + 3\big|f(B)\big\rangle + 2\big|f(G)\big\rangle \\
&= 4\big|Y\big\rangle + 3\big|P\big\rangle + 2\big|Y\big\rangle \\
&= 6\big|Y\big\rangle + 3\big|P\big\rangle.
\end{aligned}$$

There is no such thing as repainting of ballots in a ballot box, but nevertheless we shall soon see the usefulness of functoriality in vote counting. But first we collect (without proof) some basic properties.

Lemma 4.2 *In the context of Definition 4.1,*

1. *$\mathcal{L}$ preserves function composition and identities: $\mathcal{L}(g \circ f) = \mathcal{L}(g) \circ \mathcal{L}(f)$ and $\mathcal{L}(id) = id$;*
2. *$\mathcal{N}$ also preserves composition and identities: $\mathcal{N}(g \circ f) = \mathcal{N}(g) \circ \mathcal{N}(f)$ and $\mathcal{N}(id) = id$;*

---

[1] according to Wikipedia en.wikipedia.org/wiki/Instant-runoff_voting, consulted Nov. 29, 2025

3. *$\mathcal{L}(f)$ preserves length and $\mathcal{N}(f)$ preserves size; the latter can be expressed as $\big\|\mathcal{N}(f)(\varphi)\big\| = \big\|\varphi\big\|$ for each multiset $\varphi \in \mathcal{N}(X)$;*
4. *Accumulation is natural:* $\mathrm{acc} \circ \mathcal{L}(f) = \mathcal{N}(f) \circ \mathrm{acc}$*, for each function f.* ❑

We shall describe instant-runoff vote counting via an example. It involves two auxilary functions on lists, namely a ‘head’ and ‘delete’ function hd and del_a for an element a. The head function hd selects the first element of a (non-empty) list, and the delete function del_a removes the element a from the entire list; it returns the list that remains. For instance:

$$hd\Big(\langle c,d,a,b\rangle\Big) = c \qquad\qquad del_a\Big(\langle c,d,a,b\rangle\Big) = \langle c,d,b\rangle.$$

We assume we have an election where one candidate must be chosen out of set $X = \{a,b,c,d\}$. There are 100 voters and they each give a list of preferences, $\langle x_1,x_2,x_3,x_4\rangle$ for $x_i \in X$, all different. We assume the 100 votes are divided in the following way as top-down preference sequences.

	7	2	7	8	7	19	12	12	8	6	12
1st	a	a	a	b	b	b	c	d	d	d	d
2nd	b	c	d	a	c	d	a	a	a	b	c
3rd	d	d	c	c	d	c	d	b	c	c	a
4th	c	b	b	d	a	a	b	c	b	a	b

(6)

We reformulate this table as a multiset $\varphi_0 \in \mathcal{N}\big(\mathcal{L}(X)\big)$ over lists, of the form:

$$\begin{aligned}\varphi_0 \;=\; & 7\big|\,a,b,d,c\,\big\rangle + 2\big|\,a,c,d,b\,\big\rangle + 7\big|\,a,d,c,b\,\big\rangle + 8\big|\,b,a,c,d\,\big\rangle \\ & + 7\big|\,b,c,d,a\,\big\rangle + 19\big|\,b,d,c,a\,\big\rangle + 12\big|\,c,a,d,b\,\big\rangle + 12\big|\,d,a,b,c\,\big\rangle \\ & + 8\big|\,d,a,c,b\,\big\rangle + 6\big|\,d,b,c,a\,\big\rangle + 12\big|\,d,c,a,b\,\big\rangle.\end{aligned}$$

In an instant-runoff counting process one first looks at the top row and one counts which candidates occurr there, with which multiplicities. By inspecting the above table (6) we see:

$$\begin{cases} \text{for } a, & 16 \;=\; 7+2+7 \\ \text{for } b, & 34 \;=\; 8+7+19 \\ \text{for } c, & 12 \\ \text{for } d, & 38 \;=\; 12+8+6+12. \end{cases} \tag{7}$$

Interestingly, these numbers can also be obtained via functoriality of $\mathcal{N}$, using the head function:

$$\begin{aligned}\mathcal{N}\big(hd\big)(\varphi_0) \overset{(5)}{=}\; & 7\big|\,hd(a,b,d,c)\,\big\rangle + 2\big|\,hd(a,c,d,b)\,\big\rangle + 7\big|\,hd(a,d,c,b)\,\big\rangle + 8\big|\,hd(b,a,c,d)\,\big\rangle \\ & + 7\big|\,hd(b,c,d,a)\,\big\rangle + 19\big|\,hd(b,d,c,a)\,\big\rangle + 12\big|\,hd(c,a,d,b)\,\big\rangle + 12\big|\,hd(d,a,b,c)\,\big\rangle \\ & + 8\big|\,hd(d,a,c,b)\,\big\rangle + 6\big|\,hd(d,b,c,a)\,\big\rangle + 12\big|\,hd(d,c,a,b)\,\big\rangle \\ =\; & 7\big|\,a\,\big\rangle + 2\big|\,a\,\big\rangle + 7\big|\,a\,\big\rangle + 8\big|\,b\,\big\rangle + 7\big|\,b\,\big\rangle + 19\big|\,b\,\big\rangle + 12\big|\,c\,\big\rangle + 12\big|\,d\,\big\rangle + 8\big|\,d\,\big\rangle + 6\big|\,d\,\big\rangle + 12\big|\,d\,\big\rangle \\ =\; & 16\big|\,a\,\big\rangle + 34\big|\,b\,\big\rangle + 12\big|\,c\,\big\rangle + 38\big|\,d\,\big\rangle\end{aligned}$$

These numbers correspond to the above count by hand (7). They are obtained via Convention 2.1 (3).

We see two things: (1) no-one has an absolute majority, so there is no winner yet; (2) candidate c has the lowest number of first preferences. With the instant-runoff method, this candidate c is removed as possible winner.

This removal again makes use of functoriality of $\mathcal{N}$, but now with del_c as function. We elaborate the details of the computation. It gives us a new multiset φ_1 with which the process is repeated.

$$\begin{array}{rcl}
\varphi_1 & \coloneqq & \mathcal{N}\big(del_c\big)(\varphi_0) \\
& \overset{(5)}{=} & 7\big|\,del_c(a,b,d,c)\,\big\rangle + 2\big|\,del_c(a,c,d,b)\,\big\rangle + 7\big|\,del_c(a,d,c,b)\,\big\rangle + 8\big|\,del_c(b,a,c,d)\,\big\rangle \\
& & \quad + 7\big|\,del_c(b,c,d,a)\,\big\rangle + 19\big|\,del_c(b,d,c,a)\,\big\rangle + 12\big|\,del_c(c,a,d,b)\,\big\rangle + 12\big|\,del_c(d,a,b,c)\,\big\rangle \\
& & \quad + 8\big|\,del_c(d,a,c,b)\,\big\rangle + 6\big|\,del_c(d,b,c,a)\,\big\rangle + 12\big|\,del_c(d,c,a,b)\,\big\rangle \\
& = & 7\big|\,a,b,d\,\big\rangle + 2\big|\,a,d,b\,\big\rangle + 7\big|\,a,d,b\,\big\rangle + 8\big|\,b,a,d\,\big\rangle + 7\big|\,b,d,a\,\big\rangle + 19\big|\,b,d,a\,\big\rangle \\
& & \quad + 12\big|\,a,d,b\,\big\rangle + 12\big|\,d,a,b\,\big\rangle + 8\big|\,d,a,b\,\big\rangle + 6\big|\,d,b,a\,\big\rangle + 12\big|\,d,a,b\,\big\rangle. \\
& = & 7\big|\,a,b,d\,\big\rangle + 21\big|\,a,d,b\,\big\rangle + 8\big|\,b,a,d\,\big\rangle + 26\big|\,b,d,a\,\big\rangle + 32\big|\,d,a,b\,\big\rangle + 6\big|\,d,b,a\,\big\rangle.
\end{array}$$

We can now iterate and start the second round, in essentially the same way. At first positions we now get:

$$\mathcal{N}\big(hd\big)(\varphi_1) \overset{(5)}{=} 28\big|\,a\,\big\rangle + 34\big|\,b\,\big\rangle + 38\big|\,d\,\big\rangle.$$

Still no-one has an absolute majority, and a has the lowest number. Hence this candidate a is deleted in this second step, giving as next multiset:

$$\varphi_2 \coloneqq \mathcal{N}\big(del_a\big)(\varphi_1) \overset{(5)}{=} 41\big|\,b,d\,\big\rangle + 59\big|\,d,b\,\big\rangle.$$

At this stage things begin to become clearer. We still start the third round by computing the first preferences:

$$\mathcal{N}\big(hd\big)(\varphi_2) \overset{(5)}{=} 41\big|\,b\,\big\rangle + 59\big|\,d\,\big\rangle.$$

We now have an absolute majority for d and can declare that this candidate is the winner. We see how functoriality of $\mathcal{N}$ does the work for us.

5 De Borda counts

The 18th-century French mathematician Jean-Charles de Borda devised several counting mechanisms. We elaborate one such method, often called the *modified* De Borda count — but we drop this qualification. This counting method can be described abstractly and efficiently using that multisets form a *monad*. This means that it comes with two special operations, which we call unit and flatten. They satisfy certain equations, which are not directly relevant here.

Definition 5.1 *For an arbitrary set X there are functions* unit: $X \to \mathcal{N}(X)$ *and* flat: $\mathcal{N}\big(\mathcal{N}(X)\big) \to \mathcal{N}(X)$ *given as:*

$$unit(x) = 1|x\rangle \qquad flat\Big(\sum_i n_i \,\big|\, \varphi_i\big\rangle\Big) = \sum_i n_i \cdot \varphi_i.$$

This last expression uses (repated) addition on multisets, where $n \cdot \varphi = \varphi + \cdots + \varphi$.

The unit map turns an element into a singleton multiset. That is easy. The flatten map 'flattens' a multiset of multisets into a multiset. What actually happens is not so complicated, as illustrated in:

$$\begin{aligned}
&\mathit{flat}\Big(2\,\big|\,3|a\rangle+4|b\rangle\big\rangle+3\,\big|\,5|a\rangle+6|b\rangle\big\rangle\Big)\\
&= \Big(3|a\rangle+4|b\rangle\Big)+\Big(3|a\rangle+4|b\rangle\Big)+\Big(5|a\rangle+6|b\rangle\Big)+\Big(5|a\rangle+6|b\rangle\Big)+\Big(5|a\rangle+6|b\rangle\Big)\\
&= 21|a\rangle+26|b\rangle.
\end{aligned}$$

We explain the De Borda count also via an example. We actually reuse the situation described in Table (6). A preference sequence $\langle x_1,x_2,x_3,x_4\rangle$ is now counted in a weighted manner, like in Example 2.3. Let's say the first preference counts for four, the second for three, the third for two and the fourth for one. This means that we interprete the preference sequence $\langle x_1,x_2,x_3,x_4\rangle$ as a multiset $4|x_1\rangle+3|x_2\rangle+2|x_3\rangle+1|x_4\rangle$. The whole table in (6) is then interpreted as a multiset over multisets. The election outcome is now obtained via flattenening. Explicitly:

$$\begin{aligned}
&\mathit{flat}\Big(7\,\big|\,4|a\rangle{+}3|b\rangle{+}2|d\rangle{+}1|c\rangle\big\rangle+2\,\big|\,4|a\rangle{+}3|c\rangle{+}2|d\rangle{+}1|b\rangle\big\rangle\\
&\qquad+7\,\big|\,4|a\rangle{+}3|d\rangle{+}2|c\rangle{+}1|b\rangle\big\rangle+8\,\big|\,4|b\rangle{+}3|a\rangle{+}2|c\rangle{+}1|d\rangle\big\rangle\\
&\qquad+7\,\big|\,4|b\rangle{+}3|c\rangle{+}2|d\rangle{+}1|a\rangle\big\rangle+19\,\big|\,4|b\rangle{+}3|d\rangle{+}2|c\rangle{+}1|a\rangle\big\rangle\\
&\qquad+12\,\big|\,4|c\rangle{+}3|a\rangle{+}2|d\rangle{+}1|b\rangle\big\rangle+12\,\big|\,4|d\rangle{+}3|a\rangle{+}2|b\rangle{+}1|c\rangle\big\rangle\\
&\qquad+8\,\big|\,4|d\rangle{+}3|a\rangle{+}2|c\rangle{+}1|b\rangle\big\rangle+6\,\big|\,4|d\rangle{+}3|b\rangle{+}2|c\rangle{+}1|a\rangle\big\rangle\\
&\qquad+12\,\big|\,4|d\rangle{+}3|c\rangle{+}2|a\rangle{+}1|b\rangle\big\rangle\Big)\\
&= 240|a\rangle+240|b\rangle+226|c\rangle+294|d\rangle.
\end{aligned}$$

Candidate d thus wins.

In this example we have mapped a sequence $\langle x_1,x_2,x_3,x_4\rangle$ to a multiset $4|x_1\rangle+3|x_2\rangle+2|x_3\rangle+1|x_4\rangle$. There are alternative ways to do this. For instance, in the Eurovision Song contest, each participating country votes by providing a list of ten (other) countries $\langle x_1,\ldots,x_{10}\rangle$. They are not counted as $10|x_1\rangle+9|x_2\rangle+\cdots+1|10\rangle$, but as $12|x_1\rangle+10|x_2\rangle+8|x_3\rangle+7|x_2\rangle+\cdots+1|10\rangle$. This gives rise to the famous phrase: *the twelve points go to …* The next time you hear it you may wish to think of the monad at work.

6 Counting single transferable votes

We now consider a situation where voters can vote via an ordered list of candidates, where, for simplicity, we again assume that these lists all have the same length. Now the goal is not to find a single winner, like in Section 4, but to fill multiple seats. Thus we need to determine multiple winners. If there are N votes and S seats, then a candidate needs at least $\frac{N}{S}$ votes to get a seat. This fraction is called the 'quota'. It is computed below in a slightly different manner, in so-called Droop style. A candidate may have received more votes than the quota. The idea with single transferable vote counting is that these 'surplus' votes are not lost, but are redistributed according to the next preferences on the list with the chosen candidate. This redistribution happens via a 'transfer value' fraction. We compute it below via the so-called 'weighted Gregory method' [8] that is closely related to the vote counting in the Western Australian upper house[2]. It is a priori not obvious how to do this transfer, but the multiset perspective

---

[2] For further details, see the website of the Australian Electoral Commission: https://www.parliament.wa.gov.au/WebCMS/WebCMS.nsf/resources/file-31-proportional-representation-lc/$file/Sheet 31 - Proportional Representation in the Legislative Council.pdf

steers us in the right direction.

This process will be illustrated below. The surplus leads to a fraction of the votes that need to be redistributed, so at this stage we need to relax the concept of multiset and allow fractions in instead of only natural numbers as multiplicities — as in Notation 2.2.

Notation 6.1 *For a set X, we now write $\mathcal{M}(X)$ for the set of multisets $q_1|x_1\rangle + \cdots + q_K|x_K\rangle$, with elements $x_i \in X$ and with non-negative fractions $q_i \in \mathbb{Q}_{\geq 0}$ as multiplicities. This $\mathcal{M}$ is also a functor, and even a monad.*

There is another new element that we use below, namely subtraction $\psi - \varphi$ of multisets. This works in the obvious way, elementwise, like for addition. We make sure that no negative multiplicities arise.

For the example of single transferable votes, we assume that there are five candidates a, b, c, d, e and a table with 250 vote sequences of the following form.

	20	23	27	23	7	14	13	17	12	13	7	14	9	15	16	20
1^{st}	a	a	a	a	b	b	c	c	c	d	d	d	e	e	e	e
2^{nd}	b	b	c	e	a	e	a	d	e	b	c	e	a	b	c	d
3^{rd}	c	d	e	b	d	a	d	b	a	c	b	a	c	c	b	b
4^{th}	e	e	d	c	e	d	e	a	d	e	e	c	b	d	a	a
5^{th}	d	c	b	d	c	c	b	e	b	a	a	b	d	a	d	c

(8)

This table translates into the following multiset $\varphi_0 \in \mathcal{N}[250]\big(\mathcal{L}(\{a,b,c,d,e\})\big) \subseteq \mathcal{M}\big(\mathcal{L}(\{a,b,c,d,e\})\big)$.

$$\begin{aligned}
\varphi_0 \;=\; & 20\big|a,b,c,e,d\big\rangle + 23\big|a,b,d,e,c\big\rangle + 27\big|a,c,e,d,b\big\rangle + 23\big|a,e,b,c,d\big\rangle \\
& + 7\big|b,a,d,e,c\big\rangle + 14\big|b,e,a,d,c\big\rangle + 13\big|c,a,d,e,b\big\rangle + 17\big|c,d,b,a,e\big\rangle \\
& + 12\big|c,e,a,d,b\big\rangle + 13\big|d,b,c,e,a\big\rangle + 7\big|d,c,b,e,a\big\rangle + 14\big|d,e,a,c,b\big\rangle \\
& + 9\big|e,a,c,b,d\big\rangle + 15\big|e,b,c,d,a\big\rangle + 16\big|e,c,b,a,d\big\rangle + 20\big|e,d,b,a,c\big\rangle.
\end{aligned}$$

In this illustration we assume that there are $S_0 = 3$ seats to be filled. The Droop quota function is defined, on arbitrary arguments, as:

$$qu(\varphi, S) \;:=\; \left\lfloor \frac{\|\varphi\|}{S+1} \right\rfloor + 1.$$

We go through the following iterative steps.

1. Starting form the multiset φ_0 and the number of seats $S_0 = 3$ we compute $qu\big(\varphi_0, S_0\big) = \lfloor \frac{250}{4} \rfloor + 1 = \lfloor 62.5 \rfloor + 1 = 63$. We then check if there is a candidate who has at least 63 votes as first preference. We do this, like in Section 4, via functoriality of $\mathcal{M}$ applied to the the head-of-list function hd. This gives:

   $$\mathcal{M}\big(hd\big)(\varphi_0) \overset{(5)}{=} 93|a\rangle + 21|b\rangle + 42|c\rangle + 34|d\rangle + 60|e\rangle.$$

   We see that only candidate a has more than 63 first preference votes, and thus gets a seat — at this stage. There are $93 - 63 = 30$ surplus votes for a that need to be divided. Formulated, differently, we need to subtract from φ_0 the fraction $\frac{63}{93}$ of the sequences that start with a. This fraction of the votes was used up for the seat of a. This subtraction takes the form:

   $$\varphi_0 - \frac{63}{93} \cdot \Big(20\big|a,b,c,e,d\big\rangle + 23\big|a,b,d,e,c\big\rangle + 27\big|a,c,e,d,b\big\rangle + 23\big|a,e,b,c,d\big\rangle\Big).$$

We obtain the new multiset of votes φ_1 by deleting candidate a from this difference:

$$\begin{aligned}\varphi_1 &\coloneqq \mathcal{M}\big(del_a\big)\Bigg(\varphi_0 - \frac{63}{93}\cdot\Big(20\big|a,b,c,e,d\big\rangle + 23\big|a,b,d,e,c\big\rangle + 27\big|a,c,e,d,b\big\rangle + 23\big|a,e,b,c,d\big\rangle\Big)\Bigg)\\ &= 6.45\big|b,c,e,d\big\rangle + 14.42\big|b,d,e,c\big\rangle + 14\big|b,e,d,c\big\rangle + 17\big|c,d,b,e\big\rangle +\\ &\qquad + 13\big|c,d,e,b\big\rangle + 20.71\big|c,e,d,b\big\rangle + 13\big|d,b,c,e\big\rangle + 7\big|d,c,b,e\big\rangle +\\ &\qquad + 14\big|d,e,c,b\big\rangle + 22.42\big|e,b,c,d\big\rangle + 25\big|e,c,b,d\big\rangle + 20\big|e,d,b,c\big\rangle.\end{aligned}$$

One sees that in this way candidate b gets quite a few extra votes because it is listed as second in $20+23=43$ cases where people had a has first preference. Of all of those, a 'transfer' fraction of $1-\frac{63}{93}=\frac{30}{93}$ with b as first preference appears in φ_1. Candidates c and e also get additional (first preference) votes, but not d.

The new multiset φ_1 contains fractions as multiplicities, and thus forms an element of the set $\mathcal{M}\big(\mathcal{L}(\{a,b,c,d,e\})\big)$, see Notation 6.1. One may check that the size of the multiset φ_1 is a natural number, namely $250-63=187$. This is the number of (as yet) unused votes. The number of available seats at this stage is $S_1 \coloneqq S_0 - 1 = 2$.

In our example, only one candidate has reached the quota. There could be more. In that case, for each winning candidate x, the sequences starting with x are subtracted from φ_0, with corresponding fractions, given by the quota divided by the number of first preference votes for x.

2. In the next step we (happen to) have the same quota, but now computed as $qu\big(\varphi_1,S_1\big) = \lfloor\frac{187}{3}\rfloor + 1 = \lfloor 62.333\rfloor + 1 = 63$. The first preferential votes are at this stage:

$$\mathcal{M}\big(hd\big)(\varphi_1) \;=\; 34.87|b\rangle + 50.71|c\rangle + 34|d\rangle + 67.42|e\rangle.$$

We see that candidate e now wins a seat, with a small surplus of $67.42-63$. We adapt the number of available seats to $S_2 \coloneqq S_1 - 1 = 1$, and we form the next successor multiset of votes:

$$\begin{aligned}\varphi_2 &\coloneqq \mathcal{M}\big(del_e\big)\Bigg(\varphi_1 - \frac{63}{67.42}\cdot\Big(22.42\big|e,b,c,d\big\rangle + 25\big|e,c,b,d\big\rangle + 20\big|e,d,b,c\big\rangle\Big)\Bigg)\\ &= 7.92\big|b,c,d\big\rangle + 28.42\big|b,d,c\big\rangle + 1.64\big|c,b,d\big\rangle + 50.71\big|c,d,b\big\rangle + 14.31\big|d,b,c\big\rangle + 21\big|d,c,b\big\rangle.\end{aligned}$$

This multiset has size $250-63-63=124$.

3. In the next round we get again as quota: $qu\big(\varphi_2,S_2\big) = \lfloor\frac{124}{2}\rfloor + 1 = \lfloor 62\rfloor + 1 = 63$. The first preferences are at this stage:

$$\mathcal{M}\big(hd\big)(\varphi_2) \;=\; 36.34|b\rangle + 52.35|c\rangle + 35.31|d\rangle.$$

There is no winner. Then, candidate b with the lowest number of votes is removed from the current multiset φ_2, giving as next multiset:

$$\varphi_3 \coloneqq \mathcal{M}\big(del_b\big)\big(\varphi_2\big) \;=\; 60.27\big|c,d\big\rangle + 63.73\big|d,c\big\rangle.$$

4. We now trivially get:

$$\mathcal{M}\big(hd\big)(\varphi_3) \;=\; 60.27\big|c\big\rangle + 63.73\big|d\big\rangle,$$

with quota $qu\big(\varphi_3,S_2\big) = 63$, so that candidate d gets the last seat.

7 Conclusions

This paper illustrates (and demonstrates) the usefulness of multisets, and of their categorical properties, in vote counting algorithms that are used in practice. This is applied category theory in elections. The paper is not 'deep' from a categorical perspective, but it introduces categorical techniques in a new field where they have not been identified before.

We believe that this usage of multisets brings clarity to an area where transparency and correctness are important. There is more work to do, both at a theoretical level and at a more practical level.

- This paper concentrates on representing the counting algorithms in terms of multisets. This multiset representation can now be used to prove basic properties about these algorithms. For instance, a first requirement is stability of the outcome under permutation of votes. This is automatic when one uses multisets, since the order of their elements (votes) is irrelevant.
- We have used various simplifying assumptions that may not hold in practice, such as: all voters express lists of candidate preferences of the same length. Our descriptions can be adapted to situations where this fails, but they then have to deal with 'undefinedness', for instance when using the head-of-list operation. This is manageable, but makes the descriptions less smooth. Edge cases like a tied vote are also not covered here.
- Given the new abstract vote count algorithms presented here, it makes sense to develop reference implementations of the frequently used versions in programming languages that support multisets, like Haskell. One could then do a recount or a parallel count of earlier and upcoming elections. And of course, these multiset-based implementations could be used in actual, future elections.

References

[1] S. Awodey. *Category Theory*. Oxford Logic Guides. Oxford Univ. Press, 2006. `doi:10.1093/acprof:oso/9780198568612.001.0001`.

[2] M. Barr and Ch. Wells. *Category Theory for Computing Science*. Prentice Hall, Englewood Cliffs, NJ, 1990. Available from URL: `www.tac.mta.ca/tac/reprints/articles/22/tr22abs.html`.

[3] J.-C. de Borda. Mémoire sur les élections au scrutin. *Archives de l'Académie des sciences*, 1781.

[4] E. Cheng. *The Joy of Abstraction. An Exploration of Math, Category Theory, and Life*. Cambridge Univ. Press, 2022. `doi:10.1017/9781108769389`.

[5] K. Cho and B. Jacobs. The EfProb library for probabilistic calculations. In F. Bonchi and B. König, editors, *Conference on Algebra and Coalgebra in Computer Science (CALCO 2017)*, volume 72 of *LIPIcs*. Schloss Dagstuhl, 2017. `doi:10.4230/LIPIcs.CALCO.2017.25`.

[6] P. Emerson. The original Borda count and partial voting. *Social Choice and Welfare*, 40:353–358, 2013. `doi:10.1007/s00355-011-0603-9`.

[7] P. Emerson. *From majority rule to inclusive politics*. Springer, 2016. `doi:10.1007/978-3-319-23500-4`.

[8] D. Farrell and I. McAllister. *The Australian Electoral System: origins, variations and consequences*. UNSW Press, 2006.

[9] B. Jacobs. *Structured Probabilitistic Reasoning*. Cambridge Univ. Press, 2026, to appear. Preliminary version at: `http://www.cs.ru.nl/B.Jacobs/PAPERS/ProbabilisticReasoning.pdf`.

[10] T. Leinster. *Basic Category Theory*. Cambridge Studies in Advanced Mathematics. Cambridge Univ. Press, 2014. Available online via `https://arxiv.org/abs/1612.09375`.

[11] S. Mac Lane. *Categories for the Working Mathematician*. Springer, Berlin, 1971. `doi:10.1007/978-1-4757-4721-8`.

[12] P. Perrone. *Starting Category Theory*. World Scientific, Singapore, 2024. `doi:10.1142/13670`.

[13] B. Pierce. *Basic Category Theory for Computer Scientists*. MIT Press, Cambridge, MA, 1991. `doi:10.7551/mitpress/1524.001.0001`.

[14] H. Simons. *An Introduction to Category Theory*. Cambridge Univ. Press, 2011. `doi:10.1017/CBO9780511863226`.