DETERMINATION OF THE ELECTION RESULT

Product:	OSV2020 Kiesraad
Date of creation:	11.6.2009
Created by:	Joachim Nottebaum
Date of modification	01.10.20 16:49
Modified by:	Joachim Nottebaum
Version	7.3

Change History

The following table documents the major changes in version starting from version 5.0.

Date	Autor	Version	Change
15.-27.07.09	JON	5.1	Worked over section 3.6 (mostly 3.6.2), because assignment of priority seats to candidates may occur simultaneously.
28.07.09	JON	5.1	Corrected some errors in section 3.6
29.07.09	JON	5.1	Incorporated several of Jan's comments
17.08.09	JON	5.2	Added borough council (DR1 and DR2) elections
20.08.09	JON	5.2	Removed deposit refund
01.09.09	JON	5.2	Added drawing lots events that apply to P2-lists in section 6.2
11.09.09	JON	5.4	Handling total votes = 0 in section 2.3.1
17.09.09	JON	5.4	Prevent drawing lots in cosalg mode in step 8 of section 2.4.3
21.09.09	JON	5.4	3.6.3 Nomination of all remaining candidates, steps 4b, 11 and 16: Changed the condition from "$\phi_{i-1}(s)$ > $\#E_{i,3}(s)$ " to "$\phi_{i-1}(s) > 0$ and $\#E_{i,3}(s) = 0$".
24.09.09	JON	5.5	Removed empty step 14 in section 3.6.3, renumbered the following steps.
25.09.09	JON	5.7	Added and updated implementation notes
28.09.09	JON	5.8	Updated sections 7.4.2 "Steps of the algorithm" and 7.4.3 "Refererce of all implementation classes and interfaces"
28.09.09	JON	5.8	In section 2.4.4: added step numbers, added step 1, corrected sentence on who is considered for this step.

28.09.09	JON	5.8	Corrected section 7.1.3
28.09.09	JON	5.8	Added section 7.5 "Parts of the algorithm that are weakly backed up by the electoral law"
14.10.09	JON	5.10	Implementation notes in 2.3.5, 3.6.2, 3.6.6 improved due to remarks from SIG.
14.10.09	JON	5.10	Clarification in section 2.4.3, steps 8 and 9.
14.10.09	JON	5.10	Cited article P7 sub 1 and article P8 sub 3 in section 2.4.4.
14.10.09	JON	5.10	Cited article P7 sub 1 in section 2.4.5.
14.10.09	JON	5.10	Cited article P19a in dutch, added clarification in section 3.2.2.
14.10.09	JON	5.10	Clarification in section 5.1, steps 2 and 3.
21.01.10	JON	5.12	Added election to the senate.
13.04.10	JON	5.13	Section 2.4.1, step 4: List with no votes shall not receive a seat.
24.09.10	JON	5.14	Section 1.5: preferential barrier = 100% of KT at EK elections. Section 2.1: No fictitious seat distribution at EK elections. Section 6.2: Some drawing lots events are n/a to EK elections.
24.09.10	JON	5.14	Section 4.2.3: Divided this into two subsections. New Section 4.2.3.1 “Determination of the valid combined lists and their total number of votes for EK elections”
04./05.10.10	JON	5.14	Section 1.4.6: no combined lists Section 3.6.2: preferential barrier = KT, split step 3 into 3a and 3b, introduced B_g Sections 3.6.5: Replaced $C'_g(VS)$ by B_g Section 4.2.3.2: Removed step 1b-4 (fictitious seat distribution for EK election) Section 8: added B_g
05.10.10	JON	5.15	Section 3.6.2, Implementation notes after step 3b: Note on implementation of B_g added.

9.-15.1.14	JON	6.0	Islands council (ER) elections added. Elections to borough councils are now BC and GC elections. Changed regulations to borough councils (no list exhaustion for GC -> new parameter b_8, min. 50% of KT for seat by largest remainder for BC -> parameter b_3 is now a number). Removed DR1 and DR2.
28.01.14	JON	6.1	Undid some change for GC elections (no list exhaustion) -> parameter b_8, is no longer needed
11.11.14	RAR	6.2	Added the elections for water councils (Waterschappen) with election type AB1 in case < 19 seats and AB2 for 19 or more seats.
11.01.18	JON	6.5	Parameter b_3 min. 25% of KT for seat by largest remainder for BC.
06. - 23.03.20	JON	7.0	Remove combined lists, P4-Lists, P4.2-Lists and fictitious seat distribution, checking validity of combined lists, parameter b_1. Updates index and implementation notes.
02.06.20	JON	7.1	Central list submission for TK elections
02.06.20	JON	7.2	Accepted all tracked changes from Version 7.1, deleted some comments
02.- 25.06.20	JON	7.3	Use meaningful variables

1 PREFACE

1.1 INTRODUCTION

This document declares the determination of the election result, as defined in the electoral law, for direct elections. This concerns election of the House of Representatives (Tweede Kamer), the senate (Eerste Kamer), the provincial states, municipal councils, water councils and the European Parliament. Due to differences in the election process, the following elections are explained in more detail:

- House of Representatives
- The Senate
- Provincial states consisting of more than one electoral district
- Provincial states consisting of one electoral district, municipal and water councils with 19 or more seats.
- Municipal, island or water councils consisting of less than 19 seats
- Borough councils in Amsterdam (which consist of less than 19 seats)
- Borough councils in Rotterdam (which consist of less than 19 seats)
- European Parliament

The determination of the election result is divided into two parts: the assignment of seats and the nomination of elected candidates. Within these two parts several intermediate steps follow. This procedure leads to a scheme that is consulted when dealing with the different elections.

A. *Assignment of seats*

1. Determination of total number of votes per political party, total number of votes (turnout of voters) and the electoral quota.
2. Determination of the total number of votes.
3. Assignment of the seats to lists based on if they reached the electoral quota
4. Assignment of all residual seats
5. Reassignment of the seat distribution if one list obtained the absolute majority of all votes.
6. Modification of the seat distribution if a candidate list is exhausted.
7. Distribution of seats within groups of list.

B*. Nomination of elected candidates*

1. Nomination of candidates elected via preferential vote
2. Nomination of remaining elected candidates
3. Order of candidates on candidate lists

Starting point for all descriptions in the following is the receipt of the principal electoral committees' protocols by the central electoral committee, (respectively the principal electoral committee's protocol, in case of one electoral district). These protocols for each electoral district contain the number of votes per candidate and per list.

1.2 COLOUR GUIDE

This is a working document. To make it easier to use, parts of the text are marked with colours. This has the following meaning:

- Yellow text is taken from the Vaststellen_van_de_uitslag_van_verkiezingen_NL document provided by de kiesraad.
- Cyan text denotes passages that are incomplete or provisional.
- Turquoise text is used for different purposes:
  - to mark text that refers to other step numbers, chapter numbers or section numbers
  - to mark text that refers to drawing lots
  - to mark text where certain variables are used: $steptype_{step}$ and $param_{step}$.
- Green passages are specific for the named election kind.

1.3 MATHEMATICAL SYMBOLS AND DEFINITIONS

1.3.1 BOOLEAN LOGIC

Logical AND	$\wedge$
Logical OR	$\vee$
For all	$\forall$
Exists	$\exists$
Boolean values true and false.	true := 1
We use the integer numbers 0 and 1 as representation of the boolean values true and false.	false := 0
	$\mathbb{B}$:= { 0, 1 } = { true, false }

1.3.2 SETS

Empty set	$\varnothing$

Subset	$N \subset M \Leftrightarrow \forall\ n \in N : n \in M$
Function	f: $X \to Y$, $f(x)$ = ...
Zero-function	$f \equiv 0 \Leftrightarrow \forall x \in X$: $f(x) = 0$
Union of 2 sets	$N \cup M$,
Union of n sets	$\bigcup_{i=1}^{n} M_i$
Intersection of 2 sets	$N \cap M$
Intersection of n sets	$\bigcap_{i=1}^{n} M_i$
Set difference	$A \setminus B = \{ \mathbf{x} \in A \mid \mathbf{x} \notin B \}$
Natural number	$\mathbf{N_0}$ = {0, 1, 2, 3, ...}, $\mathbf{N}$ = {1, 2, 3, ...}
Integer	$\mathbf{Z}$ = {..., -2, -1, 0, 1, 2, ...}
Rational numbers (the set of all fractions of integers)	$\mathbf{Q}$
Real number	$\mathbf{R}$
Rounding off (floor-function)	$\lfloor x \rfloor$:= largest integer, that is $\leq x$
Cartesian product	$N \times M \coloneqq \{(n,m) \mid n \in N \wedge m \in M\}$ Set of all pairs (n,m) with $n \in N$ and $m \in M$
Tuple	$(x_1, x_2, ..., x_n)$ For a definition see for example http://en.wikipedia.org/wiki/Tuple.
Disjoint	Two sets A and B are disjoint if $A \cap B = \emptyset$.
Mutually disjoint	A set P of sets is mutually disjoint if $\forall\ p, p' \in P$: $p \neq p' \Rightarrow p \cap p' = \emptyset$.

Partitioning	Partitioning of a set M is a set P of subsets of M, that are mutually disjoined and whose union results in M, that is: $M = \bigcup_{p \in P} p$ and $\forall\ p, p' \in P$: $p \neq p' \Rightarrow p \cap p' = \varnothing$.
Power set	$\mathbf{P}(A) := \{ B \mid B \subset A \}$. The power set of a set A is the set of all subsets of A.

1.3.3 BINARY RELATIONS AND ORDER RELATIONS

Binary relation on a set A	In mathematics, a **binary relation** on a set A is a collection of ordered pairs of elements of A. In other words, it is a subset of the Cartesian product $A \times A$. (Source: http://en.wikipedia.org/wiki/Binary_relation) If $R \subset A \times A$ is a binary relation, we usually write $a\ R\ b$ instead of $(a, b) \in R$.
reflexive	A binary relation $R \subset A \times A$ is **reflexive** if $\forall\ a \in A : a\ R\ a$.
antisymmetric	A binary relation $R \subset A \times A$ is **antisymmetric** if $\forall\ a, b \in A : a\ R\ b$ and $b\ R\ a \Rightarrow a = b$.
transitive	A binary relation $R \subset A \times A$ is **transitive** if $\forall\ a, b, c \in A : a\ R\ b$ and $b\ R\ c \Rightarrow a\ R\ c$.
total	A binary relation $R \subset A \times A$ is **total** if $\forall\ a, b \in A : a\ R\ b$ or $b\ R\ a$.

Partial order	A binary relation is called a **partial order** if it is reflexive, antisymmetric and transitive (see http://en.wikipedia.org/wiki/Partial_order).
Total order	A binary relation is called a **total order** if it is antisymmetric, transitive and total (http://en.wikipedia.org/wiki/Strict_weak_ordering#Total_preorders). This also implies reflexivity. A total order is also called **linear order**. For order relations we often use the symbol "≤". So we write $a \leq b$. Given a total order on a finite set A, there is a **unique** way to **sort** the set A in an ascending order. This means if $n = \#A$, there is a unique way to write $A = \{a_1, a_2, ..., a_n\}$ with $a_1 \leq a_2 \leq ... \leq a_n$.
Preorder	A binary relation is called a **preorder** if it is reflexive and transitive (see http://en.wikipedia.org/wiki/Preorder)
Total preorder	A binary relation is called a **total preorder** if it is reflexive, transitive and total (http://en.wikipedia.org/wiki/Strict_weak_ordering#Total_preorders). Given a total preorder on a finite set A, we can **sort** the set A in an ascending order. This means if $n = \#A$, there is a way to write $A = \{a_1, a_2, ..., a_n\}$ with $a_1 \leq a_2 \leq ... \leq a_n$. But this must not be unique.
Induced (total) preorder	Given two sets A and B, a (total) preorder $\leq$ on B and an arbitrary function $f\colon A \to B$. Then f induces a (total) preorder R_f on the A by $a\ R_f\ a' :\Leftrightarrow\ f(a) \leq f(a')$
Inverse relation	Given a set A and binary relation R. The inverse relation $\bar{R}$ on A is defined by $a\ \bar{R}\ b :\Leftrightarrow\ a\ R\ b.$ It is easy to see that if R is a partial order / total order / preorder / total preorder, then the same holds for the inverse relation $\bar{R}$.

1.4 TERMS OF THE ELECTION

1.4.1 ELECTORAL DISTRICTS, PROVINCES

Electoral districts

Let *DISTRICTS* be the set of all electoral districts.

#DISTRICTS is the number of electoral districts.

Electoral district number

The electoral districts have unique index numbers starting from 1 to *#DISTRICTS*. So we have a one-to-one mapping function

$$idx : DISTRICTS \rightarrow \{ 1, 2, ..., \#DISTRICTS \}$$

that maps each electoral district to its unique index $idx(district) \in \{1, 2, ..., \#DISTRICTS\}$

Provinces

For the EK election, there are no electoral districts. The elements of *DISTRICTS* are the provinces and public bodies. Throughout this document we will still talk about electoral districts only, i.e. the term "electoral districts" in this document really refers to provinces and public bodies in the case of EK elections.

1.4.2 CANDIDATES

Candidates

Let *CANDIDATES* be the set of all candidates for one election.

#CANDIDATES is the number of all candidates.

Deceased candidates

Let $DEADCANDIDATES \subset CANDIDATES$ be the set of all candidates, whose death has been put on record.

1.4.3 LISTS

List

A list is a pair, consisting of an electoral district *district* and a tuple $t = (t_1, t_2, ..., t_{\#list})$ of candidates: $list = (district, t) = (district, (t_1, t_2, ..., t_{\#l}))$. *#list* is the number of candidates of the list *list*.

	Candidates of a candidate list are pair wise different, i.e. $t_i \neq t_j$ if $i \neq j$.
Set of candidates on a list	Let $\tilde{C}_{list}$:= { t_1, t_2, ..., $t_{\#list}$ } be the set of candidates nominated on the candidate list *list* (disregarding the order).
Set of all lists	Let *LISTS* = {$list_1$, $list_2$, ...} = set of all lists, authorised to participate in the election. *#LISTS* is the number of lists, authorised to participate in the election.
Position on a list	To find out the position on the list for a given candidate we define the function $position_{list}: \tilde{C}_{list} \rightarrow \mathbf{N}$, $position_{list}(t_i) := i$ for the t_i as defined above. This function is well-defined, because the candidates of a candidate list are pair wise different.
Electoral district number	Also for a candidate list *list* = (*district*, *t*) $\in$ *LISTS* we define the electoral district number by *idx*(*list*) := *idx*(*district*).

1.4.4 SET OF IDENTICAL LISTS, P2-LISTS

Set of identical lists	A **set of identical lists** is a subset of *LISTS* that fulfils certain conditions according to article H 11 of the electoral law. Among these conditions are: • A set of identical lists contains at least two lists. • The lists that are contained in the same set of identical lists belong to pair wise different electoral districts. • Two elements of a set of identical lists have the same candidates in the same order. • Two sets of identical lists are mutually disjoint, i.e. no list may belong to more than one set of identical lists. Let *SETSOILISTS* := { *setoilists* $\subset$ *LISTS* | *setoilists* is a set of identical lists} be the set of sets of identical lists.

According to article P 2 of the electoral law sets of identical lists are to be treated as being single lists. In this document both types are subsumed under the term **P2-list**.

Let $SINGLELISTS$:= { $singlelist \subset LISTS$ | $\#singlelist = 1$ and $singlelist$ is not a subset of any set of identical lists}. $SINGLELISTS$ contains one-element sets of lists, that do not belong to any set of identical lists.

So we define

$$P2LISTS := SETSOILISTS \cup SINGLELISTS$$

to be the set of all **P2-lists**. Then is $P2LISTS$ a partitioning of $LISTS$.

So if $p2list \in P2LISTS$ is a P2-list, then $p2list \neq \emptyset$ and

$$\forall\ list = (district, t), list' = (district', t') \in p2list : t = t'.$$

If $p2list \in P2LISTS$ and $\#p2list > 1$, then $p2list$ is a set of identical lists.

Set of candidates on a P2-list

Obviously $\tilde{C}_{list} = \tilde{C}'_{list}$ for all $list, list' \in p2list$. So let $\tilde{C}_{p2list} := \tilde{C}_{list}$ be the set of candidates nominated on the P2-list $p2list$, where $list \in p2list$ is an arbitrary element of $p2list$.

Position on a list

So we can define the position on the P2-list for a given candidate by the function

$$position_{p2list} : \tilde{C}_{p2list} \to \mathbf{N}\ ,\ position_{p2list}(c) := position_{list}(c)$$

where $list \in p2list$ is an arbitrary element of $p2list$.

Electoral district number

Also for a P2-list $p2list \in P2LISTS$ we define the electoral district number as the minimum of the electoral district numbers of the containing candidate lists:

$$idx(p2list) := \min \{ idx(list) \mid list \in p2list \}.$$

1.4.5 LIST GROUPS, P3-LISTS

Set of all list groups

A list group is a subset of $LISTS$ that fulfils certain conditions according to article H 11 of the electoral law. Among these conditions are:

- The lists that are contained in the same list group belong to pair wise different electoral districts.

Article P 3 of the electoral law states that a group of lists is to be treated as a single list for the purposes of determining the number of seats to be allocated to it. In this document the three terms:

- Group of lists
- Set of identical lists not belonging to a group of lists
- Independent list not belonging to a group of list

are subsumed under the term **P3-list**.

- Two list groups are disjoint, i.e. no list may belong to more than one list group.
- A list group is the union of at least two P2-lists.

Let $LISTGROUPS$ be the set of all list groups that are authorized to participate in the election.

Due to the definition of a list group in the electoral law it is for a P2-list only possible to belong entirely to a list group or not at all (regardless if the P2-list is a set of identical lists or if it is composed of only one element (i.e. one candidate list)). Hence:

$$\forall\ p2list \in P2LISTS, group \in LISTGROUPS: p2list \subset group \vee p2list \cap group = \varnothing$$

Let $GROUPS''$:= { $p2list \in P2LISTS$ | $p2list$ is not subset of a list group }.

So $GROUPS''$ "contains" all sets of identical lists and independent lists that do not belong to a list group.

Let $P3LISTS := LISTGROUPS \cup GROUPS''$. Then $P3LISTS$ is the set of all P3-lists.

$P3LISTS$ is also a partitioning of $LISTS$.

For every $p3list \in P3LISTS$ let

$$P2LISTS_{p3list} = \{p2list \in P2LISTS \mid p2list \subset p3list\}$$

be the set of identical lists or independent lists that belong to $p3list$.

Then $P2LISTS_{p3list}$ is a partitioning of $p3list$. If and only if $\#P2LISTS_{p3list} > 1$, $p3list$ is a group of lists.

The lists comprising a group of lists belong to different electoral districts.

More generally for all P3-lists $p3list \in P3LISTS$ holds the following: If $(district, t), (district', t') \in p3list$, then $district \neq district'$ or $(district, t) = (district', t')$.

1.4.6 VOTES

Votes

The number of votes per candidate and electoral district makes up the election result. This is a function

$$votes: DISTRICTS \times CANDIDATES \rightarrow \mathbf{N}_0$$

A candidate only receives votes in electoral district, in which he is nominated:

$votes(district, candidate) > 0 \Rightarrow \exists\ i \in \mathbf{N}$, $list = (district, (t_1, t_2, ...)) \in LISTS$ with $candidate = t_i$

In each electoral district a candidate may be nominated only on one candidate list and has a unique position on that list. So if

$\exists\ i \in \mathbf{N}$, $list = (district, (t_1, t_2, ...)) \in LISTS$ with $candidate = t_i$

we can define the number of votes of candidate *candidate* on list *list* as

$votes_{list}(candidate) := votes(district, candidate)$.

For candidates $candidate \in CANDIDATES$ that are not nominated on the candidate list *list*, define

$votes_{list}(candidate) := 0$.

Votes on a P2-list

Define the number of votes of a candidate $candidate \in CANDIDATES$ on a P2-list $p2list \in P2LISTS$ by

$$votes_{p2list}(candidate) := \sum_{list \in p2list} votes_{list}(candidate).$$

Votes on a P3-list

Define the number of votes of a candidate $candidate \in CANDIDATES$ on a P3-list $p3list \in P3LISTS$ by

	$votes_{p3list}(candidate) := \sum_{p2list \in P2LISTS_{p3list}} votes_{p2list}(candidate)$.
Total number of votes of a candidate.	For each candidate *candidate* ∈ *CANDIDATES* the total number of votes is $votes(candidate) := \sum_{district \in DISTRICTS} votes(district, candidate)$.

1.4.7 VOTES CAST AND VOTE VALUES

In this section we introduce a differentiation between the term "votes", that was introduced in the previous section, and the term "votes cast". For most elections, "votes" and "votes cast" are the same, so a differentiation would not be necessary. A differentiation is necessary only for the the EK election.	
The **votes cast** for a candidate are usually determined by counting the ballots. One valid, non-blanc ballot corresponds to one vote cast.	The number of **votes cast** per candidate and electoral district makes up the election result. This is a function *votes'*: *DISTRICTS*×*CANDIDATES* → $\mathbf{N_0}$ A candidate may only have a non zero number of votes cast in an electoral district, in which he is nominated: *votes'*(*district*, *candidate*) > 0 ⇒ ∃ *i* ∈ **N,** *list* = (*district*, (t_1, t_2, ...)) ∈ *LISTS* with *candidate* = t_i
For EK elections the number of votes is determined by multiplying the votes cast with the **vote value**. The vote value is a positive integer that depends on the electoral district only. Article U 2 of the electoral law describes how the vote values shall be determined for EK elections. The calculation of the vote value is not subject of this document, so for the purpose of this document the vote values are given, for example entered into the system by the user.	The vote value is a function *ω*: *DISTRICTS* → **N**. For all elections other that the EK election, *ω*(*district*) = 1 for all *district* ∈ *DISTRICTS*.
The number of **votes** is the most important parameter for the determination of the election result.	The number of **votes** per candidate and electoral district is determined as follows: *votes*(*district*, *candidate*) := *votes'*(*district*, *candidate*) · *ω*(*district*)

For most elections, all votes cast have the same value, i.e. they should have the same effect on the result of the election. In that case one "vote cast" counts as one "vote". This is not true for the EK elections.	So, for all elections other that the EK election $votes(district, candidate) = votes'(district, candidate)$.

1.4.8 SEATS

Seats	Let $n \in \mathbf{N}$ be the total number of seats to assign within an election.

1.5 DIFFERENCES BETWEEN DIFFERENT KINDS OF ELECTIONS

The specified program shall support the following different kinds of elections. They correspond to the denoted values for the variable $electiontype$:

- European Parliament (EP, $electiontype = 1$)
- Senate (EK, $electiontype = 6$)
- House of Representatives (TK, $electiontype = 2$)
- Provincial states with more than one electoral district (PS2, $electiontype = 3$)
- Provincial states with one electoral district (PS1, $electiontype = 4$)
- Municipal and water councils with 19 or more seats (GR2, AB2, $electiontype = 4$)
- Municipal councils, water and island councils with less than 19 seats (GR1, AB1, ER, $electiontype = 5$)
- Borough councils in Amsterdam (BC, $electiontype = 7$)
- Borough councils in Rotterdam (GC, $electiontype = 8$)

The algorithms for these kinds of elections differ in the following criteria:

	EP	**EK**	**TK**	**PS2**	**PS1/ GR2/ AB2**	**GR1/ AB1/ ER**	**BC**	**GC**
Criteria $electiontype$ **:=**	1	6	2	3	4	5	7	8
Multiple electoral districts	yes	yes	yes	yes	no	no	no	no
Different nomination in different electoral districts	no	yes	yes	yes	no	no	no	no
Method for distribution of residual seats amongst P3-lists etc. (LA = largest average, LR = largest remainder)	LA	LA	LA	LA	LA	1. LR 2. LA	1. LR 2. LA	1. LR 2. LA
Minimum for taking part in residual seat distribution (KT = electoral quota)	KT		KT					
Minimum for taking part in residual seat distribution by largest remainder						75% of KT	25% of KT	75% of KT
Restriction to one seat by largest average						yes	yes	yes
Preferential barrier (*) For EK elections, the number of votes must **reach**, while for all other elections the number of votes must **exceed** the given percentage of the electoral quota for a candidate to receive a priority seat.	10%	100% (*)	25%	25%	25%	50% for GR1, ER 25% for AB1	50%	100%
Absolute majority regulation in seat distribution amongst P3-lists	yes	no	yes	yes	yes	yes	yes	yes
Vote values		yes						

Please note that the algorithms for provincial states with one electoral district (PS1) and for municipal, water (AB2) and borough councils with 19 or more seats (GR2) are the same. Therefore, they correspond to the same value $electiontype = 4$. Also, the algorithms for municipal, water and island councils with less than 19 seats (GR1, AB1 and ER) are the same, only the preferential barrier varies. Therefore, they correspond to the same value $electiontype = 5$.

Explanation:

a. **Multiple electoral districts:** "yes" means that the election takes place in more than one electoral district. "no" means that there is only one electoral district and all P2-lists and all P3-lists contain only one candidate list.

b. **Different nomination in different electoral districts:** "yes" means that political parties may nominate different lists in different electoral districts.

c. **Method for distribution of residual seats amongst P3-lists etc.:** "LA" means, that the distribution of residual seats to P3-lists is performed by the method of largest average. "1. LR 2. LA" means that the distribution of residual seats to P3-lists is performed by the method of largest remainder first. If residual seats remain after that, the remaining residual seats are distributed by the method of largest average.

d. **Minimum for taking part in residual seat distribution (KT = electoral quota):** "KT" means, that lists that have received less votes than the electoral quota do not take part in the distribution of residual seats to P3-lists.

e. **Minimum for taking part in residual seat distribution by largest remainder:** "75% of KT" means, that only lists that have received at least 75% of the electoral quota take part in the residual seat distribution by largest remainder. This shall not imply any restriction if later seats are distributes by largest average.

f. **Restriction to one seat by largest average:** "yes" means, that in the distribution of residual seats to P3-lists by largest average, there is a restriction to one seat per list. This restriction is dropped after each P3-list taking part in the distribution has received one residual seat by largest average.

g. **Preferential barrier:** This means that the preferential barrier for the respective election is 10%, 25%, 50% or 100% of the electoral quota respectively.

h. **Absolute majority regulation in seat distribution amongst P3-lists:** In distribution of residual seats to P3-lists the regulation of article P 9 electoral law applies.

i. **Vote values:** The number of votes cast for the candidates in the electoral district is multiplied by the vote value to determine the votes of the candidates in the electoral district. "yes" in this row means, that the vote value is determined according to article U 2 of the electoral law for this elections. Vote values may be different from 1 in that case. In all other case, the vote value equals to 1.

Note that for the TK and PS2 elections where different nomination in different electoral districts is possible, there are some regulations in the nomination of elected candidates that require the algorithm to go back to an earlier step. This earlier step is the seat distribution to P2-lists within a P3-list. To handle these regulations, we will memorize from the seat distribution to P2-lists within a P3-list not only the result (i.e. which P2-list

receives how many seats), but also the sequence of the assignment and in addition to that, the potential sequence of assignment, if "suddenly" more residual seats where available.

2 SEAT DISTRIBUTION ALGORITHMS

2.1 VARIATIONS OF SEAT DISTRIBUTION ALGORITHMS

Parts of the algorithms of the distribution of seats follow a common pattern that comprises the following steps:

- A quota is calculated (electoral quota, list group electoral quota etc.). The quota is given by dividing the total number of votes in this distribution by the total number of seats in this distribution.
- First distribution by attaining a given quota.
- Residual seat distribution by varying methods.
- Drawing lots in case of equal remainders or averages. Optional.
- Considering absolute majority (optional)
- Considering list exhaustion.

Seat distribution according to this common pattern is applied in the following cases / elections:

- Seat distribution to P3-lists for EP and TK
- Seat distribution to P3-lists for EK
- Seat distribution to P3-lists for PS2, PS1, AB2 and GR2
- Seat distribution to P3-lists for GR1, AB1 and ER, BC and GC
- Seat distribution to P2-lists within a list group (all elections)

Even though the pattern is the same in all the above cases, there are still significant differences between the named distribution algorithms. Still it is our aim to describe the algorithms that follow this pattern only once in common to avoid duplication both in the specification and in the code. In order to describe these algorithms in common, we need to know exactly what the algorithms have in common and in how far they differ. The following table gives a complete list of all differences between the listed algorithms:

Description	para-meter	P3-distr. EP/ TK	P3-distr. EK	P3-distr. PS1/ AB2/ GR2/ PS2	P3-distr. AB1/ GR1/ ER/ GC	P3-distr. BC	most P2-distr. *	P2-distr. EK/ TK/ PS2
Electoral quota is minimum for taking part in residual seat distribution	$param_2$	true	false	false	false	false	false	false
A given percentage of electoral quota is minimum for taking part in residual seat distribution by largest remainder	$param_3$	0%	0%	0%	75%	25%	0%	0%
Include largest remainder	$param_4$	false	false	false	true	true	true	true
Restriction to one seat by largest average	$param_5$	false	false	false	true	true	false	false
Absolute majority regulation	$param_6$	true	false	true	true	true	false	false
Continued seat distribution in list groups	$param_7$	false	false	false	false	false	false	true

* all P2-distributions except for EK, TK and PS2 elections.

In the common description of all algorithms we need to describe all the steps that may occur, even if they occur only in some of the algorithms. For example the consideration of the absolute majority will be part of the common description but will be omitted in the algorithms for the distribution of seats to P2- or P3-lists. To activate or deactivate certain parts of the common description, we define seven parameters that control if a certain part of the description will be reached in the respective algorithm. These parameters are named $param_2$ to $param_7$. All of them but $param_3$ are flag (Boolean parameters), $param_3$ is a numerical parameter. The above table shows, which values the parameters have for the respective algorithms.

Explanation of the parameters:

- $param_1$**:** is no longer used.
- **Electoral quota is minimum for taking part in residual seat distribution (**$param_2$**):** If the flag is true, only lists that have reached the quota take part in the steps that follow the first assignment. This flag is true only for the distribution to P3-lists in the EP and TK elections.
- **A given percentage of electoral quota is minimum for taking part in residual seat distribution by largest remainder (**$param_3$**):** If the value is greater than 0 and a distribution by largest remainder takes place, only those list may take part, that have reached at least the given percentage of the electoral quota. This value is 75% for the distributions of seats to P3-lists in the GR1, AB1, ER and GC elections and 25% in the BC elections.

- **Include largest remainder ($param_4$):** First please note that each of the algorithms may include steps that distribute seats by largest average. There is no flag necessary to omit this kind of steps. But the distribution of residual seats by largest remainder is explicitly excluded from some algorithms - in this case the flag $param_4$ is false. So this flag is true only in the distribution of seats to P2-lists for all elections and in the distribution of seats to P3-lists in the GR1, AB1, ER, BC and GC elections.
- **Restriction to one seat by largest average ($param_5$):** If the flag is true, in the distribution of residual seats by largest average, there is a restriction to one seat per list. This restriction is dropped after each list taking part in the distribution has received one residual seat by largest average. This flag is true only in the distribution of seats to P3-lists in the GR1, AB1, ER, BC and GC elections.
- **Absolute majority regulation ($param_6$):** If the flag is true, the absolute majority regulation must be considered in the algorithm. This is the case only in the distribution of seats to P3-lists for all elections except EK election.
- **Continued seat assignment in list groups ($param_7$):** If the flag is true, the distribution algorithm must be enhanced such that it memorizes not only the result (i.e. which list receives how many seats), but also the sequence of the assignment and in addition to that, the potential sequence of assignment, if "suddenly" more residual seats where available. This flag is true only for the distribution to P2-lists within a P3-list in the EP, EK and TK elections. Note that the "normal" result of the algorithm is not changed by this flag in any way.

The following list shall give an informal overview of the steps that are contained in the common description of all these algorithms. The details of the steps as well as the conditions under which and the ordering in which they are executed are subject to the following chapter:

- Calculation of a quota.
- First assignment, based on attaining the quota. If $param_2$ is true, this step also excludes all lists from the following steps that did not reach the quota.
- Residual seat distribution by largest remainder. If $param_4$ = false, this is completely omitted. If $param_3 > 0$, only lists are considered here that have reached a certain percentage of the quota given by $param_3$.
- Residual seat distribution by largest average restricted to one seat per list. If $param_5$ = false, this is completely omitted.
- Residual seat distribution by largest average.
- Considering absolute majority. If $param_6$ = false, this is completely omitted.
- Considering list exhaustion.

2.2 DEFINITION OF THE GENERAL SEAT DISTRIBUTION ALGORITHM

In this section we define a general seat distribution algorithm that covers all the cases described previously. The algorithm is based on a given set of lists P. This may be an arbitrary finite set but will in our cases be a set of P3-lists, P2-lists or candidate lists, respectively.

This algorithm needs the following parameters

- The number of seats $seats \in \mathbf{N}$ that shall be distributed.
- The numbers of votes cast to the lists. This is given by the function $votes$: $P \to \mathbf{N_0}$. The set of all such functions is denoted by $\mathbf{N_0}^P$. So we can write $votes \in \mathbf{N_0}^P$.
- The number of candidates that may receive a vote for each list. This is again given by a function $noofcandidatsinlist$: $P \to \mathbf{N_0}$. So $noofcandidatsinlist \in \mathbf{N_0}^P$.
- A tuple of six parameters $param = (param_2, param_3, param_4, param_5, param_6, param_7) \in \mathbf{B} \times \mathbf{Q} \times \mathbf{B^4}$ that configure the described varying details of the algorithm. $param_1$ is no longer used.

The result of the algorithm is the seat distribution which is an element of $\mathbf{Z}^P$, i.e. a function $seatstolist$: $P \to \mathbf{Z}$. In fact as a consequence of the algorithm, all values of this function are non-negative.

The algorithm may include the drawing of lots. So strictly speaking it is not possible to define it as a mathematical function. But disregarding this we may write the algorithm A as a function

$$A: \mathbf{N} \times \mathbf{N_0}^P \times \mathbf{N_0}^P \times \mathbf{B} \times \mathbf{Q} \times \mathbf{B^4} \to \mathbf{Z}^P$$

where $seatstolist := A(seats, votes, noofcandidatsinlist, param) \in \mathbf{Z}^P$ is a function and for each list $p \in P$, $seatstolist(p)$ is the number of seats assigned to list p in this algorithm.

In the following section we will define the details of how $seatstolist(p)$ shall be determined.

2.2.1 CONTINUED SEAT ASSIGNMENT IN LIST GROUPS

As stated before, the flag $param_7$ triggers if continued seat assignment in list groups is performed. In this case, i.e. if $param_7$ = true, the resulting function $seatstolist \in \mathbf{Z}^P$ of algorithm is the same as with $param_7$ = false. But the algorithm generates two additional results:

- Roll back sequence. This is a tuple $rbs = (rbs_1, rbs_2, ...)$ of previously unknown size of subsets of P. The elements of this tuple are the lists that receive a positive seat assignment in the respective step, but in the reverse order. I.e. rbs_1 contains the lists that received the last

seats in the algorithm, rbs_2 contains the lists that received the seats just before the last assignment etc.. Negative assignments (e.g. from list exhaustion) are not regarded.

- Roll forward sequence. This is a tuple $rfs = (rfs_1, rfs_2, ...)$ of previously unknown size of subsets of P. It is determined subsequent to the determination of the result function *seatstolist*. After the seat assignment should normally (i.e. if $param_7$ = false) be terminated, it is continued in a different mode. In this mode that we call cosalg-mode, the number of available seats is re-calculated to be equal to the total number of candidates. So, the seat distribution terminates when all lists are exhausted. Also, in the cosalg-mode, no drawing lots is performed. If drawing lots would be required, instead all list receive a seat. The elements of roll forward sequence are the lists that receive a positive seat assignment in the steps in the cosalg-mode.

The roll back sequence is needed in case article P 16 subsection 2 applies. The roll forward sequence is needed in case in the nomination of elected candidates on a P2-list there are still unassigned seats but no more unelected candidates. In that case the seat passes to another P2-list which will be taken from the roll forward sequence.

2.3 INTRODUCTION TO THE GENERAL SEAT DISTRIBUTION ALGORITHM

The core of the general seat distribution algorithm is an iteration. Some calculations must be performed before the first step of the iteration. Then, in each iteration step, seats may be assigned to lists, depending on what kind of iteration step is performed. If after an iteration step, certain conditions are met, the iteration is terminated. The sum of the seats assigned in the iteration steps define the result of the algorithm.

The details will be described in this section.

2.3.1 CALCULATIONS PRIOR TO SEAT ASSIGNMENT

stap	*bewerking*	*Formalization*
1	The number of votes for each list are added. The result of this addition is the **total number of votes**.	Define the total number of votes $totalvotes \in \mathbf{N}$ as $totalvotes := \max\{\sum_{p \in P} votes(p), 1\}$

1a	In case the total number of votes is 0, we replace it by 1 to avoid a division by zero. The electoral law does not mention this case because it is irrelevant for calculations with complete information about the votes cast for the candidates. If the calculation is started at a time where the numbers of votes are yet unknown or not entered in the software, this situation may still occur in praxis.	**Note:** The case that the above sum is zero must be handled as a special case. In that case we set *totalvotes* := 1. Leaving *totalvotes* = 0 would lead to *quota* = 0 and cause problems when dividing by *quota* in the first assignment. On the other hand, *totalvotes* = 1 leads to the following distribution: In the first assignment, no list receives a seat. In the residual seat distribution all seats are distributed evenly between all the lists.
2	The total number of votes is divided by the number of seats to be distributed. The result of the division is the **quota** for this seat distribution. *Implementation notes:*	Define the quota *quota* as $quota := totalvotes \,/\, seats$ This is implemented in the constructor `GeneralSeatDistributor()`.

2.3.2 OVERVIEW OF THE SEAT ASSIGNMENT

stap	*bewerking*	*Formalization*
1	In the following sections the details of the general seat distribution algorithm are specified. This assignment is carried out step by step.	Each step corresponds to a value of $step \in \mathbf{N}$.
2	In each step one or more seats are assigned or the seat distribution is modified in some other way.	The number of seats assigned to a list $p \in P$ in step *step* is given by $newseats_{step}(p)$. This defines for each $step \in \mathbf{N}$ a function $newseats_{step} : P \to \mathbf{Z}$ of the seats assigned in step *step*. We will see later how exactly this function is defined.

3 After each step we can calculate the **total number of seats assigned** to each list and the number of **residual seats**.

The function

$$seatstolist_{step} : P \to \mathbf{Z}$$

is defined by seatstolist_{step}\left(p\right):=\sum_{j=1}^{step}{newseats}_{j}\left(p\right)

$$seatstolist_{step}(p) := \sum_{j=1}^{step} newseats_j(p)$$

So $\mathrm{seatstolist}_{\mathrm{step}}(\mathrm{p})$ is the **total number of seats** assigned to a list $p \in P$ in the steps **up to step** *step*. The definition of the number of **residual seats** $residualseats_{step}$ will follow.

4 If certain conditions are met, the seat assignment may be terminated because further steps do not result in any further assignment of seats. This is usually the case when all seats are assigned.

What exactly these conditions are will be specified later.

We will see that there is a value $step \in \mathbf{N}$, such that $newseats_j \equiv 0$ for all $j \geq step$, i.e. $seatstolist_j = seatstolist_{step}$ for all $j \geq step$. For any such value $step \in \mathbf{N}$ we define the **resulting function** $seatstolist \in \mathbf{Z}^P$ **of the algorithm** by

$$seatstolist := seatstolist_{step}.$$

In cases where the cosalg-mode is switched on, the final "result" also contains assignments from the cosalg-mode, which is not desired. So *seatstolist* must be defined differently in that case:

(Please refer to the next section on how $steptype_{step}$ is defined.)

If there is a value $step \in \mathbf{N}$, such that $steptype_{step} = 12$, define

$$seatstolist := seatstolist_{step}.$$

It is clear that there may be at most one such value. If there is no such value, the earlier definition of *seatstolist* applies.

5

The roll back sequence is determined as follows:

Let

$$a := \min \{ step \in \mathbf{N} \mid steptype_{step} \in \{ 2, 11, 12 \} \}.$$

For each $step = 1, ..., a - 1$ define

$$rbs_{step} := \{ p \in P \mid z_{a\text{-}step}(p) > 0 \} \subset P.$$

So, the roll back sequence is the $(a - 1)$-tuple

$$rbs := (rbs_1, rbs_2, ..., rbs_{a\text{-}1}).$$

6

The roll forward sequence is determined as follows:

If $\{ step \in \mathbf{N} \mid steptype_{step} = 12 \}$ is empty, the cosalg-mode was never switched on and the roll forward sequence is empty: $rfs := (\varnothing)$. Otherwise let

$$a := \min \{ step \in \mathbf{N} \mid steptype_{step} = 12 \}.$$

$$b := \min \{ step \in \mathbf{N} \mid steptype_{step} \in \{ 2, 11 \} \}.$$

For each $step = 1, ..., b - a - 1$ define

$$rfs_{step} := \{ p \in P \mid z_{a+step}(p) > 0 \} \subset P.$$

So, the roll forward sequence is the $(b - a - 1)$-tuple

$$rfs := (rfs_1, rfs_2, ..., rfs_{b\text{-}a\text{-}1}).$$

Implementation notes:	This is implemented in the class `GeneralSeatDistributor`: • *step* is the attribute `_index` • $newseats_{step}(p)$ is stored in `_assignmentTracer._assignmentLists.get(p).get(_index - 1)` • $seatstolist_{step}(p)$ is stored in `_assignmentTracer._totalAssignment.get(p)`. Finally this is also the value of $seatstolist(p)$. The roll back sequence is calculated in `AssignmentTracer.getRollBackSequence()`. The roll forward sequence is calculated in `AssignmentTracer.getRollForwardSequence()`.

2.3.3 THE DIFFERENT KINDS OF SEAT ASSIGNMENT STEPS

stap	*bewerking*	*Formalization*
1	There are the following different kinds of steps. Note that depending on the parameters $param = (param_2, ..., param_7)$ some of the steps are completely omitted:	

stap	*bewerking*	*Formalization*
	a. first assignment (based on attaining the quota)	For each $step \in \mathbf{N}$, $steptype_{step} \in \mathbf{N}$ is defined to identify the kind of step as follows: a. first assignment: $steptype_{step} = 1$
	b. assignment of residual seats by the method of largest remainder (with or without drawing lots)	b. assignment of residual seats by the method of largest remainder: $steptype_{step} = 3$
	c. assignment of residual seats by the method of largest average restricted to one seat per list (with or without drawing lots)	c. assignment of residual seats by the method of largest average restricted to one seat per list: $steptype_{step} = 4$
	d. assignment of residual seats by the method of largest average (with or without drawing lots)	d. assignment of residual seats by the method of largest average: $steptype_{step} = 5$
	e. modification of the seat distribution, if a list attained the absolute majority of all votes	e. modification of the seat distribution, if a list attained the absolute majority of all votes: $steptype_{step} = 6$
	f. modification of the seat distribution, if a list is exhausted	f. modification of the seat distribution, if a list is exhausted: $steptype_{step} = 7$
	g. continued drawing lots in the assignment by the method of largest remainder	g. continued drawing lots in the assignment by the method of largest remainder: $steptype_{step} = 8$
	h. continued drawing lots in the assignment by the method of largest average restricted to one seat per list	h. continued drawing lots in the assignment by the method of largest average restricted to one seat per list: $steptype_{step} = 9$
	i. continued drawing lots in the assignment by the method of largest average	i. continued drawing lots in the assignment by the method of largest average: $steptype_{step} = 10$
2	Note that continued drawing lots is only required if in a previous residual seat assignment step, drawing more than one list by lot occurred.	
3	There are two technical kinds of steps where the seats distribution is not changed. These kinds of steps indicate that the iteration has come to an end, because further iteration does not result in any further assignment of seats:	

stap	*bewerking*	*Formalization*
	j. all lists exhausted	j. all lists exhausted: $steptype_{step} = 2$
	k. all seats were assigned	k. all seats were assigned: $steptype_{step} = 11$
4	There is another technical kind of steps where the cosalg-mode is switched on.	l. all seats were assigned, switch cosalg-mode on: $steptype_{step} = 12$
5	The details of these steps will be described below in section 2.4. Some kinds of steps may occur only once (a, e), some may occur more often (b, c, d, f, g, h, i). Some kinds of steps only add seats (a, b, c, d, g, h, i), f subtracts seats, e both adds and removes a seat.	
6	In the following description, we aim to treat all kinds of steps uniformly as far as possible. We will define conditions that can be checked at the beginning of the assignment process and after each step. These conditions determine, which kind of step is performed next or if the assignment process can be terminated. These conditions are described in detail in section 2.3.6.	
	Implementation notes:	The current step type is $steptype_{step}$ stored as `GeneralSeatDistributor._stepType`. `StepType` is an enum type, the `_id` of the `_stepType` is the value of $steptype_{step}$.

2.3.4 TERMINATION OF THE ITERATION

stap	*bewerking*	*Formalization*
1	The assignment of seats usually terminates if all seats are assigned to lists that have the required number of candidates, i.e. are not exhausted.	We will see later that if $steptype_{step} = 11$, we will have $steptype_{step} = 11$ also for all $j \geq step$. Also $steptype_{step} = 11$ implies $newseats_{step} \equiv 0$. So we also have $newseats_j \equiv 0$ also for all $j \geq step$. This means that we may terminate the calculation as soon as $steptype_{step} = 11$, because further iteration will not change the total seat assignment $seatstolist_{step}$ any more. More precisely, $seatstolist_j \equiv seatstolist_{step}$ for all $j \geq step$.
2	Another reason for terminating the calculation may happen in theory but is not mentioned in the electoral law: If all lists are exhausted, the lists may not receive any more seats than they have candidates.	Similarly, if $steptype_{step} = 2$, we have $steptype_j = 2$ and $newseats_{step} \equiv 0$ for all $j \geq step$. So we may terminate the calculation as soon as $steptype_{step} = 2$ for the same reason.
3		A proof that the algorithm does in fact terminate is given in section 2.3.7.
4	Also in the case of the seat assignment within list groups, when all seats are assigned, the iteration should terminate and the resulting function m can be determined. After that has happened, the algorithm switches to the cosalg-mode such that "suddenly" in the next step, new residual seats are available.	This can happen at most once if $param_7$ = true and the cosalg-mode is not yet switched on and otherwise the conditions for $steptype_{step} = 11$ are given. In that case, $steptype_{step} = 12$.
	Implementation notes:	This is implemented in `GeneralSeatDistributor` in the methods `calculate()`, `mainLoop()` and `isTerminationStep()`.

2.3.5 VARIABLES DEFINED IN EACH STEP

stap	*bewerking*	*Formalization*

1	Before each step, we determine the **kind of step** that is performed next.	The kind of the step *step* is given by $steptype_{step} \in \mathbf{N}$: For *step* = 1 we define $steptype_1 := 1$. For *step* > 1, $steptype_{step}$ is defined recursively in section 2.3.6 in terms of the variables $LISTSTAKINGPART_{step-1}$, $seatstolist_{step-1}$, $residualseats_{step-1}$, D_{step-1} and M_{step-1}.
2	In each step seats will be assigned or the seat distribution will be changed in some way.	$newseats_{step}$ is defined recursively in step *step*, depending on the step kind $steptype_{step}$. The details are described in section 2.4. This determines $seatstolist_{step}$ as defined in section 2.3.2.
3	This also determines the **number of residual seats** after the step.	We defined the **number of residual seats** $residualseats_{step} \in \mathbf{Z}$ after step *step* by residualseats_{step} := seats - \sum_{p ∈ P} seatstolist_{step} (p) $$residualseats_{step} := seats - \sum_{p \in P} seatstolist_{step}(p)$$ if $cosalg_{step}$ = false. If $cosalg_{step}$ = true, set residualseats_{step} := \sum_{p ∈ P} noofcandidatesinlist(p) - seatstolist_{step} (p) $$residualseats_{step} := \sum_{p \in P} noofcandidatesinlist(p) - seatstolist_{step}(p)$$ This way there are enough seats to distribute until all lists are exhausted.
4		Let $LISTSTAKINGPART_{step} \subset P$ be the set of **lists that take part** in the step *step* + 1. Please note that not all lists that take part in a step receive a seat. We define $LISTSTAKINGPART_0 := P$ because all lists take part in the first assignment. We define $LISTSTAKINGPART_{step} := LISTSTAKINGPART_{step-1}$ unless stated otherwise in the description of the respective step in section 2.4..

4a	For the distribution of seats to P3-lists for the EP and TK elections, all lists that obtained less votes than the electoral quota are excluded from the residual seat assignment. If a list is exhausted, all lists that do not have more candidates than seats are subsequently excluded from then the residual seat distribution.	**Note:** We will later see, that in most cases $LISTSTAKINGPART_{step} = LISTSTAKINGPART_{step\text{-}1}$. $LISTSTAKINGPART_{step} \neq LISTSTAKINGPART_{step\text{-}1}$ may only occur in two cases: • After the first assignment ($steptype_{step} = 1$), but only if $param_2$ = true, i.e. only for the distribution of seats to P3-lists for the EP and TK elections. • After an "exhausted list step" ($steptype_{step} = 7$). So we will see, that for $step > 1$, $LISTSTAKINGPART_{step}$ contains those lists, that were not found to be exhausted in an "exhausted list step" and that were not excluded from the residual seat assignment for not reaching the electoral quota.
5	If in the residual seat assignment drawing lot is required and more than one alternative must be drawn, each of them is drawn in a separate step. In this case after drawing a lot we need to remember the alternatives that remain for the consecutive step.	D_{step} is the set of **lists from which in a "continued drawing lots step"** one alternative must be drawn. We define $D_0 := \varnothing$ and $D_{step} := \varnothing$ for all unless stated otherwise in the description of the respective step in section 2.4..
5a		**Note:** We will later see, that $D_{step} \neq \varnothing$ may only occur in two cases (given some additional conditions): • After an assignment of residual seats (by the method of largest average or largest remainder). • After a continued drawing lots step. We will also see, that $D_{step} \neq \varnothing$ implies $\#D_{step} > residualseats_{step} > 0$. So if $D_{step} \neq \varnothing$, there is always at least one residual seat left, that will be assigned by lot in the next step $step + 1$.

6	After each step where a residual seat is assigned, we need to remember and keep track of the **set of lists** that is referred to in the article P 9 of the electoral law (modification of the seat distribution, if a list attained the absolute majority of all votes). This set of lists contains the lists that obtained a **seat for the lowest average** or the **smallest remainder**.	For each $step \in \mathbf{N}$, J_{step} is the set of lists that obtained a seat for the lowest average or the smallest remainder up to step $step$. We define $J_{step} := \emptyset$ in all steps $step \in \mathbf{N}$ unless stated otherwise in the description of the respective step in section 2.4.
6a		**Note:** We will later see, that $J_{step} \neq \emptyset$ only occurs in two cases: • In an assignment of residual seats (by the method of largest average or largest remainder): Here we set $J_{step} := Z_{step}$. This means that in case the next step is an "absolute majority step", J_{step} contains the lists that may loose a seat to the list with the absolute majority. • Continued drawing lots: Here we set $J_{step} := J_{step-1} \cup Z_{step}$, i.e. the list drawn in this step is added to the lists that already are in the set J_{step-1} from the previous step. In case the next step is an "absolute majority step", again J_{step} contains the lists that may loose a seat to the list with the absolute majority. But in this case the lists in J_{step} may have received the seat in step $step$ or $step$-1 or possibly earlier.
7	For the distribution of seats by the method of largest average **restricted to one seat per list**, we need to memorize the **set of lists** which have already **received a seat by this method**.	For each $step \in \mathbf{N}_0$, M_{step} is the set of lists that received a seat by the method of largest average restricted to one seat per list up to step i. We define $M_0 = \emptyset$ and $M_{step} := M_{step-1}$ in all steps $step \in \mathbf{N}$ unless stated otherwise in the description of the respective step in section 2.4.
8	In the seat assignment in list groups, we have to keep track if the cosalg-mode is switched on.	For each $step \in \mathbf{N}$, $cosalg_{step} \in \mathbf{B}$ indicates if the cosalg-mode is switched on. We define $cosalg_0$:= false and $cosalg_{step} := cosalg_{step-1}$ in all steps $step \in \mathbf{N}$ unless stated otherwise in the description of the respective step in section 2.4.

Implementation notes:

The frame of this algorithm is the method `GeneralSeatDistributor.mainLoop()` .The values *step*, $steptype_{step}$, $LISTSTAKINGPART_{step}$, $newseats_{step}$, $residualseats_{step}$, D_{step}, J_{step}, M_{step}, $cosalg_{step}$ are represented in the implementation as follows:

- *step*: `GeneralSeatDistributor._index`
- $steptype_{step}$: `GeneralSeatDistributor._stepType`
- $LISTSTAKINGPART_{step}$: `GeneralSeatDistributor._listsTakingPart`
- $newseats_{step}$ and $seatstolist_{step}$: are represented within `GeneralSeatDistributor._assignmentTracer`
- $residualseats_{step}$: is calculated in `GeneralSeatDistributor.getUnassignedSeats()`
- D_{step}: `GeneralSeatDistributor._drawingLotsLoosers`
- J_{step}: `GeneralSeatDistributor._assignmentForSmallestAverageOrRemainder`
- M_{step}: `GeneralSeatDistributor._receivedLargestAverageRestrictedSeat()`
- $cosalg_{step}$: `_cosalgModeOn`

2.3.6 CONDITIONS FOR THE NEXT ASSIGNMENT STEP

The following conditions in this section are designed in such a way that for each value of *step*, one and only one of these conditions applies. So if you check, which of them applies, formally it does not matter, in which order you do your checks.

Still the order is chosen carefully, because after checking one condition, some work for checking the next one is already done.

stap	*bewerking*	*Formalization*
1	The **assignment of seats based on attaining the quota** that was calculated in 2.3.1 step 2 is always the first step.	If $step = 1$ let $steptype_{step} := 1$ and the next assignment (i.e. the first assignment) $z_1 : P \rightarrow \mathbf{Z}$ is defined in section 2.4.1.
2	If **all lists are exhausted**, no more seats can be assigned.	If $step > 1$ and $P_{step\text{-}1} = \varnothing$ let $steptype_{step} := 2$. $newseats_{step}$ is defined in section 2.4.2. In this case there is no list left to take part in any other step. The assignment of seats may be terminated. This also terminates the cosalg-mode if $cosalg_{step\text{-}1}$ = true.
2a		Note: This can occur in two cases, which are similar: • After the first assignment (i.e. $steptype_{step\text{-}1} = 1$), but only if $param_2$ = true, i.e. only for the distribution of seats to P3-lists for the EP and TK elections. In this case, no party would receive a seat because there are so many of them and each of them receives less votes than the electoral quota. This case is possible in theory but irrelevant in praxis. • After a modification of the seat distribution, if a list is exhausted (i.e. $steptype_{step\text{-}1} = 7$). In this case, all lists are exhausted that were not excluded from the residual seat assignment for not reaching the electoral quota.

stap	*bewerking*	*Formalization*
2b	If in the previous step **drawing lots** of multiple lists has not been finished, this has to be **continued** immediately. Note that this kind of step was introduced to reflect the order in which lists are drawn by lot. This is not explicitly mentioned in the electoral law.	If $step > 1$ and $LISTSTAKINGPART_{step\text{-}1} \neq \varnothing$ and $residualseats_{step\text{-}1} > 0$ and $D_{step\text{-}1} \neq \varnothing$ the next assignment $newseats_{step} : P \rightarrow \mathbf{Z}$ is a continued drawing lots and is defined in section 2.4.8. If $steptype_{step\text{-}1} = 3$ or $steptype_{step\text{-}1} = 8$, we define $steptype_{step} := 8$. If $steptype_{step\text{-}1} = 4$ or $steptype_{step\text{-}1} = 9$, we define $steptype_{step} := 9$. Otherwise $steptype_{step\text{-}1} = 5$ or $steptype_{step\text{-}1} = 10$ and we define $steptype_{step} := 10$.
3		For all steps where residual seats are assigned, a common condition must be met. Define the boolean function $condition_0 : \mathbf{N} \rightarrow \mathbf{B}$ by $condition_0(step)$:= true if $step > 1$ and $LISTSTAKINGPART_{step\text{-}1} \neq \varnothing$ and $residualseats_{step\text{-}1} > 0$ and $D_{step\text{-}1} = \varnothing$ and $condition_0(step)$:= false otherwise.
	Implementation notes:	$condition_0(step)$ is stored in the local variable `c0_residualSeatCondition`.

stap	*bewerking*	*Formalization*
3a	For some seat distribution algorithms the distribution of residual seats is started using the **method of largest remainder**. These algorithms are given by the flag $param_4$: • distribution to P3-lists for AB1, GR1, ER, BC and GC elections, • distribution to P2-list for all elections. The assignment of residual seats by the method of largest remainder is repeated as long as there are still residual seats left to assign and as long as there are still lists that may receive a residual seat by largest remainder. For some seat distribution algorithms, this is restricted to lists that have reached at least 75% or 25% of the electoral quota. These algorithms are given by the parameter $param_3$: • for distribution to P3-lists for AB1, GR1, ER and GC elections the parameter is 75%, • for distribution to P3-lists for BC elections the parameter is 25%. For all other elections the parameter is set to 0%. *Implementation notes:*	Define the boolean function $condition_3 : \mathbf{N} \rightarrow \mathbf{B}$ by $condition_3(step)$:= true if $condition_0(step)$ = true and $param_4$ = true and \exists p \in LISTSTAKINGPART_{step-1} : {seatstolist}_{step - 1} (p) = {newseats}_1 (p) \land {votes} (p) \geq {param}_3 \cdot quota ($\exists p \in LISTSTAKINGPART_{step-1}: seatstolist_{step-1}(p) = newseats_1(p) \wedge votes(p) \geq param_3 \cdot quota$). and $condition_3(step)$:= false otherwise. If $condition_3(step)$ = true, the next assignment $newseats_{step}$: $LISTSTAKINGPART \rightarrow \mathbf{Z}$ is an assignment by the method of largest remainder (with or without drawing lots) and is defined in section 2.4.3. In this case we define $steptype_{step}$:= 3. $condition_3(step)$ is calculated in `GeneralSeatDistributor.c3ConditionForLargestRemainder()`.

stap	*bewerking*	*Formalization*
4	For some seat distribution algorithms the distribution of residual seats is continued by using the **method of largest average restricted to one seat** per list. These are given by the flag *param*$_5$: • distribution to P3-lists for AB1, GR1, ER, BC and GC elections. It is terminated if all residual seats are assigned or if all lists taking part have received one residual seat by this method.	Define the boolean function $condition_4 : \mathbf{N} \rightarrow \mathbf{B}$ by $condition_4(step)$:= true if $condition_0(step)$ = true and $condition_3(step)$:= false and *param*$_5$ = true and \exists p\in {LISTSTAKINGPART}_{step - 1} : p \notin M_{step - 1} $\exists p \in \text{LISTSTAKINGPART}_{step-1} : p \notin M_{step-1}$ and $condition_4(step)$:= false otherwise. If $condition_4(step)$ = true the next assignment $newseats_{step} : P \rightarrow \mathbf{Z}$ is an assignment by the method of largest average restricted to one seat per list (with or without drawing lots) and is defined in section 2.4.4. In this case we define *steptype*$_{step}$:= 4.
	Implementation notes:	$condition_4(step)$ is calculated in `GeneralSeatDistributor.c4ConditionForLargestAverageRestricted()`.
5	For all seat distribution algorithms the residual seat distribution includes the **method of largest average** (without restriction to one seat per list). If *param*$_4$ = false and *param*$_5$ = false, this is the first and only method used to assign residual seats, otherwise other methods are used before (see steps 3 and 4). The assignment of residual seats by the method of largest average is repeated as long as there are still residual seats left to assign and not all lists are exhausted.	If $condition_0(step)$ = true and $condition_3(step)$ = false and $condition_4(step)$ = false the next assignment $newseats_{step} : P \rightarrow \mathbf{Z}$ is an assignment by the method of largest average (not restricted to one seat per list, with or without drawing lots) and is defined in section 2.4.5. In this case we define *steptype*$_{step}$:= 5.

stap	*bewerking*	*Formalization*
6		For checking if a modification of the seat distribution, because a list attained the absolute majority of all votes, is required, the following condition has to be checked: $\exists\, p \in LISTSTAKINGPART_{step\text{-}1} : seatstolist_{step\text{-}1}(p) \leq seats / 2 \wedge votes(p) > totalvotes / 2$ This condition is called "absolute majority condition" in the following.
6a	When all residual seats have been assigned, the **absolute majority** regulation has to be considered. This step may only be performed if $param_6$ = true. This is the case only for all seat distribution algorithms to P3-lists except for the EK election (for any kind of election) or for fictitious seat distributions except for the EK election.	Define the boolean function $condition_6 : \mathbf{N} \rightarrow \mathbf{B}$ by $condition_6(step)$:= true if $step > 1$ and $LISTSTAKINGPART_{step\text{-}1} \neq \emptyset$ and $residualseats_{step\text{-}1} \leq 0$ and $param_6$ = true and the "absolute majority condition" holds and $condition_6(step)$:= false otherwise. If $condition_6(step)$ = true, the next assignment $newseats_{step} : P \rightarrow \mathbf{Z}$ is a modification of the seat distribution, because a list attained the absolute majority of all votes and is defined in 2.4.6. In this case we define $steptype_{step}$:= 6. **Note:** We will prove in section 2.4.6 that this may be the case for at most one value of *step*.
	Implementation notes:	$condition_6(step)$ is calculated in `GeneralSeatDistributor.c6ConditionForAbsoluteMajority()`.

stap	*bewerking*	*Formalization*
7	After all the previous steps, if all residual seats are assigned, **list exhaustion** may cause that seats that have been assigned to a list may have to be re-assigned to one or more other lists by continuing the described procedure. Note that in the step where list exhaustion is detected, seats are taken away from the exhausted lists. The re-assignment of these seats to other lists occurs in the following separate step(s).	Define the boolean function $condition_7 : \mathbf{N} \rightarrow \mathbf{B}$ by $condition_7(step)$:= true if $step > 1$ and $LISTSTAKINGPART_{step\text{-}1} \neq \varnothing$ and $residualseats_{step\text{-}1} \leq 0$ and $c_6(step)$ = false and $\exists p \in P: \text{seatstolist}_{step-1}(p) > noofcandidatsinlist(p)$ and $condition_7(step)$:= false otherwise. If $condition_7(step)$ = true, the next assignment $newseats_{step} : P \rightarrow \mathbf{Z}$ is a modification of the seat distribution, if a list is exhausted and is defined in 2.4.7. In this case we define $steptype_{step}$:= 7.
	Implementation notes:	To calculate $condition_7(step)$ the helper method `GeneralSeatDistributor. isAnyListExhausted()` is used.
(8-10)	(Step numbers 8-10 are not used)	
11	If all **residual seats are assigned**, the absolute majority regulation has been considered and no list is exhausted, the **seats assignment is terminated**.	If $step > 1$ and $LISTSTAKINGPART_{step\text{-}1} \neq \varnothing$ and $residualseats_{step\text{-}1} \leq 0$ and $condition_6(step)$ = false and $condition_7(step)$ = false and ($param_7$ = false or $cosalg_{step\text{-}1}$ = true) let $steptype_{step}$:= 11. $newseats_{step}$ is defined in section 2.4.11.. In this case there are no residual seats left to distribute. The assignment of seats may be terminated.

stap	*bewerking*	*Formalization*
12	Only if in the previous case, a continued seat assignment in list groups is required, a special kind of step is preformed in which the **cosalg-mode** is **switched on**.	If $step > 1$ and $LISTSTAKINGPART_{step\text{-}1} \neq \varnothing$ and $residualseats_{step\text{-}1} \leq 0$ and $condition_6(step)$ = false and $condition_7(step)$ = false and $param_7$ = true and $cosalg_{step\text{-}1}$ = false let $steptype_{step} := 12$. $newseats_{step}$ is defined in section 2.4.12.. The regular assignment of seats is terminated, but the seat assignment is continued in the cosalg-mode to determine the roll back sequence and roll forward sequence.
	Implementation notes:	In `GeneralSeatDistributor . determineNextStepType()` the next step type is determined.

2.3.7 PROOF THAT THE ALGORITHM TERMINATES

This prove only covers the case that b_7 = false. Then the cosalg-mode is never switched on and $steptype_{step} = 12$ never occurs.

stap	*bewerking*	*Formalization*
1	After a sufficiently large number of steps, one of the termination conditions will occur.	For sufficiently large numbers of $step$, further iterations do not change $seatstolist_{step}$ any more, because $steptype_{step} = 2$ or $steptype_{step} = 11$ and thus $newseats_j \equiv 0$ for all $j \geq step$. For any such value of $step$ we define the total distribution of seats to lists by: $seatstolist: P \rightarrow \mathbf{N}_0$, $seatstolist(p) := seatstolist_{step}(p)$
2		We will now sketch a proof that the algorithm does in fact terminate as described. We will give the main arguments of this proof but leave out some of the details.

stap	*bewerking*	*Formalization*
3		The first argument is, that the set $K_1 := \{ step \in \mathbf{N} \mid steptype_{step} = 1 \vee steptype_{step} = 6 \vee steptype_{step} = 7 \}.$ is finite. This is obvious for $steptype_{step} = 1$. For $steptype_{step} = 6$ (absolute majority) this is proved in section 2.4.6. For $steptype_{step} = 7$ (exhausted lists) this follows from the fact that $\#LISTSTAKINGPART_{step} \leq \#LISTSTAKINGPART_{step\text{-}1}$ is true for all values $step \in \mathbf{N}$ and that $steptype_{step} = 7$ implies $\#LISTSTAKINGPART_{step} < \#LISTSTAKINGPART_{step\text{-}1}$.
4		For all other values of *step*, i.e. $step \in \mathbf{N} \setminus K_1$ we have $residualseats_{step} \leq residualseats_{step\text{-}1}$. Hence also the set $K_2 := \{ step \in \mathbf{N} \mid steptype_{step} = 3 \vee steptype_{step} = 4 \vee steptype_{step} = 5 \vee steptype_{step} = 8 \vee steptype_{step} = 9 \vee steptype_{step} = 10 \}$ (residual seat assignments) is finite, because $step \in K_2$ implies both $residualseats_{step} < residualseats_{step\text{-}1}$ and $0 < residualseats_{step\text{-}1}$.
5		So the set $K_3 := \{ step \in \mathbf{N} \mid steptype_{step} = 2 \vee steptype_{step} = 11 \} = \mathbf{N} \setminus K_1 \setminus K_2$ is infinite, especially non-empty.

2.4 DETAILS OF THE DIFFERENT KINDS OF STEPS

2.4.1 ASSIGNMENT OF SEATS BASED ON ATTAINING THE QUOTA (FIRST ASSIGNMENT)

Regulations by law:

- Article P 6 and U8 of the electoral law for distribution to P3-lists.
- Article P 12 subsection 4 and U 12 subsection 4 of the electoral law for distribution to P2-lists.

stap	*bewerking*	*Formalization*
1	Van elke lijst wordt het stemtotaal gedeeld door de kiesdeler.	
2	Aan elke lijst wordt een aantal zetels toegedeeld, gelijk aan het aantal malen dat de kiesdeler in het stemtotaal is begrepen.	The first assignment of seats is defined by a function: $newseats_1 : P \rightarrow \mathbf{Z} \quad , \quad newseats_1(p) := \left\lfloor \frac{votes(p)}{quota} \right\rfloor$ This means that $newseatsz_1(p)$ is the number of seats that are assigned to p in the first assignment.
3	Vastgesteld wordt hoeveel zetels in totaal zijn toegedeeld.	Hence $\sum_{p \in P} newseats_1(p) \leq seats$. Recall that $residualseats_1$ = $seats$ - $\sum_{p \in P} newseats_1(p)$. So $residualseats_1$ is the number of residual seats.
4	Only for distributions to P42-lists in the EP or TK elections, i.e. if $param_2$ = true: De lijsten waaraan bij de eerste toedeling van zetels een of meer zetels zijn toegekend, worden geselecteerd. - Lijsten waaraan bij de eerste toedeling geen zetels zijn toegewezen, komen dus niet voor een restzetel in aanmerking	If $param_2$ = false, let $LISTSTAKINGPART_1 := \{ p \in P \mid votes(p) \geq 1 \}$ be the set of all lists that received at least one vote. Only those take part in the following iterations. If $param_2$ = true, let $LISTSTAKINGPART_1 := \{ p \in P \mid newseats_1(p) \geq 1 \}$ be the set of all lists that have been awarded a seat in the first assignment. Only those take part in the following iterations.
5		D_1, J_1 and M_1 are defined as usual: $D_1 := \emptyset$, $J_1 := \emptyset$, $M_1 := M_0 = \emptyset$, $cosalg_{step} := cosalg_{step\text{-}1}$.
	Implementation notes:	The first assignment is implemented in `GeneralSeatDistributor.performFirstAssignment()`.

2.4.2 ALL LISTS EXHAUSTED

Regulation by law: none.

stap	*bewerking*	*Formalization*
1	The case that all lists are exhausted is not explicitly mentioned in the electoral law. Still it may theoretically occur.	This kind of step is executed in case $steptype_{step} = 2$.
2	In this case no further seats are assigned.	We define $LISTSTAKINGPART_{step}$:= $LISTSTAKINGPART_{step\text{-}1}$ and $newseats_{step} \equiv 0$.
3	The algorithm may be terminated, because all the following steps will be of the same kind and will not change the seat distribution any more.	The definition of $LISTSTAKINGPART_{step}$ implies $steptype_{step+1} = 2$. So, all following step will be of the same kind. The definition of $newseats_{step}$ implies that the seat distribution does not change any more.
4		D_{step}, J_{step} and M_{step} are defined as usual: $D_{step} := \emptyset$, $J_{step} := \emptyset$, $M_{step} := M_{step\text{-}1}$, $cosalg_{step} := cosalg_{step\text{-}1}$.
	Implementation notes:	This step type terminates the algorithm, see `GeneralSeatDistributor.isTerminationStep()`.

2.4.3 ASSIGNMENT OF RESIDUAL SEATS BY LARGEST REMAINDER

Regulation by law: article P 8 electoral law.

The method used to assign residual seats is called **method of the largest remainder (according to Hare-Niemeyer)**.

stap	*bewerking*	*Formalization*
1		We describe this step kind in a way that allows us to re-use the description also for other kinds of steps that assign residual seats (sections 2.4.4 and 2.4.5). For this reason we introduce the set $H_{step\text{-}1} \subset P$. It is the set of lists that is considered to receive a seat in the current step. Also we introduce a function $f_{step} : H_{step\text{-}1} \rightarrow \mathbf{Q}$ that in this case is used to calculate the remainder. In other cases it will be defined differently such that it is used to calculate the next average of a list.

stap	*bewerking*	*Formalization*
2	Vastgesteld wordt hoeveel zetels nog toegedeeld moeten worden (restzetels).	Recall that $$residualseats_{\text{step}-1} = seats - \sum_{p \in P} \text{seatstolist}_{step-1}(p) > 0$$ is the number of residual seats that have not been assigned to any list, yet.
3a	If $param_3$ = 0, proceed like this: - Deze methode van toedeling van restzetels impliceert dat maar één restzetel per lijst kan worden toegekend. Van alle lijsten worden als volgt de overschotten aan stemmen berekend:	If $param_3 = 0$, set $H_{\text{step}-1}$: $= \{p \in LISTSPATINGPART_{step-1} \mid seatstolist_{step-1}(p) = newseats_1(p)\}$ If this step is performed, $steptype_{step} = 3$ and thus $H_{i\text{-}1} \neq \emptyset$. ($H_{step\text{-}1}$ will be defined differently in sections 2.4.4 and 2.4.5)
3b	If $param_3 > 0$, proceed like this: - Deze methode van toedeling van restzetels impliceert dat maar één restzetel per lijst kan worden toegekend. Van de lijsten met een stemtotaal dat ten minste 75% van de kiesdeler bedraagt, worden als volgt de overschotten aan stemmen berekend:	If $param_3 > 0$, set $H_{step\text{-}1} := \{p \in LISTSTAKINGPART_{step\text{-}1} \mid seatstolist_{step\text{-}1}(p) = newseats_1(p) \wedge votes(p) \geq param_3 \cdot quota\}$ If this step is performed, $steptype_{step} = 3$ and thus $H_{step\text{-}1} \neq \emptyset$.
4	- Voor lijsten waaraan bij de eerste toedeling een of meer zetels zijn toegekend, is het overschot de rest die overblijft bij deling van het stemtotaal door de (combinatie-, groeps-) kiesdeler. - Voor lijsten waaraan bij de eerste toedeling nog geen zetel is toegekend, geldt het stemtotaal als overschot.	We define the function $f_{step} : H_{step\text{-}1} \rightarrow \mathbf{Q}$ such that it calculates the remainder: $f_{\text{step}}(p) := votes(p) - newseats_1(p) \cdot quota$ (f_{step} will be defined differently in sections 2.4.4 and 2.4.5)

stap	*bewerking*	*Formalization*
5		**Note:** $votes(p) - \text{newseats}_1(p) \cdot quota$ may be written as fraction of integers: $$f_{\text{step}}(p) = \frac{votes(p) \cdot seats - \text{newseats}_1(p) \cdot totalvotes}{seats}$$
6		The largest remainder is $$d_{\text{step}} := \max\{f_{\text{step}}(p) \mid p \in H_{\text{step}-1}\}$$
7		The lists, that reach this maximum are $$Z_{max,step} := \{\, p \in H_{step\text{-}1} \mid f_{step}(p) = d_{step} \,\}$$
8	De eerste nog niet toegekende zetel wordt toegekend aan de lijst met het grootste overschot, [de tweede aan de lijst met het op een na grootste overschot en zo vervolgens]. **Remark:** The assignment of seats to the list with the second largest average, third largest average etc. will be performed in later steps if sufficient seats are available.	Now, we define Z_{step} being the set of lists, to which another seat is assigned in this iteration: If $residualseats_{step\text{-}1} \geq \#Z_{max,step}$ or $cosalg_{step}$ = true, then define $Z_{step} := Z_{max,step}$.
9a	Zijn overschotten gelijk, dan is, als er minder zetels ter verdeling over zijn dan het aantal gelijke overschotten, toekenning van de zetel bij loting in de zitting van het centraal stembureau noodzakelijk.	Otherwise $Z_{step} \subset Z_{max,step}$ must be defined by lot, such that $\#Z_{step} = 1$. In both cases we define $$\text{newseats}_{\text{step}}(p) := \begin{cases} 1 if p \in Z_{\text{step}} \\ 0 if p \notin Z_{\text{step}} \end{cases}$$
9b		**Note:** In case drawing lots is required, the residual seats are assigned one by one, i.e. in each iteration only one seat is drawn. This is to reflect the order in which the seats are assigned by lot.

stap	*bewerking*	*Formalization*
9c		To make sure that in the next step a continued drawing lots is performed if necessary, we define
	This applies similarly to averages instead of remainder in the cases of other step kinds (sections 2.4.4 and 2.4.5).	$D_{step} := Z_{max,step} \setminus Z_{step}$
		if $1 < residualseats_{step\text{-}1} < \#Z_{max,step}$. In all other cases, i.e. if $residualseats_{step\text{-}1} \geq \#Z_{max,\,step}$ or $residualseats_{step\text{-}1} = 1$, we define
		$D_{step} := \emptyset$.
		Define $J_{step} := Z_{step.}$
10		**Note:** We have either $D_{step} = \emptyset$ or $\#D_{step} > residualseats_{step} > 0$.
11		$LISTSTAKINGPART_{step}$ and M_{step} are defined as usual: $LISTSTAKINGPART_{step} := LISTSTAKINGPART_{step\text{-}1}$, $M_{step} := M_{step\text{-}1,}$ $cosalg_{step} := cosalg_{step\text{-}1}$.
	Implementation notes:	This step type is implemented in the method `GeneralSeatDistributor`.
		`performNiemeyerAssignment()`. H_{step} is determined in the method `getListWithoutResidualSeat()`. The values $votes(p) - newseats_1(p) \cdot quota$ are represented as `FractionFromList`s.
		The reusable part of this step type is implemented in the method `assignResidualSeats()`. This method receives as parameter `lists` the set $H_{step\text{-}1}$ of lists that are considered in that step. The second parameter `fractionFactory` corresponds to the function f_{step}.
		d_{step} is represented by the variable `max` in `assignResidualSeats()`. $Z_{max,step}$ is represented by the variable `listsWithLargestFraction`.

2.4.4 ASSIGNMENT OF RESIDUAL SEATS BY LARGEST AVERAGE RESTRICTED TO ONE SEAT

Regulation by law: article P 8 subsection 3 and article P 7 subsection 1 electoral law.

The method used to assign residual seats is called **method of the largest average (according to d'Hondt)** provided that no more than one seat may be assigned in this way to any of the lists.

stap	*bewerking*	*Formalization*
1		Recall from section 2.3.5 step 7: M_{step} is the set of lists that received a seat by the method of largest average restricted to one seat per list up to step *step*.
2	The assignment of residual seats by largest average restricted to one seat is similar to the assignment of residual seats by largest remainder. Therefore in this description we refer to a large degree to the description of the method of largest remainder in section 2.4.3. De overblijvende zetels, die restzetels worden genoemd, worden, indien het aantal te verdelen zetels negentien of meer bedraagt, achtereenvolgens toegewezen aan de lijsten die na toewijzing van de zetel het grootste gemiddelde aantal stemmen per toegewezen zetel hebben. Indien gemiddelden gelijk zijn, beslist zo nodig het lot.	To describe this step kind we re-use the description of section 2.4.3. Deviating from section 2.4.3 we define the set $H_{step\text{-}1} \subset P$ by $H_{step\text{-}1} := LISTSTAKINGPART_{step\text{-}1} \setminus M_{step\text{-}1}$, i.e. all lists that take part in this step and did not receive a residual seat by largest average restricted in an earlier step are considered for a seat. Also deviating from section 2.4.3 we define the function $f_{step} : H_{step\text{-}1} \to \mathbf{Q}$ by $$f_{\text{step}}(p) := \frac{votes(p)}{seatstilists_{\text{step}-1}(p) + 1}$$
3		d_{step}, $Z_{max,step}$, Z_{step}, D_{step} and J_{step} are defined exactly as in section 2.4.3.
4	... worden deze zetels toegewezen volgens het stelsel van de grootste gemiddelden als bedoeld in artikel P 7, eerste lid, met dien verstande, dat bij deze toewijzing aan geen van de lijsten meer dan één zetel wordt toegewezen.	Deviating from section 2.4.3 we define $M_{step} := M_{step\text{-}1} \cup Z_{step}$.
5		$LISTSTAKINGPART_{step}$ is defined as usual: $LISTSTAKINGPART_{step} := LISTSTAKINGPART_{step\text{-}1}$, $cosalg_{step} := cosalg_{step\text{-}1}$.

stap	*bewerking*	*Formalization*
	Implementation notes:	This step type is implemented in the method `GeneralSeatDistributor.performDHondtAssignmentRestricted()`. The fraction $\frac{votes(p)}{seatstolist_{step-1}(p)+1}$ is represented by a `FractionFromList`. These fractions are created by the anonymous class in the attribute `_dHondtFractionFactory`.

2.4.5 ASSIGNMENT OF RESIDUAL SEATS BY LARGEST AVERAGE

Regulation by law: article P 7 and U 9 electoral law.

The method used to assign residual seats is called **method of the largest average (according to d'Hondt)**.

stap	*bewerking*	*Formalization*
1	The assignment of residual seats by largest average is similar to the assignment of residual seats by largest remainder. Therefore in this description we refer to a large degree to the description of the method of greatest remainder in section 2.4.3. De overblijvende zetels, die restzetels worden genoemd, worden, indien het aantal te verdelen zetels negentien of meer bedraagt, achtereenvolgens toegewezen aan de lijsten die na toewijzing van de zetel het grootste gemiddelde aantal stemmen per toegewezen zetel hebben. Indien gemiddelden gelijk zijn, beslist zo nodig het lot.	To describe this step kind we re-use the description of section 2.4.3. Deviating from section 2.4.3 we define the set $H_{step\text{-}1} \subset P$ by $H_{step\text{-}1} := LISTSTAKINGPART_{step\text{-}1}$, i.e. all lists that take part in this step are also considered for a seat. Also deviating from section 2.4.3 we define the function $f_{step} : H_{step\text{-}1} \to \mathbf{Q}$ by $$f_{\text{step}}(p) := \frac{votes(p)}{\text{seatstolist}_{\text{step}-1}(p) + 1}$$
2		d_{step}, $Z_{max,step}$, Z_{step}, D_{step} and J_{step} are defined exactly as in section 2.4.3.
3		P_{step} and M_{step} are defined as usual: $LISTSTAKINGPART_{step} := LISTSTAKINGPART_{step\text{-}1}$, $M_{step} := M_{step\text{-}1}$, $cosalg_{step} := cosalg_{step\text{-}1}$.

stap	*bewerking*	*Formalization*
	Implementation notes:	The assignment by largest average is implemented in the method `GeneralSeatDistributor.performDHondtAssignment()`. `_assignmentTracer.getTotalAssignment(P)` provides the value $seatstolist_{step-1}(p)$. The fraction $\frac{votes(p)}{seatstolist_{step-1}(p)+1}$ is represented by a `FractionFromList`. These fractions are created by the anonymous class in the attribute `_dHondtFractionFactory` implementing the interface `FractionFactory`.

2.4.6 MODIFICATION OF THE SEAT DISTRIBUTION, IF A LIST ATTAINED THE ABSOLUTE MAJORITY OF ALL VOTES

Regulations by law: article P 9 electoral law.

stap	*bewerking*	*Formalization*
		This step is performed in case $steptype_{step} = 6$.
1	De volstrekte meerderheid van het aantal uitgebrachte stemmen wordt berekend.	The absolute majority of votes is $AMV := \lfloor totalvotes/2 \rfloor + 1$

stap	*bewerking*
2	Is aan een lijst die dit aantal heeft bereikt, niet de volstrekte meerderheid van het totale aantal te verdelen zetels toegewezen, dan wordt aan die lijst één zetel extra toegekend.

Formalization

The absolute majority of seats is

$$AMS := \lfloor seats/2 \rfloor + 1 > seats/2$$

Let

$$P_{AM} := \{\ p \in LISTSTAKINGPART_{step\text{-}1} \mid seatstolist_{step\text{-}1}(p) < AMS \wedge votes(p) \geq AMV\}$$

$$= \{\ p \in LISTSTAKINGPART_{step\text{-}1} \mid seatstolist_{step\text{-}1}(p) \leq seats/2 \wedge votes(p) > totalvotes/2\}$$

be the set of all lists, that attained the absolute majority of all votes, but not the absolute majority of all seats. Obviously $\#P_{AM} \in \{0, 1\}$, i.e. there is at most one list with this property, because if one list has more than half of the votes the remaining lists altogether have less than half of the votes, so each of them has less than half of the votes.

Recall that this step is only performed if the "absolute majority condition" is fulfilled (see section 2.3.6, step 6):

$$\exists\ p \in LISTSTAKINGPART_{i\text{-}1} : seatstolist_{step\text{-}1}(p) \leq seats/2 \wedge votes(p) > totalvotes/2$$

i.e. if $\#P_{AM} = 1$. Let $p_{AM} \in P_{AM}$ be the list with the absolute majority of all votes, but not the absolute majority of all seats.

First remark: We will later define $newseats_{step}$ in terms of $J_{step\text{-}1}$ which is non-empty if and only if the previous step was an assignment of residual seats ($steptype_{step\text{-}1} \in \{3, 4, 5\}$) or a continued drawing lots ($steptype_{step\text{-}1} \in \{8, 9, 10\}$).

To prove this we logically exclude all other possible kinds of steps:

stap *bewerking*

Formalization

The previous step cannot be the first assignment (*steptype*$_{step\text{-}1}$ = 1). We prove this by contradiction:

steptype$_{step\text{-}1}$ = 1 would imply $step = 2$ and

$$0 \geq residualseats_1 = seats - \sum_{p \in P} newseats_1(p) = seats - \sum_{p \in P} \left\lfloor \frac{votes(p)}{quota} \right\rfloor \geq$$

$$seats - \sum_{p \in P} \frac{votes(p)}{quota} = seats - \sum_{p \in P} \frac{seats \cdot votes(p)}{totalvotes} = 0,$$

thus the inequality really is an equality:

$$newseats_1(p) = \left\lfloor \frac{votes(p)}{quota} \right\rfloor = \frac{votes(p)}{quota} \text{ for all } p \in P.$$

Dividing both sides by *seats* gives *newseats*$_1$*(p)/seats* = *votes*(*p*)*/totalvotes*. Now for $p = p_{AM}$ we get

$$\frac{1}{2} < \frac{AMV}{totalvotes} \leq \frac{votes(p_{AM})}{totalvotes} = \frac{newseats_1(p_{AM})}{seats} = \frac{seatstolist_1(p_{AM})}{seats}.$$

But since p_{AM} does not have the absolute majority of seats,

$$\frac{\text{seatstolist}_1(p_{AM})}{seats} \leq \frac{1}{2}$$

which is a contradiction.

The previous step cannot be a modification of the seat distribution, if a list attained the absolute majority (*steptype*$_{step\text{-}1}$ = 6), because this kind of step can only occur once.

stap	*bewerking*

Formalization

The previous step cannot be a modification of the seat distribution, if a list is exhausted ($steptype_{step\text{-}1} = 7$).

Because in that case the "absolute majority condition" would also hold for *step* - 1. Also $steptype_{step\text{-}1} = 7$ would imply $step - 1 > 1$ and $LISTSTAKINGPART_{step\text{-}2} \neq \varnothing$ and $residualseats_{step\text{-}2} \leq 0$. Also $steptype_{step} = 6$ implies $b_6 = \text{true}$. Thus $c_6(step - 1) = \text{true}$ which contradicts $steptype_{step\text{-}1} = 7$.

The previous step cannot be an "all lists exhausted" step ($steptype_{step\text{-}1} = 2$), because that would imply $steptype_{step} = 2$.

The previous step cannot be an "all seats were assigned" step ($steptype_{step\text{-}1} = 11$), because that would imply $steptype_{step} = 11$.

Second remark: It is always true that $p_{AM} \notin J_{step\text{-}1}$. This is essential to keep the below definition of $newseats_{step}$ reasonable. We prove this by contradiction:

Suppose $p_{AM} \in J_{step\text{-}1}$. This implies that p_{AM} has received a residual seat in one of the previous steps j with $1 < j < step$. Thus

$$\begin{aligned} seatstolist_{step-1}(p_{AM}) &\geq newseats_1(p_{AM}) + newseats_j(p_{AM}) \\ &= \left\lfloor \frac{votes(p_{AM})}{quota} \right\rfloor + 1 > \frac{votes(p_{AM})}{quota} \\ &= seats \cdot \frac{votes(p_{AM})}{totalvotes} > \frac{seats}{2} \end{aligned}$$

thus p_{AM} has the absolute majority of seats which is a contradiction.

stap	*bewerking*	*Formalization*
3	Het zetelaantal van de lijst waaraan de laatste restzetel is toegewezen, wordt met 1 verminderd.	So now we can define $newseats_{step}$ as follows: If $\#J_{step\text{-}1} = 1$, let $p_{LR} \in J_{step\text{-}1}$ be the single list, that received the last residual seat. If $\#J_{step\text{-}1} > 1$, then $p_{LR} \in J_{step\text{-}1}$ must be determined by lot. Then we define: $$newseats_{step}(p) := \begin{cases} 1 & if \quad p = p_{AM} \\ -1 & if \quad p = p_{LR} \\ 0 & otherwise \end{cases}$$
4	Waren er meer lijsten waaraan voor hetzelfde gemiddelde of overschot als van de in stap 3 bedoelde lijst een restzetel is toegewezen, dan wordt bij loting in de zitting van het centraal stembureau bepaald van welke lijst het zetelaantal met 1 wordt verminderd.	$LISTSTAKINGPART_{step}$, D_{step}, J_{step} and M_{step} are defined as usual: $LISTTAKINGPART_{step} := LISTTAKINGPART_{step\text{-}1}$, $D_{step} := \emptyset$, $J_{step} := \emptyset$, $M_{step} := M_{step\text{-}1}$, $cosalg_{step} := cosalg_{step\text{-}1}$.

stap	*bewerking*
5	
	Implementation notes:

Formalization

Third remark: We noted earlier that this step can only occur once. This can be proven as follows:

There are only two cases in all different kinds of steps where a P3-list looses a seat. The first case is list exhaustion. I.e. if a list $p \in P$ is exhausted in a step j with $steptype_{step} = 7$, then $seatstolist_j(p) < seatstolist_{j-1}(p)$. But this also implies $p \notin LISTSTAKINGPART_{j'}$ for all $j' \geq j$, i.e. k does not take part in any of the following steps. The second case is losing a seat in the modification of the seat distribution, if another list attained the absolute majority of all votes ($steptype_{step} = 6$).

Suppose $steptype_{step} = 6$ and p_{AM} is defined as above. Then p_{AM} in step $step$ receives one more seat. So after step $step$, p_{AM} has the absolute majority of seats:

$$seatstolist_{\text{step}}(p_{AM}) \geq newseats_1(p_{AM}) + newseats_{\text{step}}(p_{AM}) \geq \left\lfloor \frac{votes(p_{AM})}{quota} \right\rfloor + 1 \geq \left\lfloor \frac{seats}{2} \right\rfloor + 1 > \frac{seats}{2}$$

Suppose in another step again the "absolute majority condition" is fulfilled. Because at most one list can have the absolute majority of votes, it must be p_{AM} again that fulfils this condition. This implies that p_{AM} has lost at least one seat in one of the steps in between. So either p_{AM} was exhausted which is a contradiction, because then p_{AM} would not take part in further steps. Or p_{AM} lost a seat to another list with the absolute majority of votes, which is also impossible, because at most one list can have the absolute majority of votes.

This step type is implemented in the method `GeneralSeatDistributor. considerAbsoluteMajority()`. The list that has the absolute majority is determined in `getAbsoluteMajorityList()` using the integer condition

$$2 \cdot votes(p) > totalvotes \text{ and } 2 \cdot seatstolist_{step}(p) \leq seats.$$

2.4.7 MODIFYING THE DISTRIBUTION OF SEATS IN CASE OF LIST EXHAUSTION

Regulations by law: article P 10 and U 10 electoral law.

stap	*bewerking*	*Formalization*
1	Het aantal aan een lijst toegekende zetels wordt vergeleken met het totale aantal kandidaten van de lijst.	Let $LISTSTAKINGPART_{step} := \{ p \in LISTSTAKINGPART_{step\text{-}1} \mid noofcandidatsinlist(p) > seatstolist_{step\text{-}1}(p) \}$ be the lists that still have more candidates than seats. Only those lists will take part in the following steps.
2		In this step there will be a "negative assignment" to all lists that have so far received more seats than they have candidates. Define $newseats_{\text{step}}(p) := \begin{cases} noofcandidatsinlist(p) - \text{seatstolist}_{step-1}(p) & if \quad \text{seatstolist}_{step-1}(p) > noofcandidatsinlist(p) \\ 0 & otherwise \end{cases}$
3		**Note 1:** As a result of this step, for those lists $p \in LISTSTAKINGPART_{step\text{-}1} \setminus LISTSTAKINGPART_{step}$, that do not take part in any further steps we have: $\text{seatstolist}_{step}(p) = noofcandidatsinlist(p)$
4		**Note 2:** Only in the first assignment and in this kind of step we may have $LISTSTAKINGPART_i \neq LISTSTAKINGPART_{step\text{-}1}$. In all kinds of steps we have $LISTSTAKINGPART_{step} \subset LISTSTAKINGPART_{step\text{-}1}$. I.e. a list that does not take part in one step will not take part in any of the following steps.
5	De lijsten waarvan zetels niet bezet konden worden, blijven bij deze voortgezette toepassing uiteraard buiten beschouwing.	**Note 3:** In this step, only lists with $seatstolist_{step-1}(p) > noofcandidatsinlist(p)$ loose one or more seats. But all lists with $seatstolist_{step-1}(p) \geq noofcandidatsinlist(p)$ are excluded from the following steps.

6	De zetels die niet bezet kunnen worden, worden toegekend aan andere lijsten. Welke lijsten dat achtereenvolgens zijn, wordt bepaald door voortgezette toepassing van de regels voor het toewijzen van restzetels (zie onderdeel 4).	**Note 4:** The electoral law states that in this step seats shall pass to other lists. We deviate from the words of the electoral law here. In this step we only take away seats from the exhausted lists but do not assign them to other lists. This has two reasons: 1. It is in theory possible that all lists are exhausted. In this case we assume that it is the intention of the law that the exhausted lists loose seats even though they cannot be assigned to any other list. 2. The assignment of the lost seats to other lists is complex and follows the specification of the other kinds of steps. So it is not reasonable to describe this assignment as part of the "list exhaustion" step. D_{step}, J_{step} and M_{step} are defined as usual: $D_{step} := \emptyset$, $J_{step} := \emptyset$, $M_{step} := M_{step\text{-}1}$, $cosalg_{step} := cosalg_{step\text{-}1}$.
	Implementation notes:	This step type is implemented in the method `GeneralSeatDistributor.performExhaustedListsStep()`. The method that determines which lists are exhausted and how many seats of which exhausted list have to be re-assigned is determined in `AssignmentTracer.getExhaustedListsMap()`.

2.4.8 CONTINUED DRAWING LOTS IN ASSIGNMENT OF RESIDUAL SEATS

Regulations by law: articles P 7 subsection 1, P 8 subsection 1, P 12 subsection 5, U 9 subsection 1, U 12 subsection 5 electoral law.

stap	*bewerking*	*Formalization*
1		This step is only performed if $D_{step\text{-}1} \neq \emptyset$ and $residualseats_{step\text{-}1} > 0$. This implies that the previous step was an assignment of residual seats ($steptype_{step\text{-}1} \in \{3, 4, 5\}$) or a continued drawing lots step ($steptype_{step\text{-}1} \in \{8, 9, 10\}$). Hence $\#D_{step\text{-}1} > residualseats_{step\text{-}1} > 0$.
2		Determine $Z_{step} \subset D_{step\text{-}1}$ by lot, such that $\#Z_{step} = 1$. Define $$newseats_{\text{step}}(p) := \begin{cases} 1 if p \in Z_{\text{step}} \\ 0 if p \notin Z_{\text{step}} \end{cases}$$

3		If $residualseats_{step\text{-}1} > 1$, we have to make sure that in the next step a continued drawing lots is performed. So in this case we define $D_{step} := D_{step\text{-}1} \setminus Z_{step}$. Otherwise $residualseats_{step\text{-}1} = 1$, and we define $D_{step} := \varnothing$. Define $J_{step} := J_{step\text{-}1} \cup Z_{step}$.
4		If $steptype_{step} = 9$, define $M_{step} := M_{step\text{-}1} \cup Z_{step}$. Otherwise M_{step} is defined as usual: $M_{step} := M_{step\text{-}1}$.
5		$LISTSTAKINGPART_{step}$ is defined as usual: $LISTSTAKINGPART_{step} := LISTSTAKINGPART_{step\text{-}1}$, $cosalg_{step} := cosalg_{step\text{-}1}$.
6		**Note:** Again we have either $D_{step} = \varnothing$ or $\#D_{step} > residualseats_{step} > 0$.
	Implementation notes:	The continued drawing lots is implemented in the method `GeneralSeatDistributor.performContinuedDrawingLots()`.

2.4.9 (LEFT EMPTY)

In order to keep the numbers of $steptype_{step}$ in sync with the number of the subsections, this subsection is left empty. Please see the previous subsection for the case $steptype_{step} = 9$.

2.4.10 (LEFT EMPTY)

In order to keep the numbers of $steptype_{step}$ in sync with the number of the subsections, this subsection is left empty. Please see the previous subsection for the case $steptype_{step} = 10$.

2.4.11 ALL SEATS ASSIGNED

stap	*bewerking*	*Formalization*
1	If all residual seats are assigned, the absolute majority regulation has been considered and no list is exhausted, the seats assignment is terminated.	This kind of step is executed in case $steptype_{step} = 11$.
2	In this case no further seats are assigned.	We define $LISTSTAKINGPART_{step}$:= $LISTSTAKINGPART_{step-1}$ and $newseats_{step} \equiv 0$.
3	The algorithm may be terminated, because all the following steps will be of the same kind and will not change the seat distribution any more.	The definition of $LISTSTAKINGPART_{step}$ implies $steptype_{step+1} = 11$. So the following step will be of the same kind. The definition of z_i implies that the seat distribution does not change any more.
4		D_{step}, J_{step} and M_{step} are defined as usual: $D_{step} := \varnothing$, $J_{step} := \varnothing$, $M_{step} := M_{step-1}$, $cosalg_{step} := cosalg_{step-1}$.
	Implementation notes:	This step type terminates the algorithm, see `GeneralSeatDistributor.isTerminationStep()`.

2.4.12 SWITCH COSALG-MODE ON

stap	*bewerking*	*Formalization*
1	If all residual seats are assigned, the absolute majority regulation has been considered and no list is exhausted, the seats assignment is terminated. If this seat assignment shall be continued to determine the roll back sequence and roll forward sequence, but the cosalg-mode is not yet on, switch it on.	This kind of step is executed in case $steptype_{step} = 12$. Define $cosalg_{step}$:= true.
2	In this case no further seats are assigned.	We define $LISTSTAKINGPART_{step}$:= $LISTSTAKINGPART_{step-1}$ and $newseats_{step} \equiv 0$.

stap	*bewerking*	*Formalization*
3		D_{step}, J_{step} and M_{step} are defined as usual: $D_{step} := \emptyset$, $J_{step} := \emptyset$, $M_{step} := M_{step-1}$.
	Implementation notes:	This step type is implemented in the method, `GeneralSeatDistributor.switchCosalgModeOn()`. In the implementation the important part of this step is to memorize the result of the distribution to that point (number of seats for each list), because that is the primary result of the algorithm. The continued seat assignment does not alter this primary result, it only aims to calculate the roll forward sequence.

3 ELECTION TO THE HOUSE OF REPRESENTATIVES AND THE EUROPEAN PARLIAMENT

Implementation notes:

The calculations of the election result for all elections is performed in the method `ElectionResultDeterminator.calculate()`.

The differences between the election types are either determined by

- the number of electoral districts,
- the presence or absence of list groups,
- the fraction "preferential barrier / electoral quota" which is given by the methods `Election.getPreferencialBarrierNumerator()` and `getPreferencialBarrierDenominator()`,

directly by the election subcategory which is given by the method `Election.getElectionSubcategory()`

3.1 CHARACTERISTICS OF THE ELECTIONS

3.1.1 CHARACTERISTICS OF THE ELECTIONS TO THE HOUSE OF REPRESENTATIVES

The following limitations result from special specifications for the election to the House of Representatives in the Netherlands:

Stap	*Bewerking*	*Formalization*
1	There are 20 electoral districts for the election to the House of Representatives.	$\#DISTRICTS = 20$
2	Handing in of list is carried out centrally. Every list belongs to a set of 20 identical lists.	$\forall\ p3list \in P3LISTS: \#p3list = \#DISTRICTS$

3	Every P3-list is identical to a set of identical lists. There are no independent lists and no groups of lists. Every P2-list is also a P3-list and the other way around.	$\forall\ p3list \in P3LISTS$: $\#P2LISTS_{p3list} = 1$ $\forall\ p3list \in P3LISTS$: $P2LISTS_{p3list} = \{\ p3list\ \}$ $P3LISTS = P2LISTS$
	Implementation notes:	The number of electoral districts is not reflected in the source code.

3.1.2 Characteristics of the election for the European Parliament

The following limitations result from special specifications for the election of the European Parliament in the Netherlands:

Stap	*Bewerking*	*Formalization*
1	There are 19 electoral districts for the election of the European Parliament.	$\#DISTRICTS = 19$
2	Handing in of list is carried out centrally. Every list belongs to a set of 19 identical lists.	$\forall\ p3list \in P3LISTS$: $\#\ p3list = \#DISTRICTS$
3	Every P3-list is identical to a set of identical lists. There are no independent lists and no groups of lists. Every P2-list is also a P3-list and the other way around.	$\forall\ p3list \in P3LISTS$: $\#P2LISTS_{p3list} = 1$ $\forall\ p3list \in P3LISTS$: $P2LISTS_{p3list} = \{\ p3list\ \}$ $P3LISTS = P2LISTS$
	Implementation notes:	The number of electoral districts is not reflected in the source code.

3.2 Calculations prior to assignment of seats

3.2.1 Determination of total number of votes and calculation of the electoral quota

Legal regulation: article P 5 and U 7 electoral law.

Stap	*Bewerking*	*Formalization*

0		The number of votes for each candidate and electoral district *votes*(*district*, *candidate*) is determined as described in section 1.4.8.
1a	Van elk stel gelijkluidende lijsten en van elke lijstengroep worden de stemtotalen in alle kieskringen bij elkaar opgeteld.	The **total number of votes of a list** *list* = (*district*, (t_1, t_2, ...)) ∈ *LISTS* is $votes_1(list) := \sum_{i=1}^{\#list} votes(\text{district}, t_i)$ This defines a function $votes_1 : LISTS \rightarrow \mathbf{N_0}$.
1b		The **total number of votes cast for a P2-list** *p2list* ∈ *P2LISTS*, in particular for a set of identical lists is: $votes_2(p2list) := \sum_{list \in s} votes_1(list)$ This defines a function $votes_2 : P2LISTS \rightarrow \mathbf{N_0}$.
1c		The **total number of votes cast for a P3-list** *p3list* ∈ *P3LISTS*, in particular for a group of list, is: $votes_3(p3list) := \sum_{list \in \text{p3list}} votes_2(list)$ This defines a function $votes_3 : P3LISTS \rightarrow \mathbf{N_0}$.
1d		The **total number of all votes** is the sum of all votes for all lists. This is identical to the sum of the vote cast for all candidates in all electoral districts. $totalnumberofvotes := \sum_{list \in LISTS} votes_1(list) = \sum_{district \in DISTRICTS, candidate \in CANDIDATES} votes(district, \text{candidate})$
1e	De stemtotalen van de stellen gelijkluidende lijsten, de lijstengroepen en de op zichzelf staande lijsten worden bij elkaar opgeteld. Het resultaat van deze optelling is het totale aantal uitgebrachte stemmen.	Because *P2LISTS* and *P3LISTS* each is a partitioning of *LISTS* $totalnumberofvotes = \sum_{\text{p2list} \in P2LIST} votes_2(\text{p2list}) = \sum_{p3list \in P3LISTS} votes_3(p3list)$

2	Het totale aantal uitgebrachte stemmen wordt gedeeld door het aantal te verdelen zetels. Het resultaat van deze deling is de kiesdeler. *Implementation notes:*	The **electoral quota** is the quotient of the number of votes and the number of seats to be assigned. $electoralquota := \frac{\text{totalnumberofvotes}}{n}$ The total number of votes and the number of votes is calculated in the constructor of `VotesCounter` for • each candidate $candidate \in CANDIDATES$ in total • each candidate $candidate \in CANDIDATES$ in each P2-list $p2list \in P2LISTS$ • each candidate $candidate \in CANDIDATES$ in each P3-list $p3list \in P3LISTS$ • each candidate list $list \in LISTS$ • each P2-list $p2list \in P2LISTS$ • each P3-list $p3list \in P3LISTS$.

3.2.2 DETERMINATION OF THE NUMBER OF CANDIDATES

Regulations by law: article P 10, U 10 and P 19a (formerly P 18a) of the electoral law.

stap	*bewerking*	*Formalization*

1a	Indien een gekozen kandidaat is overleden, wordt deze bij de toepassing van deze paragraaf buiten beschouwing gelaten. We calculate the number of allocatable (i.e. not deceased) candidates for each candidate list.	In order to determine if a list, a P2-list or P3-list is exhausted, the total number of candidates must be known, that may be assigned a seat. For a list *list* = (*district*, *t*) = (*district*, (t_1, t_2, ..., $t_{\#list}$)) we define $CANDIDATES_{list} := \{t_1, t_2, ..., t_{\#list}\} \setminus DEADCANDIDATES$ the set of **allocatable candidates** (hence not deceased) of a list *list*. Let $noofcandidatsinlist_1(list) := \#CANDIDATES_{list}$ be the total number of allocatable candidates of list *list*. This defines a function $noofcandidatsinlist_1 : LISTS \to \mathbf{N_0}$.
1b	We calculate the number of allocatable (i.e. not deceased) candidates for each P2-list.	For a P2-list $p2list \in P2LISTS$ we define: $CANDIDATES_{p2list} := \bigcup_{list \in p2list} CANDIDATES_{list}$, $noofcandidatsinlist_2(p2list) := \#CANDIDATES_{p2list}$ This defines a function $noofcandidatsinlist_2 : P2LISTS \to \mathbf{N_0}$.
1c	We calculate the number of allocatable (i.e. not deceased) candidates for each P3-list.	For a P3-list $p3list \in P3LISTS$ we define: $CANDIDATES_{p3list} := \bigcup_{list \in p3list} CANDIDATES_{list}$, $noofcandidatsinlist_3(p3list) := \#CANDIDATES_{p3list}$ This defines a function $noofcandidatsinlist_3 : P3LISTS \to \mathbf{N_0}$.
1d	*Implementation notes:*	The number of allocatable candidates is calculated in the constructor of `CandidatesCounter` for • each candidate list $list \in LISTS$ • each P2-list $p2list \in P2LISTS$ • each P3-list $p3list \in P3LISTS$.

3.2.3 DETERMINATION OF THE TOTAL NUMBER OF VOTES

Legal regulation: none.

stap	*bewerking*	*Formalization*
	Implementation notes:	The constructor `VotesCounter()` calculates the total number of seats.

3.3 ASSIGNMENT OF SEATS TO P3-LISTS

Regulations by law: articles P 6, P 7, P 9, P 10 and Y 2 of the electoral law.

stap	*bewerking*	*Formalization*
1	The seat distribution to P3-lists for the European Parliament and House of Representatives election is performed as described in chapter 2 with the following characteristics:	The assignment of seats to P3-lists for the European Parliament and House of Representatives election follows the common description of chapter 2. The set of lists P referred to in chapter 2 to which the seats are assigned, is the set $P3LISTS$ of P3-lists in this case. This means the seat distribution to P3-lists is a function $seatstolist := A(seats, votes, noofcandidatsinlist, param) \in \mathbf{Z}^{P3LISTS}$
2		where $seats = n \in \mathbf{N}$ is the total number of seats to be allocated in the election, $votes \equiv votes_3 \in \mathbf{N_0}^{P3LISTS}$ is the function for the number of votes of the P3-lists, $noofcandidatsinlist \equiv noofcandidatsinlist_3 \in \mathbf{N_0}^{P3LISTS}$ is the function for the number of candidates of the P3-lists and $param = (param_2, ..., param_7)$ = (true, 0%, false, false, true, false) for all elections.

stap	*bewerking*	*Formalization*
3	If necessary, decisions are made by lot. Only P3-lists that reach the electoral quota may take part in the residual seat distribution (article P 7 subsection 2 of the electoral law). The method of largest remainder is not applied. Residual seats are assigned by the method of largest average without restriction to one seat per list. The absolute majority regulation must be considered.	Note that the boolean parameters are selected for the following reasons: • $param_2$ = true, because only lists that reach the electoral quota may take part in the residual seat distribution (article P 7 subsection 2 of the electoral law). • $param_3$ = 0% and $param_4$ = false, because the method of largest remainder is not applied. • $param_5$ = false, because the method of largest average is not restricted to one seat. • $param_6$ = true, because the absolute majority regulation must be considered. • $param_7$ = false, because the cosalg-mode is not needed.
4		So $seatstolist(p3list) \in \mathbf{Z}$ is the number of seats assigned to the P3-list $p3list \in P3LISTS$.
	Implementation notes:	This calculation is performed in the method `ElectionResultDeterminator.assignSeatsToP3Lists()`. Here the `GeneralSeatDistributor` is configured by the `GsdaParameters` created in the method `ElectionResultDeterminator.getGsdaParameters()` depending on the election subcategory. For EP and TK elections, the `GsdaParameters` are created in the static method `GsdaParameters.forP42DistributionEpTk()`.

3.4 ASSIGNMENT OF SEATS TO P2-LISTS (WITHIN P3-LISTS)

Regulations by law: article P 12, P 13, U 12 and U 13 electoral law.

Two methods are used when distributing the residual seats: first the **method of the largest remainder (according to Hare - Niemeyer)** and afterwards – if not all residual seats have been distributed yet – the **method of the largest average (according to d'Hondt).**

stap	*bewerking*	*Formalization*
	De hieronder vermelde stappen worden voor elke lijstengroep afzonderlijk uitgevoerd.	The following steps are carried out for each P3-list $p3list \in P3LISTS$.
1		For P3-lists $p3list \in P3LISTS$ that are equal to a P2-list, these steps are trivial. In this case $P2LISTS_{p3list} = \{ p3list \}$. The sub-distribution for these P3-lists $p3list \in P2LISTS$ is given by: $seatstolist^{(p3list)} : P2LISTS_{p3list} \rightarrow \mathbf{Z}$, $seatstolist^{(p3list)}(p3list) := seatstolist(p3list)$
1a		For the European Parliament and House of Representatives election $P3LISTS = P2LISTS$ and each element of $P2LISTS$ is a set of 19 or 20 identical lists. So the sub-distribution within P3-lists is always trivial.
	Implementation notes:	This step is implemented in the method `AbstractElectionResultDeterminator.assignSeatsToP2Lists()`.
2		For a P3-list $p3list \in P3LISTS$, that no seats were assigned to, we have $seatstolist(p3list) = 0$. For these lists the sub-distribution is trivial, too. It is given by: $seatstolist^{(p3list)} : P2LISTS_{p3list} \rightarrow \mathbf{Z}$, $seatstolist^{(p3list)}(p2list) := 0$
	Implementation notes:	The trivial cases of assignment of seats to P2-lists within a P3-list is implemented in the method `ElectionResultForP2ListsDeterminator.calculate()`.

stap	*bewerking*	*Formalization*
3	In the following the non-trivial assignment of seats to P2-lists within a P3-list is specified.	
4a	The seat assignment to P2-lists within a P3-list is performed as described in chapter 2 with the following characteristics:	The non-trivial assignment of seats to P2-lists within a given P3-list $p3list \in P3LISTS$ for all kinds of elections follows the common description of chapter 2. The set of lists P referred to in chapter 2 to which the seats are assigned, is the set $p3list$ of P2-lists in this case. This means the seat distribution to P2-lists within a given P2-list $p3list \in P3LISTS$ is a function $seatstolist^{(p3list)} := A(seats, votes, noofcandidatsinlist, param) \in \mathbf{Z}^{p3list}$
4b		where $seats = seatstolist(p3list) \in \mathbf{N}$ is the number of seats to be allocated within the P3-list, $votes \in \mathbf{N_0}^{p3list}$ is the function $votes_2$ for the number of votes of the P2-lists restricted to $p3list$, i.e. $votes(p2list) := votes_2(p2list)$ for all $p2list \in p3list$. $noofcandidatsinlist \in \mathbf{N_0}^{p3list}$ is the function $noofcandidatsinlist_2$ for the number of candidates of the P2-lists restricted to $p3list$, i.e. $noofcandidatsinlist(p2list) := noofcandidatsinlist_2(p2list)$ for all $p2list \in p3list$. $param = (param_2, ..., param_7) =$ (false, 0%, true, false, false, $param_7$). $param_7$ = true, if $p3list$ is a list group. $param_7$ = false, if $p3list$ is not a list group.
4c		So $seatstolist^{(p3list)}(p2list) \in \mathbf{Z}$ is the number of seats assigned to the P2-list $p2list \in p3list$.

stap	*bewerking*	*Formalization*
4d	All P2-lists that form the P3-list take part in the residual seat distribution. The method of largest remainder is applied. Remaining seats after distribution by largest remainder are assigned by the method of largest average without restriction to one seat per list. There is no absolute majority regulation when distributing seats to P2-lists. The roll back sequence and roll forward sequence must only be calculated for list groups.	Note that the boolean parameters are selected for the following reasons: • $param_2$ = false, because all lists $p2list \in p3list$ take part in the residual seat distribution. • $param_4$ = true, because the method of largest remainder is applied. • $param_3$ = 0%, because all lists $p2list \in p3list$ take part in the seat distribution by largest remainder. • $param_5$ = false, because the method of largest average is *not* restricted to one seat. • $param_6$ = false, because the absolute majority regulation must *not* be considered. • If the P3-list $p3list$ is a list group, i.e. if $\#P2LISTS_{p3list} > 1$, then $param_7$ = true, because the cosalg-mode is needed. This may only happen in the PS2 elections. If the P3-list is not a list group, then $param_7$ = false.
5		Each element $p2list \in P2LISTS$ is a subset of a unique element of $P3LISTS$. So after applying the above steps for each P3-list $p3list \in P3LISTS$, we define the seat distribution to all P2-lists $p2list \in P2LISTS$ as a function $seatstolist'' : P2LISTS \to \mathbf{Z}$ by combining the functions $seatstolist^{(p3list)} : P2LISTS_{p3list} \to \mathbf{Z}$ defined above. So set $seatstolist''(p2list) := seatstolist^{(p3list)}(p2list)$ where $p3list$ is the unique element of $P3LISTS$ with $p2list \subset p3list$.

stap	*bewerking*	*Formalization*
5a		For the European Parliament and TK election, as a consequence of all P3-lists being a set of 19 or 20 identical lists, the functions *seatstolist* and *seatstolist''* are equal.
	Implementation notes:	This calculation is performed in the method `ElectionResultDeterminator.assignSeatsWithinP3Lists()`. Here for each P3-list an `ElectionResultForP2ListsDeterminator` is created that performs the distribution within that P3-list. The `ElectionResultForP2ListsDeterminator` uses a `GeneralSeatDistributor` that is configured using the `GsdaParameters.forP2P3Distribution()` or the `GsdaParameters.forP2DistributionTkPs2()` for TK or PS2 elections.

3.5 NOMINATION OF ELECTED CANDIDATES

Below is determined for each P3-list, which of the candidates has been elected. This description is - with the exception of the preferential barrier - valid for all election types.

A candidate, whose death has been put on record, is disregarded (article P 19a electoral law - formerly article P 18a).

3.5.1 PREFACE ON ORDER RELATIONS

In the description on how the elected candidates are nominated, the order in which things happen plays an important role. So it is necessary to sort sets of candidates by different criteria. To formalize this, we already introduced some commonly used mathematical terms in section 1.2.3. In this section we add some definitions needed to formally describe the nomination of elected candidates.

stap	*bewerking*	*Formalization*

1

Let A be a finite non-empty set with a total preorder $\leq$. Let

$$\mathbf{P}(A) := \{ B \mid B \subset A \}$$

be the power set of A. The power set of a set A is the set of all subsets of A. Then we define functions

$$O_{A,\leq}: A \to \mathbf{P}(A)\ ,\ \ O_{A,\leq}(a) := \{ b \in A \mid b \leq a \}$$

and

$$o_{A,\leq}: A \to \{ 1, 2, ..., \#A \}\ ,\ \ o_{A,\leq}(a) := \#O_{A,\leq}(a).$$

2

If $\leq$ is a total order, it is easy to see that $o_{A,\leq}$ is a bijection, i.e. it maps the elements of A to the integers $\{ 1, 2, ..., \#A \}$ one-by-one. So there is a reverse mapping

$$o_{A,\leq}^{-1}: \{ 1, 2, ..., \#A \} \to A.$$

3

Using this unique mapping, we can write the smallest element in A with respect to the total order $\leq$ as $o_{A,\leq}^{-1}(1)$, the second smallest as $o_{A,\leq}^{-1}(2)$ etc. and the largest as $o_{A,\leq}^{-1}(\#A)$. I.e. we have

$$o_{A,\leq}^{-1}(1) \leq o_{A,\leq}^{-1}(2) \leq ... \leq o_{A,\leq}^{-1}(\#A).$$

4

To simplify the notation we write

$$\min_{A,\leq} := o_{A,\leq}^{-1}(1).$$

If from the context it is clear which order relation on A is meant, we write even shorter

$$\min A := o_{A,\leq}^{-1}(1).$$

	Implementation notes:	It is clear that in the implementation we can obtain this ordering by using any sorting algorithm and sort the elements of A by the total order $\leq$. So in the following we will use the notation $o_{A,\leq}^{-1}(i)$ to denote the i-th smallest element etc. but in the implementation we will use the sorting algorithms provided by the Java implementation and access the element in the sorted collection by index.
5	Inherited total order in the equivalence classes of a total preorder.	In the cases where we are dealing with a total preorder $\leq$ of A that is not a total order, normal sorting does not result in a unique result. There may be distinct elements $a, b \in A$ with $a \leq b$ and $b \leq a$ in this case. The total preorder $\leq$ defines an equivalence relation $\sim_{\leq}$ of A by $a \sim_{\leq} b :\Leftrightarrow a \leq b$ and $b \leq a$. Let $E_{A,\leq} \subset \mathbf{P}(A)$ be the equivalence classes of the equivalence relation $\sim_{\leq}$. Now we inherit a total order $\leq$ of $E_{A,\leq}$ by defining for all $e, f \in E_{A,\leq}$ $e \leq f :\Leftrightarrow \exists a \in e, b \in f : a \leq b$.
	Implementation notes:	To determine the order of the equivalence classes, we use the implementation in the static utility method `OrderUtil.sortAndGroup()`.

3.5.2 NOMINATION OF CANDIDATES ELECTED BY PREFERENTIAL VOTE

Regulations by law: article P 15, P 16, U 15, U 16 and Y 23a electoral law.

The steps in this section are performed for each P3-list $p3list \in P3LISTS$.

stap	*bewerking*	*Formalization*
1a	De voorkeurdrempel wordt berekend. It depends on the type of the election. For the European Parliament election it is 10% of the electoral quota (article Y 23a electoral law).	For the EP election: *preferentialbarrier* := *electoralquota* / 10
1b	For the election for municipality councils, island councils with less than 19 seats and borough councils in Amsterdam it is 50% of the electoral quota (article P 15 subsection 2 electoral law).	For the GR1, ER and BC elections: *preferentialbarrier* := *electoralquota* / 2

stap	*bewerking*	*Formalization*
1c	For the election for borough councils in Rotterdam it is 100% of the electoral quota (article 40 subsection c. in the Kiesreglement gebiedscommissies).	For the GC elections: *preferentialbarrier* := *electoralquota*
1d	For all other elections it is 25% of the electoral quota (article P 15 subsection 1 electoral law).	For all other elections (TK, PS2, PS1, AB1, AB2, GR2): *preferentialbarrier* := *electoralquota* / 4
1e	*For the election to the Senate it is 100% of the electoral quota (article U 15 subsection 3 electoral law).*	For the EK elections: *preferentialbarrier* := *electoralquota*
	Implementation notes:	The preferential barrier is calculated in the constructor `ElectionResultForCandidatesDeterminator()` using the methods `Election.getPreferencialBarrierNumerator()` and `getPreferencialBarrierDenominator()`.

stap	*bewerking*	*Formalization*
2	Van elke kandidaat wordt het stemtotaal in alle kieskringen gezamenlijk vastgesteld.	Recall that $votes_{p3list}(c)$ is the number of votes of a candidate c on a P3-list $p3list \in P3LISTS$.
3a	For all but EK elections: Vastgesteld wordt welke kandidaten een stemtotaal hebben dat hoger is dan de voorkeurdrempel.	So we need to determine the candidates with $votes_{p3list}(c) > preferentialbarrier$ and $votes_{p3list}(c) \geq preferentialbarrier$. For the P3-list $p3list \in P3LISTS$ we are currently regarding and any real number $x \in \mathbf{R}$ we define the sets $C'_{p3list}(x) := \{ c \in C_{p3list} \mid votes_{p3list}(c) > x \}$ and $noofcandidatsinlist'_{p3list}(x) := \#C'_{p3list}(x)$ $C''_{p3list}(x) := \{ c \in C_{p3list} \mid votes_{p3list}(c) \geq x \}$ and $noofcandidatsinlist''_{p3list}(x) := \#C''_{p3list}(x)$ So for a given number x, $noofcandidatsinlist'_{p3list}(x)$ defines the number of candidates that have more than x votes and $noofcandidatsinlist''_{p3list}(x)$ defines the number of candidates that have at least x votes. So $C'_{p3list}(preferentialbarrier)$ is the set of allocatable candidates with the total amount of votes that is higher than the preferential barrier. For all but EK elections we define $B_{p3list} := C'_{p3list}(preferentialbarrier)$.

stap	*bewerking*	*Formalization*
3b	For EK elections: In afwijking van de artikelen P 15, eerste lid, en P 19, tweede lid, zijn gekozen onderscheidenlijk worden gerangschikt in de volgorde van de aantallen op hen uitgebrachte stemmen, de kandidaten die op de gezamenlijke lijsten waarop zij voorkomen een aantal stemmen hebben verkregen dat groter is dan of gelijk is aan de kiesdeler.	So $C''_{p3list}(preferentialbarrier)$ is the set of allocatable candidates with the total amount of votes that is higher than or equal to the preferential barrier. For EK elections we define $B_{p3list} := C''_{p3list}(preferentialbarrier)$.
	Implementation notes:	$noofcandidatsinlist'_{p3list}(x)$ is implemented in the method `getNumberOfCandidatesWithMoreVotes()`, $noofcandidatsinlist''_{p3list}(x)$ is implemented in the method `getNumberOfCandidatesWithMoreOrEqualVotes()`. B_{p3list} is implemented indirectly. The elements of B_{p3list} are those candidates where the local variable `isAbovePreferencialBarrier` in the method `getCandidatesForSorting()` is `true`. This variable is also a parameter in the Constructor `CandidateForSorting()`. So elements of B_{p3list} correspond to those `CandidateForSorting` where the method `isAbovePreferencialBarrier()` returns `true`.

stap	*bewerking*	*Formalization*
4	De in stap 3 bedoelde kandidaten worden in de volgorde van het in totaal in alle kieskringen gezamenlijk behaalde aantal stemmen gekozen verklaard. Er worden niet meer kandidaten gekozen verklaard dan er zetels aan de P3-lijst zijn toegewezen. - Er kunnen dus kandidaten zijn die een stemtotaal hebben dat hoger is dan de voorkeurdrempel, maar geen zetel krijgen.	Every candidate $candidate \in B_{p3list}$ with $noofcandidatsinlist''_{p3list}(votes_{p3list}(candidate)) \leq seatstolist'(p3list)$ receives a priority seat.

stap	*bewerking*	*Formalization*
5	Indien de aantallen behaalde stemmen gelijk zijn en niet voor elk van die aantallen een kandidaat gekozen kan worden verklaard, vindt loting plaats in de zitting van het centraal stembureau.	For candidates $candidate \in B_{p3list}$ with $noofcandidatsinlist'_{p3list}(votes_{p3list}(candidate)) < seatstolist'(p3list) < noofcandidatsinlist''_{p3list}(votes_{p3list}(candidate))$ that have equality of votes, the priority seat is drawn by lot. In this case, the number of candidates participating in the drawing lots, is $\alpha := noofcandidatsinlist''_{p3list}(votes_{p3list}(candidate)) - noofcandidatsinlist'_{p3list}(votes_{p3list}(candidate))$. The amount of seats that are assigned within this drawing lots is $\beta := seatstolist'(p3list) - noofcandidatsinlist'_{p3list}(votes_{p3list}(candidate))$. Now we define a function o_2: $B_{p3list} \rightarrow \mathbf{N_0}$ as follows: The candidates are drawn one by one. For the first candidate that is drawn we set $o_2(candidate) := \beta$. In case $\beta > 1$, for the second candidate that is drawn we set $o_2(candidate) := \beta - 1$ etc.. For the last candidate that is drawn we set $o_2(candidate) := 1$. These are the candidates that receive a priority seat by lot. For all other candidates - both candidates that did not take part in the drawing lots and candidates that were not drawn - we set $o_2(candidate) := 0$. A candidate $candidate \in B_{p3list}$ with $seatstolist'(p3list) \leq noofcandidatsinlist'_{p3list}(votes_{p3list}(candidate))$ receives no seat.
6		So the candidates that receive a priority seat are $P_{p3list} := \{ candidate \in B_{p3list} \mid noofcandidatsinlist''_{p3list}(votes_{p3list}(candidate)) \leq seatstolist'(p3list) \vee o_2(candidate) > 0 \}$.
	Implementation notes:	The candidates with priority seats are determined in the method `ElectionResultForCandidatesDeterminator.getCandidatesWithPrioritySeat()`.
7		In case there is more than one P2-list in the P3-list, the order in which these candidates receive their seats is relevant for the decision on which P2-list they are elected. This order is given as follows: • Order the candidates in P_{p3list} by the number of votes $votes_{p3list}(candidate)$ (largest number of votes first) • In case the numbers of votes are equal, order them by $o_2(candidate)$ (largest value first) as described in step 5 • Using only these two criteria, the ordering is not unique (it is a total preorder, but may not be a total order), because there may be candidates with an equal number of votes that receive a seat without drawing lots. In this case, these candidates receive their seats simultaneously.

stap	*bewerking*	*Formalization*
8		So we define on the set P_{p3list} a total preorder relation ≤ by: $candidate \leq candidate' :\Leftrightarrow votes_{p3list}(candidate) > votes_{p3list}(candidate') \vee$ $votes_{p3list}(candidate) = votes_{p3list}(candidate') \wedge o_2(candidate) > o_2(candidate')$.
9		From this total preorder relation ≤ of P_{p3list} we inherit a total order relation ≤ on the equivalence classes $E_{P_{p3list},\leq}$ of P_{p3list}. For $E_{P_{p3list},\leq}$ we will write E_{p3list} as a shorter notation. Sorting these equivalence classes E_{p3list} in ascending order by this total order relation ≤ results in a unique order $E_{p3list} = \{ C_1, C_2, \ldots, C_h \}$ where $C_i := o^{-1}_{E_{p3list},\leq}(i)$, $h = \#E_{p3list}$ and $C_1 \leq C_2 \leq \ldots \leq C_h$. Now the priority seats are assigned to these candidates in this order.
	Implementation notes:	The sorted equivalence classes are calculated in `OrderUtil.sortAndGroup()` and held in the variable `candidatesWithPrioritySeatSorted`. The defined order is implemented in `SortCandidatesUtil.COMPARATOR_FOR_PREF_SEATS`.

stap	*bewerking*	*Formalization*
10	For each of these candidates that receives a priority seat, we determine the P2-list on which the candidate is elected. This is done step by step for one candidate in each step in the given order. Only if candidates receive their priority seats simultaneously, the corresponding P2-list will be determined in one step.	Let $step$ = 1, 2, ..., h be the step number. In step $step$ we assign priority seats to the candidates in C_{step}. Let ξ_0 := 0. ξ_{step} will be defined recursively below. It is used later to indicate how much of the roll back sequence is "consumed" up to step $step$.
	Implementation notes:	For each set C_{step} the method `P2ListForPrioritySeatsDeterminator.determineP2ListsForElectedCandidates()` is called once and performs the following calculations up to step 18e. The result of the calculation steps is consecutively written into the `SeatDistributionInP3List` instance in the attribute `_dist`.
11		The set of candidates that are considered as elected in the P2-list $p2list \in P2LISTS_{p3list}$ in the steps up to $step$ is denoted by $E_{step}(p2list) \subset CANDIDATES$ and will be defined below. For all $p2list \in P2LISTS_{p3list}$ we define $E_0(p2list) := \varnothing$.
12		Below we will define $\phi_{step}(p2list) \in \mathbf{N_o}$ to be the number of seats of the P2-list $p2list \in P2LISTS_{p3list}$ that are still unassigned to candidates after step $step$. Define $\phi_0(p2list) := seatstolist''(p2list)$ for all $p2list \in P2LISTS_{p3list}$.
13		Recall that $votes_{p2list}(candidate)$ is the number of votes of a candidate $candidate$ on a P2-list $p2list \in P2LISTS$.

stap	*bewerking*	*Formalization*
14		Let $\sigma_{step}(candidate) \in P2LISTS_{p3list}$ be the P2-list where candidate $candidate \in C_{step}$ is elected. σ_{step} is defined as follows:
15a		For the current value of *step*, we want to determine on which P2-list the candidates $candidate \in C_{step}$ receive their seats. This may also require multiple steps. So, we iterate in an inner loop using the variable j = 1, 2, ... j_{max}. In most cases the inner loop only needs one single iteration (i.e. j_{max} = 1).
15b		In each step j of the inner loop we define $\phi_{step\text{-}1,j}(p2list) \in \mathbf{N_0}$ to be the number of seats of the P2-list $p2list \in P2LISTS_{p3list}$ that are still unassigned to candidates after step j. For j = 0 define $\phi_{step\text{-}1,0}(p2list) := \phi_{step\text{-}1}(p2list)$ for all $p2list \in P2LISTS_{p3list}$.
15c		The candidates of C_{step} that have not received a seat in the inner loop before step j are given by $C_{step,j\text{-}1}$. So define $C_{step,0} := C_{step}$.

stap	*bewerking*	*Formalization*
15d	De kandidaat geldt als gekozen op een P2-lijst waarop nog zetels te verdelen zijn en van die lijsten de lijst waarop hij het hoogste aantal stemmen behaalde. Indien aantallen gelijk zijn, geldt hij als gekozen op de P2-lijst met het laagste kieskringnummer. - Het kan voorkomen dat een kandidaat het hoogste aantal stemmen heeft behaald op een P2-lijst waarop geen zetels meer zijn te verdelen. Die P2-lijst blijft niettemin buiten beschouwing.	For each candidate $candidate \in C_{step,j\text{-}1}$ determine the P2-lists $X_{candidate,j} = \{\ p2list \in P2LISTS_{p3list} \mid candidate \in C_{p2list} \wedge \phi_{step\text{-}1,j\text{-}1}(p2list) > 0\ \} \subset P2LISTS_{p3list}$. These are the lists on which candidate c is nominated and that still have at least one unassigned seat. If $X_{candidate,j} \neq \emptyset$, let $\sigma_{candidate,j} \in X_{candidate,j}$ be the P2-list on which c received the most votes or in case of equal votes, that is nominated in the electoral district with the smallest electoral district number. This is done as follows: Define on the set $X_{candidate,j}$ the total order relation $\leq$ by $p2list \leq p2list' :\Leftrightarrow votes_{p2list}(candidate) > votes_{p2list'}(candidate) \vee$ $votes_{p2list}(candidate) = votes_{p2list'}(candidate) \wedge idx(p2list) \leq idx(p2list')$. So in this case set $\sigma_{candidate,j} :=$ min $X_{candidate,j}$. $\sigma_{candidate,j}$ is the P2-list where candidate *candidate* is elected provided that there are enough seats available.

stap	*bewerking*	*Formalization*
15e		Now for each of the P2-lists $p2list \in P2LISTS_{p3list}$ we need to check if there are enough seats available. The set of candidates $Y_{p2list,j} := \{ candidate \in C_{step} \mid \sigma_{candidate,j} = p2list \}$ contains the candidates that should receive a seats in *p2list* provided that there are enough seats available. Order the elements of $Y_{p2list,j}$ by the position on the list *p2list*, i.e. define on $Y_{p2list,j}$ the total order relation $candidate \leq candidate' :\Leftrightarrow p_{p2list}(candidate) \leq p_{p2list}(candidate')$.
15f		Recall the definition of the function $o_{Y_{\mathrm{p2list},j},\leq}: Y_{\mathrm{p2list},j} \to \{1,2,\dots,\#Y_{\mathrm{p2list},j}\}$in section 3.6.1. step 1. The first $\phi_{step\text{-}1,j\text{-}1}(p2list)$ candidates from $Y_{p2list,j}$ can receive a seat. More precisely, let $Z_{p2list,j} := \left\{ candidate \in Y_{\mathrm{p2list},j} \middle| o_{Y_{\mathrm{p2list},j},\leq}(candidate) \leq \varphi_{\mathrm{step}-1,j-1}(p2list) \right\}$. These are the candidates that receive a seat on P2-list *p2list* in the inner step j. So, the number of available seats decreases accordingly: Define $\phi_{step\text{-}1,j}(p2list) := \phi_{step\text{-}1,j\text{-}1}(p2list) - \#Z_{p2list,j}$. Also define $\sigma_{candidate} := p2list$ for all $candidate \in Z_{p2list,j}$.
15e		The total set of candidates that receive a seat in the inner step j is given by $Z_j := \bigcup_{\mathrm{p2list} \in S_{p3list}} Z_{\mathrm{p2list},j}$. Let $C_{step,j} := C_{step,j\text{-}1} \setminus Z_j$.

stap	*bewerking*	*Formalization*
15f		This inner loop is repeated until the following condition holds: $\forall\ candidate \in C_{step,j} : X_{candidate,j+1} = \varnothing$. The first value for which this is true is defined to be j_{max}. The condition is especially true if $C_{step,j} = \varnothing$, but $C_{step,j} = \varnothing$ is not necessary. Define $R_{\text{step}} \coloneqq C_{\text{step},j_{\max}}$. These are the remaining candidates that did not receive a seat by the application of article P 16 subsection 1.
	Implementation notes:	This calculation is performed in the `P2ListForPrioritySeatsDeterminator`. The variables used in this calculation are represented as follows:

stap	*bewerking*	*Formalization*
		• C_{step} and the $C_{step,j}$ are represented by the variable `c_i` in `determineP2ListsForElectedCandidates()` • $E_{step}(p2list)$ is represented by `SeatDistributionInP3List.elected.get(p2list)`, i.e. in the corresponding map entry in the attribute `elected` of type `Map<P2List, Set<CandidateForSorting>>` in the class `SeatDistributionInP3List`. • $\phi_{step}(p2list)$ and $\phi_{step\text{-}1,j}(p2list)$ are represented by `SeatDistributionInP3List.remainingSeats.get(p2list)` , i.e. in the corresponding map entry in the attribute `remainingSeats` of type `Map<P2List, Long>` in `SeatDistributionInP3List`. • The $\sigma_{candidate,j}$ for those candidates with $X_{candidate,j} \neq \varnothing$ are represented by the map preferredP2Lists and calculated in `getPreferredP2Lists()`. • The $Y_{p2list,j}$ for $p2list \in P2LISTS_{p3list}$ are represented by the map `preferringCandidates`. • The total order relation on $Y_{p2list,j}$ is represented by the comparator `SortCandidatesUtil.sortByPositionOnP2List(p2List)` • The size of $\#Z_{p2list,j}$ is given by the variable `minOfRemainingSeatsAndNoOfCandidates`. • $Z_{p2list,j}$ and Z_j are not represented explicitly in the code. • The condition $X_{candidate,j+1} = \varnothing$ in step 15f corresponds to the condition `containsNotNullValue(preferredP2Lists)` in the source code. • R_{step} is also represented by the variable `c_i`
15g		In the inner loop, $\sigma_{candidate}$ has been defined for all $candidate \in C_{step} \setminus R_{step}$. For the remaining candidates $candidate \in R_{step}$, σ_c is defined as follows:

stap	bewerking	*Formalization*
16	Indien op geen van de P2-lijsten waarop de kandidaat is vermeld, nog zetels te verdelen zijn, geldt hij niettemin als gekozen op de lijst waarop hij het hoogste aantal stemmen behaalde.	For $candidate \in R_{step}$, let $\sigma_{candidate} \in P2LISTS_{p3list}$ be the P2-list on which $candidate$ received the most votes or in case of equal votes, that is nominated in the electoral district with the smallest electoral district number. This is done as follows: Determine $Y = \{p2list \in \text{P2LIST}S_{p3list} \mid candidate \in C_{p2list} \wedge \varphi_{\text{step}-1,j_{\max}}\ \}$. Define on Y the same total order relation $\leq$ as defined in step 15d. Set $\sigma_{candidate}$:= min Y. Now we have defined $\sigma_{candidate}$ for all $candidate \in C_{step}$.
	Implementation notes:	The min Y is calculated in the method `P2ListForPrioritySeatsDeterminator.getP2ListWithMostVotesFor()`
17		Define $$E_{\text{step}}(p2list) := E_{\text{step}-1}(p2list) \cup \{candidate \in C_{\text{step}} \mid \sigma_{candidate} = p2list\}.$$
18a		Finally we need to define the function ϕ_{step}. This is mostly given by $\varphi_{\text{step}-1,j_{\max}}$. Only in case $R_{step} \neq \emptyset$, the corresponding number of seats needs to pass from other P2-lists to those P2-lists where the candidates in R_{step} were declared elected.
18b		We need to iterate over the elements of R_{step}. So let $j'_{max} := \#R_i$ and define in the following recursively the function $\phi'_{step\text{-}1,j'}(p2list)$ and the pointer $\xi_{step\text{-}1,j'} \in \mathbf{N}$ for all $j' = 1, \ldots, j'_{max}$. For $j' = 0$ define $\phi'_{step\text{-}1,0}(p2list) := \varphi_{\text{step}-1,j_{\max}}\ (p2list)$ and $\xi_{step\text{-}1,0} := \xi_{step\text{-}1}$. In each iteration we determine from the roll back sequence the P2-list from which a seat is passed to another list. For this P2-list we decrease the number of available seats by 1.

stap	*bewerking*	*Formalization*
18c	Is stap 16 van toepassing, dan vervalt daartegenover een zetel op een P2-lijst waarop nog zetels te verdelen zijn, en van die P2-lijsten de P2-lijst waaraan bij de verdeling van de zetels binnen de P3-lijst het laatst een restzetel is toegewezen.	Convert the roll back sequence *rbs* to a to a tuple of P2-lists $(p2list_1, p2list_2, \ldots, p2list_{k_{max}})$ as described in the section 3.6.5. Let $\xi_{step\text{-}1,j'} := \min \{ k \in \mathbb{N} \mid \xi_{step\text{-}1,j'\text{-}1} < k \leq k_{max}$ and $\phi_{step\text{-}1,j'\text{-}1}(p2list_k) > 0 \}$. Now let $p2list' := p2list_{\xi_{step-1,j'}}$ and define $\phi_{step\text{-}1,j'}(p2list) := \phi_{step\text{-}1,j'\text{-}1}(p2list)$ for all $p2list \neq p2list'$ and $\phi_{step\text{-}1,j'}(p2list') := \phi_{step\text{-}1,j'\text{-}1}(p2list') - 1$.
18d		It is possible that the set used to define $\xi_{step\text{-}1,j'}$ is empty. In that case the minimum of that set would be ∞ (infinity). This happens if the whole roll back sequence is "consumed". In that case, we determine the P2-list *p2list'* from the non-empty set $\{ p2list \in P2LISTS_{p3list} \mid \phi_{step\text{-}1,j'\text{-}1}(p2list) > 0 \}$. From this set select the P2-list with the smallest electoral district number.
18e		After the iteration we can set $\xi_{\text{step}} := \xi_{\text{step}-1,j'_{\max}}$ and $\varphi_{\text{step}} := \varphi'_{\text{step}-1,j'_{\max}}$.
	Implementation notes:	The correction of the function of unassigned seats ϕ_{step} is calculated in the method `passSeatsFromOtherP2Lists()`.
19		Let $E_{\text{step},2} := \bigcup_{p2list \in P2LISTS_{p3list}} E_{\text{step}}(p2list)$ be the set of candidates of $p3list \in P3LISTS$ that were assigned a seat with the first *step* steps.

stap	*bewerking*	*Formalization*
20		**Note:** For each step number *step* = 1, ..., $\#E_{p3list}$, the following statements hold. Recall that $h = \#E_{p3list}$: • The $E_{step}(p2list)$, $p2list \in P2LISTS_{p3list}$ are disjoint subsets of P_{p3list}, which again is a subset of C_{p3list}. • $E_{step,2} = E_{step\text{-}1,2} \cup C_{step} = C_1 \cup C_2 \cup \ldots \cup C_h$ } • $\sum_{k=1}^{\text{step}} \#C_k =$ $E_{\text{step},2} = \sum_s$ • $\phi_{step}(p2list) \geq 0$ for all $p2list \in P2LISTS_{p3list}$ • $\sum_{p2list \in P2LISTS_{p3list}} \varphi_{\text{step}}(\text{p2list}) = m''(p3list) - \#C_{step+1} - \ldots - \#C_h$ So especially for $n = \#P_{p3list}$ we have: $n =$ $\cup_{p2list \in P2LISTS_{p3list}} E_h(p2list) = \#E_{h,2} = \sum_{p2list \in P2LISTS_{p3list}}$, so $\cup_{p2list \in P2LISTS_{p3list}} E_h(p2list) = P_{p3list}$. So all candidates that received a priority seat in the P3-list $p3list \in P3LISTS$ are elected in one and only one of the P2-lists $p2list \in P2LISTS_{p3list}$.

3.5.3 NOMINATION OF ALL REMAINING CANDIDATES

Regulations by law: article P 17 and P 18 electoral law.

The steps in this section are performed for each P3-list $p3list \in P3LISTS$ subsequent to the nomination of candidates elected by preferential vote as described in the previous section.

The yellow text comes from the Vaststellen_van_de_uitslag_van_verkiezingen_NL.doc from the TK election section.

stap	*bewerking*	*Formalization*
1		If $seatstolist'(p3list) = \#P_{p3list}$, all seats of the P3-list $p3list \in P3LISTS$ have already been assigned to elected candidates in section 3.6.2. In this case, none of the remaining candidates is elected.

stap	*bewerking*	*Formalization*
2		Otherwise again the seats are assigned in multiple steps. We will continue the step numbering from section 3.6.2. So the step numbers are $step = h + 1, h + 2, \ldots$.
3	Vastgesteld wordt op welke tot de P3-lijst behorende P2-lijsten nog resterende zetels zijn waarvoor kandidaten gekozen verklaard moeten worden.	Recall that the number of seats of the P2-list $p2list \in P2LISTS_{p3list}$ that are still unassigned to candidates after step $step$ - 1 is given by $\phi_{step\text{-}1}(p2list)$.
4a		For each P2-list $p2list \in P2LISTS_{p3list}$, let $E_{step,3}(p2list) := C_{p2list} \setminus E_{step\text{-}1,2}$ be the set of candidates of $p2list \in P2LISTS_{p3list}$ that were not assigned a seat in the preceding steps ("unelected candidates").
4b		Earlier than documented by the kiesraad we handle the special case that on a P2-list there are still unassigned seats but no more unelected candidates, i.e. $\phi_{step\text{-}1}(p2list) > 0$ and $\#E_{step,3}(p2list) = 0$. In the documentation of the kiesraad this is described after stap 16 like this: "Zijn er bij de toepassing van stap 16 op een P2-lijst geen kandidaten meer over die nog niet gekozen zijn verklaard, dan wordt op een P2-lijst van de P3-lijst waarop geen zetels meer te verdelen waren, de eerstvolgende nog niet gekozen verklaarde kandidaat gekozen verklaard. Deze P2-lijst wordt bepaald met toepassing van de regeling voor overgang van zetels naar andere verbonden P2-lijsten bij verdeling van zetels binnen een P3-lijst."

stap	*bewerking*	*Formalization*
5		At this point of the algorithm it is easy to foresee the described situation. And at this point it is (relatively) simple to perform the described steps of declaring candidates from other P2-lists elected without any special treatment. All we need to do is to pass the unassigned seat from the P2-list that does not have enough unelected candidates to another P2-list that has more than enough unelected candidates. We will handle this by "correcting" the function $\phi_{step\text{-}1}$ of free seats of the P2-lists as follows. The "corrected" function will be denoted by $\phi'_{step\text{-}1}$.
6		The P2-list that shall receive the seat that cannot be assigned, shall be determined in the same way as described in the seat assignment to P2-lists within a P3-list. The procedure described in section 3.5 shall for this purpose be continued similarly to the case that a list is exhausted. This has already been done when the roll forward sequence *rfs* was determined. The *rfs* is then converted to a tuple of P2-lists $(\mathrm{p2list}_1, \mathrm{p2list}_2, \ldots, \mathrm{p2list}_{k_{\max}})$ as described in the following section 3.6.4. These P2-lists shall in the given order receive further seats from other P2-lists in the given situation.
7		Given this tuple $(p2list_1, p2list_2, \ldots, p2list_{k_{\max}})$ we set $\phi'_{step\text{-}1,0}(s) := \phi_{step\text{-}1}(p2list)$ for all $p2list \in P2LISTS_{p3list}$. We also define for $step = h, h+1, \ldots$ the variable ξ_{step}. Set $\xi_h := 0$, the values ξ_{step} for $step > h$ will be defined later. Also set $\xi'_{step\text{-}1,0} := \xi_{step\text{-}1}$. The variables ξ_{step} and $\xi'_{step,\,j}$ play the role of "pointers" to the elements of the tuple, i.e. give the "position" in the tuple that we have reached after the respective iteration step.

stap	*bewerking*	*Formalization*
8		Check the termination condition defined in step 11 for the value $j = 0$. If this is true, continue at step 12. Otherwise repeat steps 9 to 11 for $j = 1, 2, \ldots$. We call this iteration the inner loop.
9		Define $Z := \{ p2list \in P2LISTS_{p3list} \mid \phi'_{step\text{-}1,j\text{-}1}(p2list) > \#E_{step,3}(p2list) \}$. Z is the set of P2-lists that have more unassigned seats than unelected candidates. We know from previously checking the termination condition, that $Z \neq \emptyset$. Select from Z the element s^- with the smallest electoral district number. This is done as follows: Define on the set Z the total order relation $\leq$ by $p2list \leq p2list' :\Leftrightarrow idx(p2list) \leq idx(p2list')$. So in this case set $p2list^-$:= min Z. This is the P2-list from which a seat shall be passed to another P2-list.
10		Find the index of the P2-list that shall receive this seat. It is given by $\xi'_{step\text{-}1,j} := \min \{ k \in \mathbb{N} \mid \xi'_{step\text{-}1,j\text{-}1} < k \leq k_{max}$ and $\phi'_{step\text{-}1,j\text{-}1}(p2list_k) < \#E_{step,3}(p2list_k) \}$. Define $p2list^+ := p2list_{\xi'_{step-1,j}}$ $\phi'_{step\text{-}1,j}(p2list^+) := \phi'_{step\text{-}1,j\text{-}1}(p2list^+) + 1$ $\phi'_{step\text{-}1,j}(p2list^-) := \phi'_{step\text{-}1,j\text{-}1}(p2list^-) - 1$ $\phi'_{step\text{-}1,j}(p2list) := \phi'_{step\text{-}1,j\text{-}1}(p2list)$ for all $p2list \in P2LISTS_{p3list} \setminus \{ p2list^+, p2list^- \}$

stap	*bewerking*	*Formalization*
11		If the "termination condition" $\forall\ p2list \in P2LISTS_{p3list}\colon \phi'_{step\text{-}1,j}(p2list) = 0 \vee \#E_{step,3}(p2list) \geq 0$ holds for a value j, terminate (or for $j = 0$ skip) the iteration and continue at steps 12. If the condition is not true, repeat the steps 9 to 11 for the next value $j + 1$.
12		So this terminates the inner loop. Let j_{max} be the value for j for which the termination condition in step 11 was true. This may even be the value $j_{max} = 0$. Define $\varphi'_{(\text{step}-1)}(\text{p2list}) \coloneqq \varphi'_{(\text{step}-1,j_{max})}$ for all $p2list \in P2LISTS_{p3list}$ and $\xi_{\text{step}} \coloneqq \xi'_{(\text{step}-1,j_{max})}$.
13		This way we have defined the "corrected" function of unassigned seats $\phi'_{step\text{-}1}$.
	Implementation notes:	The calculation of the steps 4a - 13 are performed in the method `handleLateListExhaustion()`.
14		Recall that for a given candidate $candidate \in \tilde{C}_{\text{p2list}}$, the position of *candidate* on the P2-list *p2list* is given by $p_{p2list}(candidate)$. So the position of *candidate* only regarding the elements of $E_{step,3}(p2list)$, i.e. the unelected candidates, is $p_{p2list,step}(candidate) := \#\{\ b \in E_{step,3}(p2list) \mid p_{p2list}(b) \leq p_{p2list}(candidate)\ \}$.

stap	*bewerking*	*Formalization*
15	Tot het op elke P2-lijst resterende aantal toegekende zetels worden nog niet gekozen verklaarde kandidaten van de P2-lijst in de volgorde van de P2-lijst gekozen verklaard.	From the P2-list $p2list \in P2LISTS_{p3list}$, the following candidates will be assigned a seat: $E_{step,4}(p2list) := \{ candidate \in E_{i,3}(p2list) \mid p_{p2list,step}(andidatec) \leq \phi'_{step-1}(p2list) \}$. These candidates are "declared elected" on $p2list$. One candidate may be declared elected on multiple P2-lists in the same step. So we have to define on which of these P2-lists the candidate is in fact elected. **Note 1:** $\#E_{step,4}(p2list) \leq \phi'_{step-1}(p2list)$. Especially, if $\phi'_{step-1}(p2list) = 0$ then $E_{step,4}(p2list) = \emptyset$, so no further seats will be assigned to candidates in $p2list$. **Note 2:** $E_{step,4}(p2list) \cap E_{step-1,2} = \emptyset$.
16	- Deze stap wordt voor alle tot de P3-lijst behorende P2-lijsten gelijktijdig uitgevoerd. Er kunnen na deze stap kandidaten zijn die meermalen gekozen verklaard zijn.	The set of candidates that are elected in the P3-list $p3list \in P3LISTS$ in step $step$ is given by the union of the $E_{step,4}(p2list)$: $E_{step,5} := \bigcup_{p2list \in P2LISTS_{p3list}} E_{\text{step},4}(\text{p2list})$. Note that this is not a disjoint union in general.
17	Ten aanzien van elke meervoudig gekozen kandidaat worden de P2-lijsten geselecteerd waarop de kandidaat gekozen is verklaard.	For each of these candidates $candidate \in E_{step,5}$, the set of P2-lists on which he/she is declared elected is given by $S_{step}(candidate) := \{ p2list \in P2LISTS_{p3list} \mid candidate \in E_{step,4}(p2list) \}$.

stap	*bewerking*	*Formalization*
18	De kandidaat geldt als gekozen op de P2-lijst waarop hij het hoogste aantal stemmen behaalde. Indien aantallen gelijk zijn, geldt hij als gekozen op de P2-lijst met het laagste kieskringnummer. - Het kan voorkomen dat een kandidaat het hoogste aantal stemmen heeft behaald op een P2-lijst waarop hij niet gekozen is verklaard. Die P2-lijst blijft niettemin buiten beschouwing.	Sort the elements of $S_{step}(candidate)$ in descending order by the number of vote $votes_{p2list}(candidate)$ that the candidate received on that list. In case of equal numbers of votes, sort them in ascending order by the electoral district number $idx(p2list)$. Let $\sigma_{step}(candidate) \in P2LISTS_{p3list}$ be the smallest of these element. More precisely, define on the set $S_{step}(candidate)$ the total order relation $\leq$ by $p2list \leq p2list' :\Leftrightarrow v_{p2list}(candidate) > v_{p2list'}(candidate) \vee$ $votes_{p2list}(candidate) = v_{p2list'}(candidate) \wedge idx(p2list) \leq idx(p2list')$. Now set $\sigma_{step}(candidate) := \min S_{step}(candidate)$.
19		So, $candidate \in E_{step,5}$ is elected in step *step* on the P2-list $\sigma_{step}(candidate) \in P2LISTS_{p3list}$. For a given P2-list, the set of candidates elected in step *step* is given by $E_{step,6}(p2list) := \{ candidate \in E_{step,4}(p2list) \mid \sigma_{step}(candidate) = p2list \}$. And the set of candidates elected in the steps up to *step* is given by $E_{step}(p2list) := E_{step-1}(p2list) \cup E_{step,6}(p2list)$. The number of free seats after this step is given by $\phi_{step}(p2list) := \phi'_{step-1}(p2list) - \#E_{step,6}(p2list)$.
20		Again, let $E_{\mathrm{step},2} := \cup_{\mathrm{p2list} \in P2LISTS_{p3list}} \mathrm{E_{step}(p2list)}$.

stap	*bewerking*	*Formalization*
21	Zolang er nog lijsten zijn waarop zetels te verdelen zijn, worden de stappen 3 tot en met 20 herhaald. - Nadat stap 19 is toegepast, zullen op de overige lijsten waarop in stap 19 bedoelde kandidaten gekozen waren verklaard, andere kandidaten aangewezen moeten worden om de aan de lijst toebedeelde zetels te bezetten. Bij die aanwijzing kunnen ook weer kandidaten meervoudig gekozen worden verklaard. Dit betekent dat de stappen 3 tot en met 20 telkens herhaald moeten worden, totdat voor elke aan de lijstengroep toegekende zetel vastgesteld is door welke kandidaat die zetel wordt bezet.	The above steps are repeated until in step 16 the set $E_{step,5}$ is empty. If this is the case set $step_{max}$. := $step$ - 1. So for each P2-list $p2list \in P2LISTS_{p3list}$, $E_{\mathrm{step}_{\max}}$ is the set of candidates that are elected on $p2list$. The order in which they are elected is given by the sets $E_{\mathrm{step}_{\max}}$. Denote this set of elected candidates on a P2-list $p2list \in P2LISTS_{p3list}$ by $EC_2(p2list) := E_{\mathrm{step}_{\max}}$. Also denote the set of all elected candidates on a P3-list $p3list \in P3LISTS$ by $EC_3(p3list) := E_{\mathrm{step}_{\max}}$.
22		**Note 1:** $\phi_{step}(p2list) \geq 0$ because $\#E_{step,4}(p2list) = \phi'_{step\text{-}1}(p2list)$ and $E_{step,6}(p2list) \subset E_{step,4}(p2list)$. **Note 2:** $E_{step,5} = \bigcup_{p2list \in P2LISTS_{p3list}} E_{\mathrm{step},6}(p2list)$. This union is disjoint, because $candidate \in E_{step,6}(p2list) \cap E_{step,6}(p2list')$ implies $p2list = \sigma_{step}(candidate) = p2list'$. **Note 3:** The union $E_{step}(p2list) = E_{step\text{-}1}(p2list) \cup E_{step,6}(p2list)$ is also disjoint, because $E_{step,6}(p2list) \subset E_{step,4}(p2list) \subset E_{step,3}(p2list) = C_{p2list} \setminus E_{step\text{-}1,2} \subset C_{p2list} \setminus E_{step\text{-}1}(p2list)$. Thus $\#E_{step}(p2list) = \#E_{step\text{-}1}(p2list) + \#E_{step,6}(p2list)$.
	Implementation notes:	This calculation is performed in `P2ListForRemainingSeatsDeterminator. calculate()`.

3.5.4 CONVERSION OF A TUPLE OF SETS OF P2-LISTS TO A TUPLE OF P2-LISTS

In this section we describe how we convert the roll forward sequence or the roll backward sequence to a tuple of P2-lists. The conversion may require drawing lots. From the resulting tuple in our algorithm we may need only the first element or a small number of the first elements. The conversion is described in such a way that drawing lots is only performed as much as needed to determine the needed elements. The unneeded elements following later may remain undefined.

stap	*bewerking*	*Formalization*
1		Let P be a finite set and $T = (T_1, ..., T_n)$ be a tuple of subsets of P.
2		For $i = 1, ..., n$ define $f(i) := \sum_{j=1}^{i} \#T_j$. So $m := f(n)$ will be the size of the resulting tuple $t = (t_1, ..., t_m)$ of elements of P.
3		Also let $f(0) := 0$. This defines a function $f : \{ 0, ..., n \} \to \{ 0, ..., m \}$.
4		Obviously this function is monotonically increasing, i.e. if $i < j$, then always $f(i) \leq f(j)$ holds. So from $f(0) := 0$ and $f(n) = m$ we conclude that for each element $i \in \{ 1, ..., m \}$ there is a unique index j such that $f(j\text{-}1) < i \leq f(j)$. So we define a function $g : \{ 1, ..., m \} \to \{ 1, ..., n \}$ $g(i) := \min \{ j \in \{ 1, ..., n \} \mid f(j\text{-}1) < i \leq f(j) \}$. We use the min function only to select the unique element of the set and may have used "max" just as well.
5		Now we will define the tuple $t = (t_1, ..., t_m)$ in such a way that $t_i \in T_{g(i)}$ holds for all element $i \in \{ 1, ..., m \}$: First note that $T_{g(i)} \neq \emptyset$, because $f(g(i) - 1) < i \leq f(g(i))$ implies $\#T_{g(i)} = f(g(i)) - f(g(i) - 1) > 0$.

stap	*bewerking*	*Formalization*
6		If $\#T_{g(i)} = 1$, t_i is defined to be the one element of $T_{g(i)}$. If $\#T_{g(i)} > 1$, the decision has to made by lot as follows: For now we set $a = f(g(i) - 1) + 1$ and $b = f(g(i))$, so $a \leq i \leq b$. First we draw the element t_a from $T_{g(i)}$ by lot. Next determine t_{a+1} from $T_{g(i)} \setminus \{ t_a \}$. If this set has more than one element, this is decided by lot. Then, if $\#T_{g(i)} > 2$, determine t_{a+2} from $T_{g(i)} \setminus \{ t_a, t_{a+1} \}$. Again, if this set has more than one element, this is decided by lot. This is continued until t_i is determined.
	Implementation notes:	This calculation is performed in the class `PartlyRamdomIterator`.

3.5.5 ORDER OF CANDIDATES ON THE CANDIDATE LISTS

Regulations by law: article P 19 and U 15 sub 3 electoral law.

Met het oog op het vervullen van vacatures dient de volgorde van de kandidaten op een lijst gewijzigd te worden, indien dat als gevolg van het uitbrengen van voorkeurstemmen noodzakelijk is.

stap	*bewerking*	*Formalization*
1	De P3-lijsten worden geselecteerd waarop kandidaten in totaal een aantal stemmen hebben behaald, hoger dan 10% / 25% / 50% van de kiesdeler.	The following steps are executed for each P3-list $p3list \in P3LISTS$. Recall that P_{p3list} is the set of candidates that receive a priority seat.

stap	*bewerking*	*Formalization*
2	Een nieuwe rangschikking vindt plaats van elke P2-lijst, waarop kandidaten, bedoeld in stap 1, voorkomen.	For each P2-list $p2list \in P2LISTS_{p3list}$ we determine the order of candidates as follows: The new order on the list is given by four criteria. Some of these criteria were already defined in section 3.6.2, especially in step 5 and 8. There you also find the definition of the set B_{p3list} of candidates with the total amount of votes that is higher than (for EK elections: higher than or equal to) the preferential barrier.
3	Bovenaan de P2-lijst komen te staan de kandidaten die met voorkeurstemmen zijn gekozen, in de volgorde waarin aan hen een zetel is toegewezen.	For the candidates $candidate \in P_{p3list}$ the order on the list is given by the order in which the seats were assigned to them. This order is given by two criteria: the number of votes $votes_{p3list}(candidate)$ that the candidates received on the whole P3-list $p3list$ and in case of drawing lots, the function o_2 that was defined in section 3.6.2 step 5. Since the number of votes on the P3-list is only relevant for the candidates from P_{p3list}, we define the function $o_1: C_{p2list} \to \mathbf{N_0}$ by $o_1(candidate) := votes_{p3list}(candidate)$ for all $candidate \in P_{p3list}$ and $o_1(candidate) := 0$ for all $candidate \notin P_{p3list}$

stap	*bewerking*	*Formalization*
4	Daarna volgen de kandidaten die een aantal stemmen hebben behaald, hoger dan 10% / 25% / 50% van de kiesdeler (for EK elections: greater that or equal to 100% of the electoral quota), maar aan wie geen zetel is toegewezen, in de volgorde van de aantallen behaalde stemmen.	The third criterion is the number $votes_{p2list}(candidate)$ of votes in the P2-list $p2list$ for the candidates $candidate \in C_{p2list} \cap B_{p3list} \setminus P_{p3list}$. For this criterion we define on C_{p2list} the function $o_3: C_{p2list} \to \mathbf{N_0}$ by $o_3(candidate) := votes_{p2list}(candidate)$ if $candidate \in B_{p3list} \setminus P_{p3list}$ and $o_3(candidate) := 0$ otherwise.
5	Daarna volgen de overige op de P2-lijst voorkomende kandidaten, in de volgorde van de P2-lijst.	The fourth criterion is the position $p_{p2list}(candidate)$ of a candidate $candidate \in C_{p2list}$ on the P2-list $p2list \in P2LISTS_{p3list}$.
6		To combine the three criteria we define on C_{p2list} a total order relation $c \leq c' :\Leftrightarrow o_1(c) > o_1(c') \vee$ $o_1(c) = o_1(c') \wedge o_2(c) > o_2(c') \vee$ $o_1(c) = o_1(c') \wedge o_2(c) = o_2(c') \wedge o_3(c) > o_3(c') \vee$ $o_1(c) = o_1(c') \wedge o_2(c) = o_2(c') \wedge o_3(c) = o_3(c') \wedge p_{p2list}(c) \leq p_{p2list}(c')$. Using this total order we can sort the elements of C_{p2list} to receive the new ordering: $o_{C_{p2list},\leq}(1) \leq o_{C_{p2list},\leq}(2) \leq \cdots \leq o_{C_{p2list},\leq}(\#C_{p2list})$.
	Implementation notes:	This calculation is performed in the method `ElectionResultForCandidatesDeterminator.newOrderForP2List()`.

3.5.6 CANDIDATES ELECTED ON MULTIPLE P3-LISTS

Regulations by law: none.

In this section we deal with the case that a candidate is elected in more than one P3-list.

stap	*bewerking*	*Formalization*
1	In case a candidate is elected on more than one P3-lists after the application of the above steps, it must be determined on which of these P3-lists the candidate will receive his seat. On the other P3-lists, the seat passes to another candidates.	Recall that the candidates elected on P3-list $p3list \in P3LISTS$ are given by $EC_3(p3list)$.
2	Determine the set of candidates that are elected on more than one P3-lists.	For each candidate $candidate \in CANDIDATES$ let $G_7(candidate) := \{ p3list \in P3LISTS \mid candidate \in EC_3(p3list) \}$ be the set of P3-lists on which the candidate c is elected. So we define the set of candidates that are elected on more than one P3-lists by $E_8 := \{ candidate \in CANDIDATES \mid \#G_7(candidate) > 1 \}$.
3	If there are no such candidates, there is no need to modify the elected candidates as determined in section 3.6.3. The result of section 3.6.3 is the final result. In this case the calculation terminates here.	If $E_8 = \varnothing$, there is nothing to do in this section. We can define the final seats distribution by $EC_{2,\,final}(p2list) := EC_2(p2list)$ for all $p2list \in P2LISTS$ and $EC_{3,\,final}(p3list) := EC_3(p3list)$ for all $p3list \in P3LISTS$.

4	If there are such candidates, sort them to receive a deterministic order.	In order to obtain a deterministic order on these candidates, sort them by the following three criteria: • Smallest electoral district number of a P2-list on which they are nominated. • Position of the candidate on this P2-list with the smallest electoral district number on which he/she is nominated. • List number of this P2-list with the smallest electoral district number. I.e. on E_8 we define functions $o_1: E_8 \to \mathbf{N}$, $o_1(candidate) := \min \{ idx(p2list) \mid candidate \in C_{p2list} \}$ and $o_2: E_8 \to \mathbf{N}$, $o_2(candidate) := \min \{ p_{p2list}(c) \mid candidate \in C_{p2list} \wedge idx(p2list) = o_1(candidate) \}$. Let $l(p2list)$ be the list number of a P2-list $p2list \in P2LISTS$. Then define $o_3: E_8 \to \mathbf{N}$, $o_3(candidate) := \min \{ l(p2list) \mid candidate \in C_{p2list} \wedge idx(p2list) = o_1(candidate) \}$. So we can define on E_8 a total order relation $c \leq c' :\Leftrightarrow o_1(c) < o_1(c') \vee$ $o_1(c) = o_1(c') \wedge o_2(c) < o_2(c')$ $o_1(c) = o_1(c') \wedge o_2(c) = o_2(c') \wedge o_3(c) < o_3(c')$ This way we can write $E_8 = \{ e_{8,1}, e_{8,2}, \dots \}$ where $e_{8,i} := o_{E_8,\leq}(i)$ for all $i = 1, 2, \dots, \#E_8$.

stap	*bewerking*	*Formalization*
5	For each of these candidates, perform the steps 6 to 12.	If $E_8 \neq \emptyset$, the following steps are executed for i = 1, 2, ... , $\#E_8$. Define $EC_{3,0}(p3list) := EC_3(p3list)$ and define $EC_{3,i}(p3list)$ recursively below.
6	Determine the P3-list where the candidate received the highest number of votes. In case the numbers of votes are equal, select the P2-list with the smallest electoral district number. On this list the candidate remains elected. On all other lists on which he / she is elected, the seat passes to another candidate.	For $e_{8,i}$ determine $G_7(e_{8,i})$. From $G_7(e_{8,i})$ determine the element $p3list_{9,i} \in G_7(e_{8,i})$ for which the number of votes $votes_{p3list}(e_{8,i})$ is the largest. If the highest number of votes is reached in more than one P3-list from $G_7(e_{8,i})$, select one on which the candidate is elected on a P2-list with the smallest electoral district number. More precisely, define on $G_7(e_{8,i}) \subset P3LISTS$ a total order relation by $p3list \leq p3list' :\Leftrightarrow v_{p3list}(e_{8,i}) > v_{p3list'}(e_{8,i}) \vee$ $votes_{p3list}(e_{8,i}) = v_{p3list'}(e_{8,i}) \wedge o_2(p3list) < o_2(p3list')$, where o_2 is defined by o_2: $G_7(e_{8,i}) \rightarrow \mathbf{N}$, $o_2(p3list) := \min \{ idx(p2list) \mid p2list \in p3list \wedge candidate \in EC_2(p2list) \}$. Now let $p3list_{9,i} := \min G_7(e_{8,i})$.
7		For each of the remaining P3-lists $G_7(e_{8,i}) \setminus \{ p3list_{9,i} \}$ determine the set of P2-lists $S_{10,i}$ on which the candidate $e_{8,i}$ is elected.
8	Sort the other P2-lists by electoral district number.	Sort $S_{10,i}$ by electoral district number in ascending order. This way we can write $S_{10,i} = \{ s_{10,i,1}, s_{10,i,2}, \ldots \}$.
9	For each of them perform the steps 10 to 12.	The following steps are executed for j = 1, 2, ... , $\#S_{10,i}$. Define $EC_{3,i,0}(p3list) := EC_{3,i-1}(p3list)$ and define $EC_{3,i,j}(p3list)$ recursively below.

stap	*bewerking*	*Formalization*
10	Determine the "successor" from the new order of the P2-list. The successor is the candidate on the new order with the highest position that is not yet elected. If all candidates on the P2-list are already elected, the seat will not be passed to another candidate but will remain unassigned. The software will warn the user about this remaining unassigned seat.	Let { c_1, c_2, ... , c_j } be the candidates of the P2-list $s_{10,i,j}$ in the new order determined in section 3.6.5. From these candidates determine the candidate $c_{11,i,j}$ that receives the seat in place of candidate $e_{8,i}$ on P2-list $s_{10,i,j}$. This "successor" candidate $c_{11,i,j}$ is the first of the candidates c_1, c_2, ... , c_j that is not elected and that has not been declared elected as a "successor" previously. More precisely, let $I_{11,i,j} := \{ k = 1, ..., j \mid \forall\ p3list \in P3LISTS : c_k \notin EC_{3,i,j-1}(p3list) \}$ be the indices of the not yet elected candidates of the P2-list $s_{10,i,j}$. If $I_{11,i,j} = \varnothing$, no "successor" is determined by the algorithm. If $I_{11,i,j} \neq \varnothing$, let $\alpha := \min I_{11,i,j}$ and define the "successor" by $c_{11,i,j} := c_\alpha$.
11		So the set of elected candidates of the P3-list $p3list_{9,i} \in P3LISTS$ needs to be modified to $EC_{3,i,j}(p3list_{9,i}) := (EC_{3,i,j-1}(p3list_{9,i}) \setminus \{ e_{8,i} \}) \cup \{ c_{11,i,j} \}$ or $EC_{3,i,j}(p3list_{9,i}) := EC_{3,i,j-1}(p3list_{9,i}) \setminus \{ e_{8,i} \}$ in case $I_{11,i,j} \neq \varnothing$, i.e. if no "successor" was determined, while for all other $p3list_{9,i} \neq p3list \in P3LISTS$ $EC_{3,i,j}(p3list) := EC_{3,i,j-1}(p3list)$.

stap	*bewerking*	*Formalization*
12	The seat is passed to the "successor" from the candidate that had received more than one seat.	Also the set of elected candidates of the P2-list $s_{10,i,j} \in P3LISTS$ needs to be modified. These are given by $EC_{2,i,j}(s_{10,i,j}) := (EC_{2,i,j-1}(s_{10,i,j}) \setminus \{ e_{8,i} \}) \cup \{ c_{11,i,j} \}$ or $EC_{2,i,j}(s_{10,i,j}) := EC_{2,i,j-1}(s_{10,i,j}) \setminus \{ e_{8,i} \}$ in case $I_{11,i,j} \neq \emptyset$, i.e. if no "successor" was determined, while for all other $s_{10,i,j} \neq p2list \in P2LISTS$ $EC_{2,i,j}(p2list) := EC_{2,i,j-1}(p2list)$.
13		After iterating over the values for i and each time over the values of j, we have finally determined the elected candidates for P2-lists and P3-lists: Let for now be $i := \#E_8$ and $j := \#S_{10,i}$. Then $EC_{2,final}(p2list) := EC_{2,i,j}(p2list)$ for all $p2list \in P2LISTS$ and $EC_{3,final}(p3list) := EC_{3,i,j}(p3list)$ for all $p3list \in P3LISTS$.
	Implementation notes:	This calculation is performed in the class `MultipleElectedCandidateHandler`. For each element of E_8 a `MultipleElectedCandidate` is created. The order defined in step 4 is implemented in `MultipleElectedCandidate.compareTo()`. The calculation of $p3list_{9,i}$ in step 6 is implemented in `MultipleElectedCandidate.getP3ListWithMostVotes()`.

4 ELECTION TO THE SENATE, PROVINCIAL STATES CONSISTING OF MORE THAN ONE ELECTORAL DISTRICT, PROVINCIAL STATES CONSISTING OF ONE ELECTORAL DISTRICT, WATER AND MUNICIPAL COUNCILS WITH 19 OR MORE SEATS

4.1 CHARACTERISTICS OF THE ELECTIONS

4.1.1 CHARACTERISTICS OF THE ELECTION TO THE SENATE

These are the characteristics of the elections to the senate:

Stap	*Bewerking*	*Formalization*
1	There are 12 provinces and 3 public bodies (the BES islands) for these elections. The provinces and public bodies are treated as electoral districts and are refered to as electoral districts throughout this document.	*#DISTRICTS* > 1
2	Nomination of candidate lists is carried out centrally.	
3	Besides the number of electoral districts, there are no further restrictions concerning the number of candidate lists that form a P2-list or the number of P2-lists that form a P3-list.	
4		
	Implementation notes:	The number of electoral districts is not reflected in the source code.

4.1.2 CHARACTERISTICS OF THE ELECTION TO PROVINCIAL STATES CONSISTING OF MORE THAN ONE ELECTORAL DISTRICT

These are the characteristics of the elections of provincial states consisting of more than one electoral district:

Stap	*Bewerking*	*Formalization*
1	There are two or more electoral districts for these elections.	*#DISTRICTS* > 1

2	Nomination of candidate lists may be carried out both centrally or locally.	
3	Besides the number of electoral districts, there are no further restrictions concerning the number of candidate lists that form a P2-list or the number of P2-lists that form a P3-list.	
	Implementation notes:	The number of electoral districts is not reflected in the source code.

4.1.3 CHARACTERISTICS OF THE ELECTION TO PROVINCIAL STATES CONSISTING OF ONE ELECTORAL DISTRICT, WATER AND MUNICIPAL COUNCILS WITH 19 OR MORE SEATS

These are the characteristics of the elections to provincial states consisting of one electoral district, water and municipal councils with 19 or more seats:

Stap	*Bewerking*	*Formalization*
1	There is one electoral district for these elections.	$\#DISTRICTS = 1$
2	There is only one HSB. Parties can only nominate one candidate list. There are no list groups and no sets of identical lists.	
3	Each P2-list consists of one and only one candidate list. Each P3-list is identical to a P2-list.	

4.2 CALCULATIONS PRIOR TO ASSIGNMENT OF SEATS

4.2.1 DETERMINATION OF TOTAL NUMBER OF VOTES AND CALCULATION OF THE ELECTORAL QUOTA

See EP and TK election in section 3.2.1.

4.2.2 DETERMINATION OF THE NUMBER OF CANDIDATES

See EP and TK election in section 3.2.2.

4.2.3 DETERMINATION OF THE TOTAL NUMBER OF VOTES

Legal regulation: None (formerly P4, U 16)

Everything in this section is performed in the same way as described for the EP and TK elections in section 3.2.3.

4.3 ASSIGNMENT OF SEATS TO P3-LISTS

Regulations by law: articles P 6, P 7, P 9, P 10, U 8, U 9 and U 10 of the electoral law.

stap	*bewerking*	*Formalization*
1	The seat distribution to P3-lists for these kinds of elections is performed as described in chapter 2 with the following characteristics:	The assignment of seats to P3-lists for these kinds of elections follows the common description of chapter 2. The set of lists P referred to in chapter 2 to which the seats are assigned, is the set $P3LISTS$ of P3-lists in this case. This means the seat distribution to P3-lists is a function $seatstolist := A(seats, votes, noofcandidatsinlist, param) \in \mathbf{Z}^{P3LISTS}$
2		where $seats = n \in \mathbf{N}$ is the total number of seats to be allocated in the election, $votes \equiv votes_4 \in \mathbf{N_0}^{P3LISTS}$ is the function for the number of votes of the P3-lists, $noofcandidatsinlist \equiv noofcandidatsinlist_4 \in \mathbf{N_0}^{P3LISTS}$ is the function for the number of candidates of the P3-lists and $param = (param_2, ..., param_7) = (\text{false, false, false, false, true}) \in \mathbf{B} \times \mathbf{Q} \times \mathbf{B^4}$.

stap	*bewerking*	*Formalization*
3	All P3-lists take part in the residual seat distribution. The method of largest remainder is not applied. Residual seats are assigned by the method of largest average without restriction to one seat per list. The absolute majority regulation must be considered except for the EK election.	Note that the boolean parameters are selected for the following reasons: • $param_2$ = false, because all P3-lists may take part in the residual seat distribution. • $param_3$ = 0% and $param_4$ = false, because the method of largest remainder is not applied. • $param_5$ = false, because the method of largest average is not restricted to one seat. • $param_6$ = true / false, because the absolute majority regulation must be considered except for the EK election. • $param_7$ = false, because the cosalg-mode is not needed.
4		So $seatstolist(p3list) \in \mathbf{Z}$ is the number of seats assigned to the P3-list $p3list \in P3LISTS$.
	Implementation notes:	This seat distribution is performed in the method `ElectionResultDeterminator.assignSeatsToP3Lists()`.

4.4 ASSIGNMENT OF SEATS TO P2-LISTS (WITHIN P3-LISTS)

For elections to the senate, see TK election in section 3.4.

For elections to provincial states consisting of more than one electoral district, see TK election in section 3.5.

For provincial states consisting of one electoral district and municipal councils with 19 or more seats, see EP election in section 3.6.

4.5 NOMINATION OF ELECTED CANDIDATES

For elections to the senate, see TK election in section 3.5.

For elections to provincial states consisting of more than one electoral district, see TK election in section 3.5.

For provincial states consisting of one electoral district and municipal councils with 19 or more seats, see EP election in section 3.5.

Also note the preferential barrier that varies by election type as described in section 3.5.2.

5 ELECTION OF WATER AND MUNICIPAL COUNCILS CONSISTING OF LESS THAN 19 SEATS, ISLAND COUNCILS, BOROUGH COUNCILS IN AMSTERDAM AND ROTTERDAM

5.1 CHARACTERISTICS OF THE ELECTIONS

These are the characteristics of the elections of municipal councils consisting of less than 19 seats, island councils, borough councils in Amsterdam and Rotterdam:

Stap	*Bewerking*	*Formalization*
1	There is one electoral district for these elections.	$\#DISTRICTS = 1$
2	As a consequence, there is only one HSB. Parties can only nominate one candidate list. There are no list groups and no sets of identical lists.	
3	As another consequence, each P2-list consists of one and only one candidate list. Each P3-list is identical to a P2-list.	
4	For borough councils in Amsterdam in article P8 subsection 2, “75%” is replaced by “50%”.	$param_3 = 50\%$ for BC elections for the assignment of seats to P3-lists.

5.2 CALCULATIONS PRIOR TO ASSIGNMENT OF SEATS

5.2.1 DETERMINATION OF TOTAL NUMBER OF VOTES AND CALCULATION OF THE ELECTORAL QUOTA

See EP and TK election in section 3.2.1.

5.2.2 DETERMINATION OF THE NUMBER OF CANDIDATES

See EP and TK election in section 3.2.2.

5.2.3 DETERMINATION OF THE TOTAL NUMBER OF VOTES

Legal regulation: None (formerly article P 4 electoral law).

Everything in this section is performed in the same way as described for the EP and TK elections in section 3.2.3.

5.3 ASSIGNMENT OF SEATS TO P3-LISTS

Regulations by law: articles P 6, P 8, P 9 and P 10 of the electoral law.

stap	*bewerking*	*Formalization*
1	The seat distribution to P3-lists for the municipal councils consisting of less than 19 seats is performed as described in chapter 2 with the following characteristics:	The assignment of seats to P3-lists for elections of municipal councils consisting of less than 19 seats follows the common description of chapter 2. The set of lists P referred to in chapter 2 to which the seats are assigned, is the set $P3LISTS$ of P3-lists in this case. This means the seat distribution to P3-lists is a function $seatstolist := A(seats, votes, noofcandidatsinlist, param) \in \mathbf{Z}^{P3LISTS}$
2		where $seats = n \in \mathbf{N}$ is the total number of seats to be allocated in the election, $votes \equiv votes_4 \in \mathbf{N_0}^{P3LISTS}$ is the function for the number of votes of the P3-lists, $noofcandidatsinlist \equiv noofcandidatsinlist_4 \in \mathbf{N_0}^{P3LISTS}$ is the function for the number of candidates of the P3-lists and $param = (param_2, ..., param_7)$ = (false, 75%, true, true, true, false) for AB1, GR1, ER and GC elections, $param = (param_2, ..., param_7)$ = (false, 25%, true, true, true, false) for BC elections,

stap	*bewerking*	*Formalization*
3	All P3-lists take part in the residual seat distribution. The method of largest remainder is applied prior to the largest average. In the method of largest remainder, only P3-lists are considered that have reached at least 75% of the electoral quota. The method of largest average is restriction to one seat per P3-list first. The absolute majority regulation must be considered.	Note that the boolean parameters are selected for the following reasons: • $param_2$ = false, because all P3-lists $p3list \in P3LISTS$ take part in the residual seat distribution. • $param_4$ = true, because the method of largest remainder *is* applied. • $param_3$ = 75%, because only P3-lists with at least 75% of electoral quota take part in residual seat distribution by largest remainder for AB1, GR1, ER and GC elections. • $param_3$ = 25%, because only P3-lists with at least 25% of electoral quota take part in residual seat distribution by largest remainder for BC elections. • $param_5$ = true, because the method of largest average is restricted to one seat per P3-list first. • $param_6$ = true, because the absolute majority regulation *must* be considered. • $param_7$ = false, because the cosalg-mode is not needed.
4		So $seatstolist(p3list) \in \mathbf{Z}$ is the number of seats assigned to the P3-list $p3list \in P3LISTS$.
	Implementation notes:	The seat distribution is configured with the parameters created in the static methods `GsdaParameters.forP42DistributionGr1()` and `GsdaParameters.forP42DistributionBC()`.

5.4 ASSIGNMENT OF SEATS TO P2-LISTS (WITHIN P3-LISTS)

See EP election in section 3.4. Note that the distribution is always trivial.

5.5 NOMINATION OF ELECTED CANDIDATES

See EP election in section 3.5. Also note the preferential barrier that varies by election type as described in section 3.5.2.

6 DRAWING LOTS

6.1 HOW DRAWING LOTS IS PERFORMED

The algorithm of determining the result of the election requires drawing lots in various cases. In this section we define how drawing lots is performed.

Whenever drawing lots is required, the responsible authority must draw one or more elements from a given base set. The base set may be a set of P3-lists, P2-lists, lists or candidates. The number of elements to be drawn is always smaller than the number n of elements in the base set. So we have

$$1 \leq k < n.$$

Because it is necessary to keep track of the order in which the elements are drawn, the drawing of lots is split into k single drawing events. In each of these events only one element is drawn. So in the first event the responsible authority draws one element from a base set of n elements. If $k > 1$, then in the next drawing event the responsible authority draws one element from the remaining n - 1 elements etc..

6.2 WHEN DRAWING LOTS IS PERFORMED

In this section we summarize all the different cases where the electoral law requires the responsible authority to draw lots.

Note: For the EK election in some cases the same regulation is in another section of the electoral law.

Event	Article in electoral law	Elements	Base set	Consequence
Equal averages in assignment of residual seats to P3-lists by largest average.	P 7 EK: U 9	P3-lists	P3-lists with equal largest average	The drawn P3-list is assigned a residual seat.
Equal remainders in assignment of residual seats to P3-lists by largest remainder.	P 8 sub. 1 EK: n/a	P3-lists	P3-lists with equal largest remainder	The drawn P3-list is assigned a residual seat.
Modification of the seat distribution, if a P3-list attained the absolute majority of all votes but not the absolute majority of seats. More than one P3-list obtained a residual seat for an equally low average or an equally low remainder.	P 9 EK: n/a	P3-lists	P3-lists that obtained a residual seat for the lowest and equally low average or the lowest	The drawn P3-list looses the residual seat that is assigned to another P3-list which has the absolute majority of votes.

Event	Article in electoral law	Elements	Base set	Consequence
			and equally low remainder.	
Equal remainder in assignment of residual seats to P2-lists by largest remainder.	P 12 sub. 5 EK: U 12 sub. 5	P2-lists	P2-lists with equal largest remainder	The drawn P2-list is assigned a seat.
Equal averages in assignment of residual seats to P2-lists by largest average.	P13 sub. 2 (P 7) EK: U 13 sub. 2 (U 9)	P2-lists	P2-lists with equal largest average	The drawn P2-list is assigned a seat.
Candidates with more votes than the preferential barrier have equal number of votes.	P 15 sub 1	Candidates	Candidates with highest and equal number of votes, which is above the preferential barrier	The drawn candidate receives a seat.
Sufficient seats have not been awarded to any of the P2-lists on which an elected candidate appears. He is awarded a seat in respect of the P2-list on which he has obtained the highest number of votes. A seat which was last awarded in accordance with sections P 12 or P 13 to one of the P2-lists of the P3-list shall therefore be withheld. The last awarded seats were awarded simultaneously due to equal largest remainder or equal largest average.	P 16 sub 2	P2-lists	P2-lists that were awarded the last seats simultaneously.	A seat from the drawn list is withheld.
"Late list exhaustion": When assigning seats to candidates, a P2-list still has unassigned seats but no unelected candidates. The seat shall	(P 18)	P2-lists	P2-lists with equal largest remainder or equal largest average	The drawn list receives a seat from the list that has no more unelected candidates.

Event	Article in electoral law	Elements	Base set	Consequence
pass to another P2-list of the P3-list by continuing the residual seat assignment within that P3-list. In the continued residual seat assignment, equal largest remainders or equal largest averages occur.				

7 TECHNICAL NOTES

7.1 NUMBER REPRESENTATION

In the above specification, numerous calculations are performed. These calculations will be implemented in module U of the OSV2020. We use the following number representations in the implementation:

7.1.1 ROUNDING

In the whole algorithm, no rounding is performed. All numbers are represented with full precision.

7.1.2 VOTES AND SEATS

For all variables that represent numbers of votes or numbers of seats, we use the Java primitive type "long". This type allows to represent each integer value from -2^{63} to $2^{63}-1$.

We assume that numbers of votes do not exceed $2^{30} > 1.000.000.000$ and numbers of seats do not exceed $2^{15} > 32.000$. So multiplying "votes times votes" or "votes times seats times seats" can be represented as a long value without loss of precision.

7.1.3 POSITION IN LIST, NUMBERS OF LISTS OR CANDIDATES

Numbers as the position in a candidate list, the number of candidates or number of lists are insome cases represented using the Java primitive type "integer". This way values from -2^{31} to $2^{31}-1$ can be represented without loss of precision.

7.1.4 FRACTIONS

Fractions are always represented in the class `de.ivu.wahl.result.Fraction` as pair of Java "long" values. Comparing two fractions a/b and c/d will be performed by "cross multiplying", i.e. comparing $a \cdot d$ and $b \cdot c$.

7.2 IMPLEMENTATION PRINCIPLES

In this section we describe some of the programming principles used in the implementation of the specified algorithm. The aim of these principles are:

- to obtain maximum in clarity, simplicity and readability in the source code,
- to avoid programming errors before they occur.

7.2.1 USE IMMUTABLE OBJECTS WHENEVER POSSIBLE

In software engineering we discriminate between mutable and immutable objects. Immutable objects are also said to have a value semantics, i.e. they represent a value that never changes, while mutable objects have a state that may change in time.

It is good practice to use immutable objects whenever entities are represented whose (for the software) relevant properties do not change. This avoids errors by accidentally changing them. Attributes of immutable objects are declared “final” to ensure they cannot be changed accidentally. Attributes of immutable objects that are collections are additionally immutable copies of the collections used to create the object (see below), so they cannot be changed accidentally, either.

In the algorithm we have to make a lot of use of collections, i.e. lists or sets of objects. Java has a standard practice to create collections that are immutable to the receiver. This is done

using Collections.unmodifyable...(). To make sure that also the provider of the collection cannot modify if after it was provided to a receiver, we create a copy of the original collection and make it unmodifyable. We make high use of this and implemented some utility methods in the class de.ivu.wahl.result.Util.

7.2.2 DISCRIMINATION BETWEEN INPUT AND OUTPUT DATA

The input data to the algorithm is of course immutable. It is implemented in the class ElectionAndCandidatesAndVotesImpl and interface ElectionAndCandidatesAndVotes. So all input data is contained in a single object.

This object is built in three steps using three builders. The ElectionBuilder creates an Election object that contains information of electoral districts, total number of seats to be assigned and preferential barrier. The CandidatesBuilder creates an ElectionAndCandidates object by adding to the Election all required information about candidates, candidate lists and combined lists. Finally the VotesBuilder creates an ElectionAndCandidatesAndVotes object by adding to the ElectionAndCandidates the information about the number of votes for each candidate in each electoral district.

Specification	**Implementation interface** **Implementation class**
Election	de.ivu.wahl.result.Election de.ivu.wahl.result.builder.ElectionImpl (immutable)
Election + candidate lists	de.ivu.wahl.result.ElectionAndCandidates de.ivu.wahl.result.builder.ElectionAndCandidatesImpl (immutable)
Election + candidate lists + votes	de.ivu.wahl.result.ElectionAndCandidatesAndVotes de.ivu.wahl.result.builder.ElectionAndCandidatesAndVotesImpl (immutable)

The output of the election result is implemented by providing the algorithm with an object that implements the ElectionResult interface. All results will be passed to this object and further processed (e.g. saved in a database, displayed on screen, exported, printed in protocol) outside the algorithm code.

7.2.3 NO OBJECT REUSE

All Java objects created in the algorithm are used for one purpose only and are used just as long as they are needed. For example one ElectionResultForP2ListsDeterminator is used to assign the seats within one P3-list. For each P3-list a new ElectionResultForP2ListsDeterminator is created. This prevents programming errors by un-initialized object state.

7.2.4 ANONYMOUS IDENTIFICATION OF ENTITIES USING EXTERNAL KEYS

Many entities like P2-lists, P3-lists and candidates are internally stored as values in a java.util.Map. The key may be an arbitrary external object that is used to identify the entity, for example a database key or a natural key of the entity.

The algorithm makes no further use of the external keys. The builder ensures that no key is used more than once. The external keys may be used to link the entities created for the algorithm to the data source (EML file entry or database row).

This way the algorithm is decoupled from the external representation of the data but provides a way to link internal entities and external keys.

7.3 IMPLEMENTATION OF DRAWING LOTS

The implementation of drawing lots has to handle the event that one or more decisions in the process of the algorithm have to be made by lot. The drawing of lots is not part of the software. Only the result will be entered by the user to enable the calculation to proceed.

In case of such an event, the algorithm must have a way to present the alternatives to an instance that makes the decision. We will go into more details about this instance later. Potentially the first decision or one of the following decisions may be postponed to a later point in time. In between both the client and the server software may be stopped. Decisions that have once been made shall not be asked for a second time.

Here are some conclusions that arise from these requirements:

- Once a decision has been made, it has to be stored in the database.
- The algorithm cannot really wait for a decision because the software must still react to user interaction even if a decision is postponed.
- The algorithm must have knowledge of all decisions that have already been made.
- So the algorithm can terminate in two different ways:
  - With the election result. This is the case if no drawing lots is required or if all decisions have already been made.
  - With an open decision that has to be decided by lot.

So the instance that makes the decision to the algorithm is an implementation of the DrawingLotsCallback interface, namely an instance of the class DrawingLotsCallbackImpl. This DrawingLotsCallback has the knowledge of all decisions that have already been stored in the database. So whenever the algorithm finds that a decision has to be made by lot, it calls back to the DrawingLotsCallback. It the corresponding decision has already been stored in the database, DrawingLotsCallback provides the algorithm with the decision and the algorithm can proceed. If not, a DrawingLotsException will be raised.

With the DrawingLotsException the algorithm is terminated and information about the given alternatives is handed out to the exception handler. The exception handler will present this information to the user. If the user makes the decision now, it will be stored in the database and the algorithm will be completely restarted.

We try to demonstrate this in an example:

The votes have been entered and it occurs that the drawing lots is required twice (which might not be known to anyone). The user will start the algorithm and will be asked for the first decision. Since the committee is not present, no decision will be made. Some days later for the committee meeting the software (server and client) are re-started. The user will again start the algorithm and be asked for the same decision. Now the committee draws the lot and enters the result. It will be stored in the database. This also re-starts the algorithm automatically and the user is asked for the second decision. Now someone pulls the plug from the server. The server has to be re-started, the clients has to re-login. The user will start the algorithm one more time. The first decision is persistent. So the algorithm will ask the user for the second decision again. The committee draws lots for the second decision and enters the result. It will

be stored in the database. This also re-starts the algorithm automatically. Now all decisions are stored in the database. So the algorithm will terminate with the election result.

7.4 IMPLEMENTATION REFERENCE

7.4.1 ENTITY REFERENCE

The following table shows the entities and some values that are relevant for the determination of the election result. In the second column the name of the Java interface and implementation class are given.

Specification	**Implementation interface** **Implementation class**	**Implementation details**
Electoral district	de.ivu.wahl.result.ElectoralDistrict de.ivu.wahl.result.builder.ElectoralDistrictImpl	Electoral districts are identified by their number. The can be accessed by `Election.getElectoralDistricts()` or `Election.getElectoralDistrictsByExternalKeys()`.
Candidates	de.ivu.wahl.result.Candidate de.ivu.wahl.result.builder.CandidateImpl	Candidates can be accessed by `CandidateList.getCandidates()`. A specific Candidate can be accessed by `getCandidateByExternalKey(externalKey)` if it was added to the CandidateBuilder with the same `externalKey`.
Deceased candidates		The dead candidates can be accessed by `ElectionAndCandidates.getDeceasedCandidates()`.
Candidate list	de.ivu.wahl.result.CandidateList de.ivu.wahl.result.builder.CandidateListImpl	A collection of all candidate lists can be accessed by `ElectionAndCandidates.getAllCandidateLists()`. A specific CandidateList can be accessed by `getListByExternalKey(externalKey)` if it was added to the CandidateBuilder with the same `externalKey`.
P2-lists	de.ivu.wahl.result.determination.P2List de.ivu.wahl.result.builder.P2ListImpl	A Map of all P2Lists by their `externalKey` may be accessed by `ElectionAndCandidates.getP2ListsMap()`.
P3-lists	de.ivu.wahl.result.determination.P3List de.ivu.wahl.result.builder.P3ListImpl	A Map of all P3Lists by their `externalKey` may be accessed by `ElectionAndCandidates.getP3ListsMap()`.
Votes		The votes of a candidate in an electoral district can be accessed by

Specification	Implementation interface Implementation class	Implementation details
		`ElectionAndCandidatesAndVotes.getVotes()`.
Seats		The total number of seats to assign within the election: `Election.getNumberOfSeats()`

7.4.2 STEPS OF THE ALGORITHM

The following table shows the bigger steps of the determination of the election result. The second column shows the Java class and method where this part of the calculation is performed.

Step in the algorithm	Implementation
Determination of total number of votes and calculation of the electoral quota	ElectionResultDeterminator.calculate() new VotesCounter()
Determination of the number of candidates	CandidatesCounter
Assignment of seats to P3-lists	ElectionResultDeterminator. assignSeatsToP3Lists() GeneralSeatDistributor
Conditions for the next assignment step	GeneralSeatDistributor.determineNextStepType()
Assignment of seats based on attaining the electoral quota (first assignment)	GeneralSeatDistributor.performFirstAssignment()
Assignment of residual seats	GeneralSeatDistributor. performDHondtAssignment() performDHondtAssignmentRestricted() performNiemeyerAssignment()
Continued drawing lots in assignment of residual seats	GeneralSeatDistributor. performContinuedDrawingLots()
Modification of the seat distribution, if a list attained the absolute majority of all votes	GeneralSeatDistributor.considerAbsoluteMajority()
Modifying the distribution of seats in case of list exhaustion	GeneralSeatDistributor. performExhaustedListsStep
Calculation of the electoral quota and the P3-electoral quota	GeneralSeatDistributor()
Assignment of seats to P2-lists (within P3-lists)	ElectionResultDeterminator. assignSeatsWithinP3Lists()

Step in the algorithm	Implementation
	ElectionResultForP2ListsDeterminator. calculate() GeneralSeatDistributor
Calculation of the roll back sequence	AssignmentTracer.getRollBackSequence()
Calculation of the roll forward sequence	AssignmentTracer.getRollForwardSequence()
Nomination of elected candidates	ElectionResultForCandidatesDeterminator. calculate(). The core of this is to create a list of CandidateForSorting's and call `Collections.sort(candidatesForSorting)`
Nomination of candidates elected by preferential vote	ElectionResultForCandidatesDeterminator. getCandidatesWithPrioritySeat() P2ListForPrioritySeatsDeterminator. determineP2ListsForCandidatesWithPrioritySeat() SortCandidatesUtil. COMPARATOR_FOR_PREF_SEATS PartlyRamdomIterator
Nomination of all remaining candidates	P2ListForRemainingSeatsDeterminator. calculate() SortCandidatesUtil.sortByPositionOnP2List() PartlyRamdomIterator
Order of candidates on the candidate lists	P2ListForRemainingSeatsDeterminator. newOrderForP2List() SortCandidatesUtil. NewOrderOnP2ListComparator
Candidates elected on multiple P3-lists	MultipleElectedCandidateHandler. handleCandidatesElectedOnMoreThanOnP3List() MultipleElectedCandidate. getP3ListWithMostVotes()

7.4.3 REFERERCE OF ALL IMPLEMENTATION CLASSES AND INTERFACES

The following table contains a complete list of all classes in the implementation and their role in the algorithm. <base> stands for the base package "de.ivu.wahl.result".

In the column "m/iC/I/E" the Java interfaces are marked with "I", immutable Java classes are marked with "iC", mutable Java classes are marked with "mC" and enumerations are marked with "E".

Package name	Class / Interface name	m/iC/I/E	Role
<base>	Candidate	I	Entity
<base>	CandidateList	I	Entity
<base>	Election	I	Input data

<base>	ElectionAndCandidates	I	Input data
<base>	ElectionAndCandidatesAndVotes	I	Input data
<base>	ElectionSubcategory	E	Enumeration of the different types of elections
<base>	ElectoralDistrict	I	Entity
<base>	Fraction	iC	Implementation of a mathematical fraction of whole numbers within the range of long values
<base>	NamedObject	I	Super interface implemented by all entities that are displayed in the user interface.
<base>	NumberedObject	I	Common super interface of P3List and ElectoralDistrict. Used for sorting.
<base>	Util	iC	Utility class for • Creating unmodifiable copies of collections. • Displaying collections of NamedObjects. • Displaying a fraction. No instances are created.
<base>.builder	CandidateImpl	iC	Entity
<base>.builder	CandidateListImpl	iC	Entity
<base>.builder	CandidatesBuilder	mC	Builder
<base>.builder	ElectionAndCandidatesAndVotesImpl	iC	Input data
<base>.builder	ElectionAndCandidatesImpl	iC	Input data
<base>.builder	ElectionBuilder	mC	Builder
<base>.builder	ElectionImpl	iC	Input data
<base>.builder	ElectoralDistrictImpl	iC	Entity
<base>.builder	P2ListImpl	iC	Entity
<base>.builder	P3ListImpl	iC	Entity
<base>.builder	VotesBuilder	mC	Builder
<base>.determination	AbstractElectionResultDeterminator	iC	Common superclass of ElectionResultForP3ListsDeterminator and ElectionResultForP2ListsDeterminator

<base>.determination	AssignmentTracer	mC	Helper class that stores and checks the numbers of seats assigned to lists in the assignment steps. It can also calculate the roll bach sequence and roll forward sequence.
<base>.determination	CandidateForSorting	mC	Helper class of the ElectionResultForCandidatesDeterminator.
<base>.determination	CandidatesCounter	iC	Helper class to determine the number of candidates in all kinds of lists
<base>.determination	ElectionResultDeterminator	mC	“Frame” of the whole algorithm. Some simple steps are implemented here. More complex steps are delegated to other classes.
<base>.determination	ElectionResultForCandidatesDeterminator	mC	Implementation of the section “Nomination of elected candidates”
<base>.determination	ElectionResultForP2ListsDeterminator	mC	Implementation of the section “Assignment of seats to P2-lists (within P3-lists)”
<base>.determination	GeneralList	I	Common super interface of P2List, P3List and ParentOfP3List
<base>.determination	MultipleElectedCandidate	iC	Contains all information relevant for the MultipleElectedCandidateHandler about a candidate that is elected on more than one P3-list.
<base>.determination	MultipleElectedCandidateHandler	iC	Helper class that implements the part of the algorithm that deals with candidates that are elected on more than one P3-list
<base>.determination	OrderUtil	iC	Utility class for creating sorted equivalence classes from a (not totally) ordered collection. No instances are created.
<base>.determination	P2List	I	Entity
<base>.determination	P2ListForPrioritySeatsDeterminator	mC	Determines the P2-list on which a candidate is elected that is elected by preferential vote.
<base>.determination	P2ListForRemainingSeatsDeterminator	mC	Determines the P2-list on which a candidate is elected that is elected by preferential vote.

<base>.determination	P3List	I	Entity
<base>.determination	ParentOfP3List	I	Technical interface used to create an AssignmentTracer for P3Lists.
<base>.determination	SeatDistributionInP3List	mC	Holds the state of the determination of elected candidates (which candidate is elected on which P2-list).
<base>.determination	SortCandidatesUtil	iC	Utility class for sorting lists of candidates by certain criteria. No instances are created.
<base>.determination	VotesCounter	iC	Implements the section “Determination of total number of votes”. Immutable! The calculation is performed in the constructor. Then the results of the calculations can be accessed with the getters of VotesCounter.
<base>.drawlots	Decision	iC	A Decision holds information about the event that one alternative out of a given list of alternatives was drawn by lot.
<base>.drawlots	DecisionType	E	Enumeration of six different kinds of events that are identified by the corresponding article in the electoral law.
<base>.drawlots	DrawingLotsAlternative	I	Common interface for the different entities that may be drawn by lot.
<base>.drawlots	DrawingLotsCallback	I	The algorithm asks a DrawingLotsCallback to provide a decision if drawing lots is required.
<base>.drawlots	DrawingLotsCallbackImpl	mC	Implementation of DrawingLotsCallback.
<base>.drawlots	DrawingLotsException	iC	Is thrown when a required decision cannot be provided.
<base>.drawlots	DrawingLotsIdentifier	iC	Used to compare DrawingLotsAlternatives and to identify them with objects that can be accessed by external key.
<base>.gsda	FractionFactory	I	Implementation of this interface represent the functions f_i in the residual seat assignment steps (sections 2.4.3 - 2.4.5).

<base>.gsda	FractionFromList	iC	Helper class, used to compare and order fractions that occur in the assignments of seats by greatest average or greatest remainder.
<base>.gsda	GeneralSeatDistributor	mC	Implementation of the General Seat Distribution Algorithm
<base>.gsda	GsdaParameters	iC	Configuration of the GeneralSeatDistributor.
<base>.gsda	GsdaResult	iC	Contains the results of the algorithm: seats per list, foll back sequence (optional) and roll forward sequence (optional)
<base>.gsda	PartlyRamdomIterator	mC	Implementation of the conversion of a tuple of sets of P2-lists to a tuple of P2-lists. The order of the resulting tuple is partly decided by lot. Decisions by lot are made only if required.
<base>.i18n	MessageKeys	iC	Utility class for localization.
<base>.i18n	Messages	iC	Utility class for localization.
<base>.result	Assignment	iC	Information about a single assignment of a number of seats to a single list.
<base>.result	AssignmentSupplement	I	Interface for supplementary information about an assignment that is needed for creating the protocol (model P22)
<base>.result	AssignmentSupplementVotes	iC	Implements the AssignmentSupplement interface. Supplementary information containing the number of votes of the list that receive the assignment.
<base>.result	AssignmentSupplementWithRemainderFraction	iC	Implements the AssignmentSupplement interface. Supplementary information containing the number of votes and the remainder according to Niemeyer of the list that receive the assignment.
<base>.result	AssignmentType	E	Enumeration of different kinds of assignments (first, greatest average, list exhaustion etc.).
<base>.result	CandidateResult	mC	Information about the election result of a nominated candidate on a P2-list.

			Is mutable only for modifications by the MultipleElectedCandidateHandler.
<base>.result	Distribution	E	
<base>.result	DummyElectionResult		Not used.
<base>.result	ElectionResult	I	Interface that takes all results of the election. This implementation is outside of the algorithm, because it needs knowledge of the database representation.
<base>.result	StepType	E	Enumeration representing the possible values of κ_i.

7.5 PARTS OF THE ALGORITHM THAT ARE WEAKLY BACKED UP BY THE ELECTORAL LAW

In practice, many of the exceptional events like drawing lots, exhausted lists or the absolute majority regulation that are covered by the electoral law hardly ever happen. Still there are other exceptional events that may in theory happen are not explicitly mentioned by the electoral law, mostly because they are not of any practical relevance. For some of these events it is easy to come up with an election result that provokes exactly this situation. So the software must have a way to handle them.

The aim of the present specifications is to handle all these events in such a way that only one interpretation of the result is possible. In this sections we point to those parts of the specification where the electoral law was not completely specific.

7.5.1 ALL LISTS EXHAUSTED

This case is irrelevant in practice. It is clear how to handle this case even though it is not mentioned in the electoral law: Exhausted lists loose assigned seats that cannot be allocated to candidates and do not receive any seats in the following steps. If all lists are exhausted, it is possible that some seats are not assigned to a list.

7.5.2 NO VOTES FOR ANY OF THE LISTS

This case is irrelevant in practice for final results but may occur as long as only partial results are present. In that case we set the total number of votes to 1 to prevent division by 0 (see section 2.3.1 step 1a. This may lead to an even distribution of the seats between all lists in the residual seat distribution of may lead to no list receiving any seat at all, because non of them has more votes than the electoral quota.

We may assume that the law only applies to final results. So at least this treatment does not conflict with the law.

7.5.3 CANDIDATES ELECTED FOR MORE THAN ONE P3-LIST

The treatment of candidates for more than one P3-list is not mentioned in the electoral law. We received the following statement from the Kiesraad: "The Kieswet used to say that if a candidate is chosen on one than more lists, he or she is chosen on the list on which he received most votes. Although this exact text can't be found in the text nowadays, it was not the intention to change the meaning of the law about this situation. It is therefore better to apply this in the case that the situation you explained underneath occurs. This means that in that situation

candidate C would not be nominated twice but would only be nominated on the list where he had received the most votes."

This is specified now specified in section 3.5.6.

But this statement leaves some open questions:

- What happens to the seat that is not taken by the named candidate?
- If this applies to more than one candidate, shall this be handled in any specific order?

The present specification answers these questions in section 3.5.6 steps 10 and 4 respectively. For the seat that is not taken, we define a "successor". Also in steps 10 we answer the question

- What happens there is no "successor" for the seat that is not taken by the named candidate?

For these parts of the specification it must be verified by the Kiesraad that the specification complies with the intention of the electoral law.

7.5.4 ROLLING BACK MORE THAN ONCE

Article P 16 subsection 2. is not clear about what happens if it applies more than once. Our interpretation is that the seat distribution shall be "rolled back" as far as necessary. This can be seen in the present specification in the treatment of the roll back sequence.

For these parts of the specification it must be verified by the Kiesraad that the specification complies with the intention of the electoral law.

7.5.5 ROLLING BACK THE FIRST ASSIGNMENT

The roll back sequence memorizes, which list(s) received at least one seat in earlier steps. It does not memorize, exactly how many seats these lists received. So if the first assignment must be rolled back, all lists that received at least one seat in the first assignment are treated the same, no matter how many seats they received (see section 2.3.2 step 5).

For these parts of the specification it must be verified by the Kiesraad that the specification complies with the intention of the electoral law.

7.5.6 ROLL BACK SEQUENCE EXHAUSTED

It is even possible that the roll back sequence is exhausted. In that case we withhold a seat from the P2-list that still has an unassigned seat (see section 3.5.2 step 18d).

For these parts of the specification it must be verified by the Kiesraad that the specification complies with the intention of the electoral law.

7.5.7 LATE LIST EXHAUSTION

The case that in the assignment of seats to candidates there are on a P2-list still unassigned seats but no more unelected candidates is treated using the roll forward sequence (see section 3.5. steps 4a - 13). This case was not mentioned in the electoral law but in the document "Vaststellen_van_de_uitslag_van_verkiezingen_NL.doc" that was provided by the Kiesraad.

7.5.8 ROLLING BACK AND ROLLING FORWARD

The two cases treated using the roll back sequence and the roll forward sequence may occur both for the same P3-list. One might expect that a P2-list that looses a seat by application of the roll back sequence is the first to receive a seat when the seat distribution is "continued" as

described using the roll forward sequence. The present specification does not imply this. The roll forward sequence does not "compensate" in any way for seats lost in application of the "roll back sequence".

For these parts of the specification it must be verified by the Kiesraad that the specification complies with the intention of the electoral law.

7.5.9 CONFLICTS BETWEEN THE "PREFERERED" P2-LISTS OF CANDIDATES THAT ARE SIMULTANEOUSLY ELECTED BY PREFERENTIAL VOTE

In section 3.5.2 we specify on which P2-list a candidate that is elected by preferential vote receives his or her seat. If more than one candidate is elected with the same number of vote, this may lead to more candidates elected on a P2-list that it has unassigned seats. Such "conflicts" are resolved in the steps 15a-f. The current specification of this case is the result of a discussion with the Kiesraad.

8 INDEX OF MATHEMATICAL VARIABLES AND ABBREVIATIONS

A

A	General seat distribution algorithm (GSDA)
AMV	Absolute majority of votes (Variable in GDSA)
AMS	Absolute majority of seats (Variable in GDSA)
AB1	Election to water council with less than 19 seats
AB2	Election to water council with 19 or more seats

B

$\mathbf{B} := \{ 0, 1 \}$ $= \{ \text{true, false} \}$	Set of Boolean values
$param = (param_2, ..., param_7) \in \mathbf{B} \times \mathbf{Q} \times \mathbf{B^4}$	Six parameters of the general seat distribution algorithm in section 2.2
B_{p3list}	Candidates of the P3-list *p3list* who have sufficient votes to receive a priority seat.
BC	Borough council in Amsterdam

C

$CANDIDATES$, $candidate$	Candidates, a candidate
$\tilde{C}_l$, $\tilde{C}_s$, $\tilde{C}_{p3list}$	Candidates on a candidate list, P2-list, P3-list
$cosalg_{step}$	Flag indicating if the cosalg-mode is switched on (Variable in GDSA)
$condition_0$, $condition_3$, $condition_4$, $condition_6$, $condition_7$	Conditions for the next step (Boolean functions in GDSA)
C_{list}	Candidates on a candidate list that are not deceased
C_{p2list}	Candidates on a P2-list that are not deceased
C_{p3list}	Candidates on a P3-list that are not deceased
$C'_{p3list}(x)$	Candidates that have more than x votes
$C''_{p3list}(x)$	Candidates that have at least than x votes
C_1, ..., C_h	Equivalence classes of candidates who receive a priority seat

D

D_{step}	Set of lists from which in a "continued drawing lots step" one alternative must be drawn (Variable in GDSA)
d_{step}	Maximum (largest average of largest remainder) in seat assignment in GDSA

E

ε	Kind of election

$EC_2(p2list)$	Elected candidates on a P2-list
$EC_3(p3list)$	Elected candidates on a P3-list
$EC_{2,\,final}(p2list)$	Finally elected candidates on a P2-list
$EC_{3,\,final}(p3list)$	Finally elected candidates on a P3-list
$E_{A,\leq}$	Equivalence classes of the equivalence relation $\sim_{\leq}$ on a set A
$E_{p3list} = E_{P_{p3list},\leq}$	Equivalence classes of candidates who receive a priority seat
$E_{step}(p2list)$	Set of candidates that are considered as elected on the P2-list $p2list \in P2LISTS_{p3list}$ in the steps up to *step*
$E_{step,2}$	Set of candidates that are elected in the steps up to *step*
$E_{step,3}(p2list)$	
$E_{step,4}(p2list)$	
$E_{step,5}$	
EK	Election to the Senate (First chamber, Dutch: Eerste Kamer)
EP	Election to the European Parliament
ER	Island council

F

f_{step}	Function that shall be maximized in residual seat assignment (Function in GDSA)

G

LISTGROUPS	List groups
GROUPS''	P2-lists that do not belong to a list group
P3LISTS, *p3list*	P3-lists
GC	Borough council in Rotterdam
GR1	Election to municipal councils with less than 19 seats
GR2	Election to municipal councils with 19 or more seats
noofcandidatsinlist	Function of number of candidates of a list (Parameter in GDSA)
$noofcandidatsinlist_1(list)$	Number of allocatable candidates of a candidate list.
$noofcandidatsinlist_2(p2list)$	Number of allocatable candidates of a P2-list.
$noofcandidatsinlist_3(p3list)$	Number of allocatable candidates of a P3-list.
$noofcandidatsinlist'_{p3list}(x)$	Number of allocatable candidates that have more than x votes

$noofcandidatsinlist''_{p3list}(x)$	Number of allocatable candidates that have at least than x votes
H	
H_{step}	Set of lists considered in residual seat assignment (Variable in GDSA)
$h = \#E_{p3list}$	Number of equivalence classes of candidates who receive a priority seat
I	
$idx(district)$, $idx(list)$, $idx(p2list)$	Electoral district number
$step$	Step number
J	
J_{step}	Lists that obtained a seat for the lowest average or the smallest remainder (Variable in GDSA)
K	
$steptype_{step}$	Step type (variable in GDSA)
K_1, K_2, K_3	Sets of indexes in the proof that the algorithm terminates
$electoralquota$	Electoral quota
L	
$LISTS$, $list = (district, t)$	Candidate lists, a candidate list
M	
$seatstolist_{step}(p)$	Total seat assignment up to step $step$ to list p (Function in GDSA)
$seatstolist(p)$	Total seat assignment to list p (Result of GDSA)
$seatstolist(p3list)$	Result of the assignment of seats to P3-lists
M_{step}	Lists that received a seat by the method of largest average restricted to one seat per list (Variable in GDSA)
$seatstolist^{(p3list)}(s)$	Result of the assignment of seats to P2-lists within the P3-list $p3list$
$seatstolist''(s)$	Result of the assignment of seats to P2-lists
$\min_{A,\leq}$	Smallest element of A with respect to the total order $\leq$
$\min A$	Smallest element of A with respect to a total order know from the context
N	
$\mathbf{N} = \{1, 2, 3, ...\}$	Natural numbers
$\mathbf{N_0} = \{0, 1, 2, 3, ...\}$	Non-negative Integers
n	Number of seats

O

$\varnothing$	Empty set
$o_{A,\leq}$	Function defined on A with a total preorder $\leq$.
$o_{A,\leq}^{-1}$	Inverse function
$\omega(district)$	Vote value in the electoral district *district*

P

$position_{list}(candidate)$, $position_{p2list}(candidate)$	Position on the candidate list, P2-list
PS1	Election for provincial states with one electoral district
PS2	Election for provincial states with more than one electoral district
P, p	Lists (may be P3-lists, P2-lists or candidate lists), Parameter in GDSA
$LISTSTAKINGPART_{step}$	Lists that take part in the step *step* + 1 (Variable in GDSA)
P_{AM}, p_{AM}	Lists that reached the *AMV*, the list that reached the *AMV* (Variable in GDSA)
p_{LR}	List, that received the last residual seat (Variable in GDSA)
$\mathbf{P}(A)$	Power set of A
P_{p3list}	Candidates that receive a priority seat
$\phi_{step}(p2list)$	Number of unassigned seats
$\phi_{step\text{-}1,j}(p2list)$	Number of unassigned seats

Q

$\mathbf{Q}$	Rational numbers
quota	Quota (variable in GDSA)

R

$\mathbf{R}$	Real number
rbs	Roll back sequence
rfs	Roll forward sequence
$residualseats_{step}$	Number of residual seats (variable in GDSA)
R_{step}	Candidates that receive their seat by article P16 subsection 2

S

SETSOILISTS	Sets of identical lists
SINGLELISTS	Independent P2-lists
P2LISTS, p2list	P2-lists, a P2-list

	$P2LISTS_{p3list}$ = {$p2list \in P2LISTS$ | $p2list \subset p3list$}	P2-lists of a P3-list
	seats	Number of seats (Parameter in GDSA)
	totalnumberofvotes	Total number of votes
	$\sigma_{step}(candidate)$	The P2-list on which a candidate is elected
	$(p2list_1, p2list_2, \ldots, p2list_{k_{\max}})$	Tuple of P2-lists
T		
	$t = (t_1, t_2, \ldots, t_{\#list})$	Tuple of candidates
	TK	Election to the House of Representatives
	totalnumberofvotes	Total number of votes (variable in GDSA)
U		
V		
	DEADCANDIDATES	Deceased candidates
	votes(*district, candidate*)	Votes of a candidate in an electoral district
	votes'(*district, candidate*)	Votes cast for a candidate in an electoral district
	$votes_{list}(candidate)$, $votes_{p2list}(candidate)$, $votes_{p3list}(candidate)$,	Votes of a candidate on a candidate list, P2-list, P3-list
	votes(*candidate*)	Total votes of a candidate
	votes	Function of votes of a list (Parameter in GDSA)
	$votes_1(list)$	Total number of votes of a candidate list.
	$votes_2(p2list)$	Total number of votes of a P2-list.
	$votes_3(p3list)$	Total number of votes of a P3-list.
	preferentialbarrier	Preferential barrier (Dutch: voorkeurdrempel)
W		
	DISTRICTS, district	Electoral districts, an electoral districts
X		
	$N \times M$	Cartesian product
	ξ_{step}	Integer variable indicating how much of the roll back sequence or roll forward sequence is "consumed".
	$\xi'_{step,\,j}$	Another integer variable indicating how much of roll forward sequence is "consumed".
	$X_{candidate,j}$	P2-lists with unassigned seats on which a candidate is nominated
Y		
	Y	P2-lists without unassigned seats on which a candidate is nominated

$Y_{p2list,j}$	Set of candidates who should receive a seats in *p2list* provided that there are enough seats available

Z

$\mathbf{Z}$ = {..., -2, -1, 0, 1, 2, ...}	Integer numbers
$\mathbf{Z}^P$	Set of functions $P \rightarrow \mathbf{Z}$
$newseats_{step}(p)$	Seat assignment in step *step* to list p (Function in GDSA)
$Z_{max,}$	Lists that reach the maximum (largest average or largest remainder) (Variable in GDSA)
Z_{step}	Lists that are assigned a seat (Variable in GDSA)
$Z_{p2list,j}$	Set of candidates that receive a seat on P2-list *p2list* in the inner step j
Z_j	Set of candidates that receive a seat in the inner step j