Computational Thinking for the Legal Writing:
the case of Italian election law

Silvia Crafa
Dipartimento di Matematica - Università di Padova - Italy
silvia.crafa@unipd.it

Abstract

We examine the Italian election law as a case study to illustrate how the computational thinking can productively interoperate with the legal language to increase the transparency of the legal text, and to enable better reasoning about the procedural content of the law. The effort to rephrase the text of the law in algorithmic terms revealed that the election procedure is under-specified, so that the allocation of seats between constituencies may differ depending on the actual sequence of ballot operations performed by the scrutineers. This leads to legal uncertainty in a critical section of the election law that one would expect to be fully determined.

Therefore, we discuss the limits of the ‘technology of text’ in preventing unintended interpretations, questioning what is its normative power when the text of the law describes an under-specified or incomplete procedure. On the contrary, we put forward algorithmic normativity, that is the power of the algorithmic language (different from software’s code) to legally express procedures at an appropriate abstraction level, balancing transparency with scientific precision.

1 Introduction

The most distinctive feature of legal systems is that they are anchored in the technology of text, which allows to exploit the inherent ambiguity of the natural language as a powerful feature that sustains the tension between general rules and the singularity of acts and circumstances (Hildebrandt et al., 2023). On the other hand, there are situations where laws require the rigidity of unambiguous interpretations, for instance in some administrative procedure or parts of tax regulation.

A notable example is the election law of a country, where the matching from the set of citizens’ votes and the names of the elected people must be crystal clear. The procedures for vote counting, aggregating and finally assigning seats to candidates can be fairly complex: they can adopt a majoritarian or a proportional method, or some mix of the two; there can be electoral thresholds and majority bonus; candidates can run for a single party or for a coalition of parties, thus requiring a further redistribution of votes within the coalitions. The design of a suitable election system is a difficult and inherently political task, but in any case, the democratic principles require that the procedure for seats allocation be unambiguous, transparent and deterministic, which means that the list of elected people must be univocally determined by the set of citizen’s votes.

In this article we reflect on which language is the most appropriate to juridically define election procedures. The natural language guarantees the accessibility of the law to all citizens (that are able to read that language), thus enables transparency, but the textual definition of a complex procedure can be involved and confusing. By contrast, the adoption of the mathematical language prevents unclear and ambiguous textual descriptions by univocally defining a formula expressing the intended
proportional distribution of the citizens’ votes, at the price of knowing the symbolic language. On the other hand, the computational thinking is characterized by simultaneously defining a formula together with the procedure (i.e., the algorithm) to compute it, bringing in useful conceptual tools and concrete practices. Indeed, the algorithmic language is the most natural language to express procedures, and being more precise than text and more human-readable than machine code, naturally interoperates with both the legal (and political) reasoning and the mathematical analyses.

We draw attention on the important difference between the concepts of algorithm and software, so to keep a clear distinction between the specification of a procedure in a pseudo-code algorithmic language, and its possible multiple implementations through the software, that is executable machine code. While the software code provides a rigid closure to a specific procedure implementation, the algorithmic description retains part of the expressiveness of the natural language and keeps open the double play of contestability and predictability of law. Therefore, instead of the code-driven approach to law (Cohubicol project, 2019), which entails the automatic execution of software-based rules, we call attention to the algorithmic normativity, that is the power of the algorithmic language (rather than software) to express legislation.

We illustrate these ideas at work on the concrete case study of the Italian election law1, which establishes the vote counting procedures in Articles 77, 83, 83-bis, 84, 85. The textual description of the procedure for allocating seats according to the proportional method is very confusing. In particular, we identified a couple of remarkably unclear mechanisms that are repeatedly adopted: the local distribution of seats according to the proportional method of integer quotients and greatest remainders2, and the subsequent compensation procedure (D.P.R. 361/1957, Article 83, comma 1, item h) and item i), and Article 83-bis). In fact, the obscure application of the compensation procedure caused delays and public confusion over the actual outcome of the recent Italian legislative elections in September 2022.

The effort to rephrase the text of the law in algorithmic terms revealed that the election procedure is under-specified, so that the allocation of seats between constituencies may differ depending on the actual sequence of ballot operations performed by the scrutineers. Moreover, setting the procedure as an algorithm showed a number of (rare but admissible) scenarios that have not been regulated. In other terms, the analysis identified several sources of legal uncertainty in a critical section of the election law that one would expect to be fully determined.

It is widely acknowledged that a well-written and clear rule is an essential prerequisite for legal certainty. The techniques for legislative drafting tend to guarantee the intelligibility of laws, at least on a formal level, as the very basis of the constitutional principles of openness and transparency that underlie a democratic order (Piertangelo, 2023). Information technology offers wide-ranging facilities for drafting, managing and retrieving normative acts. In this article we overlook the technical tools, and we emphasize the conceptual contributions that computer science can offer to legal writing.

We argue that computational thinking provides for a useful perspective to reason about laws that intend to establish a procedure, and that the abstraction level of algorithms is best suited to capture the procedural nature of a given process. We then envision the adoption of a legal writing that relies on a richer set of technologies that goes beyond text, to include algorithmic pseudo-code, diagrams, images and other specification languages, and that productively interoperate by choosing each time the most suitable language to express a specific type of normativity.

---


2 In Italian: metodo proporzionale dei quozienti interi e dei maggiori resti.
2  *Text is law? The indeterminacy of the Italian election law*

Lessig’s “*Code is Law*” (Lessig, 1999) has now become a popular motto to hint at the advantages and the risks of replacing laws and regulations by technical regulation, which can be enforced through the automatic execution of software code. When considering law in its generality, regulation by code is always more specific and less flexible than the legal provisions it purports to implement, thereby giving software developers and engineers the power to embed their own interpretation of the law into the technical artefacts that they create (De Filippi and Hassan, 2016). By rephrasing the same argument, we can ask how instead the legislator can fix its intended interpretation in a text written in natural language, leaving no room for interpretations he meant to exclude. That is, what is the actual normative power of the "technology of text" when the legislator wants to enforce a specific interpretation?

The inherent ambiguity of the natural language is often seen as a feature of the legal system, which has an intrinsic open nature, thus requires an expressive and flexible language to ensure a proper application of the law on a case-by-case basis. However, a fruitful ambiguity often turns into a confused text that generates uncertainty. This happens in particular when using the natural language to describe a content that would be better expressed in a different language.

As an example, the Italian election law (D.P.R. 361/1957, Article 31) textually describes the geometry of the ballot papers, prescribing the relative positions of rectangles containing the symbols of parties and the names of candidates and coalitions:

2. The ballot paper shall contain the names and surnames of the candidates in the uninominal constituency, written within a dedicated rectangle, under which there shall be, within another rectangle, the symbol of the party the candidate is associated with. Next to the symbol, in the same rectangle, the names and surnames of the candidates in the multi-nominal constituency shall be listed in the order in which they were presented.

3. In the case of several parties linked in a coalition, the rectangles of each party and that of the candidate in the uninominal constituency shall be placed inside a larger rectangle. Within that larger rectangle, the rectangles containing the symbols of the parties as well as the names and surnames of the candidates in the in the multi-nominal constituency shall be placed below that of the candidate in the uninominal constituency on horizontal lines divided into two rectangles.

4. The width of the rectangle containing the first name and surname of the candidate in the uninominal constituency is double the width of the rectangles containing the symbol and the first and last names of the candidates in the multi-nominal constituency.

It would be quite difficult to produce a legally compliant ballot paper without looking at the picture provided as an attachment to the text of the election law (D.P.R. 361/1957, Allegato 3, Tabella A-bis), and available in Appendix A. Interestingly, the role of visual-nonlinguistic normativity and its relation with other legal elements has been only marginally studied (see for instance Lorini & Moroni, 2020 and Dudek, 2015).

---

3Art. 31 comma 2-4:2. *La scheda reca i nomi e i cognomi dei candidati nel collegio uninominale, scritti entro un apposito rettangolo, sotto il quale è riportato, entro un altro rettangolo, il contrassegno della lista cui il candidato è collegato. A fianco del contrassegno, nello stesso rettangolo, sono elencati i nomi e i cognomi dei candidati nel collegio plurinominale secondo il rispettivo ordine di presentazione. 3. Nel caso di più liste collegate in coalizione, i rettangoli di ciascuna lista e quello del candidato nel collegio uninominale sono posti all’interno di un rettangolo più ampio. All’interno di tale rettangolo più ampio, i rettangoli contenenti i contrassegni delle liste nonché i nomi e i cognomi dei candidati nel collegio plurinominale sono posti sotto quello del candidato nel collegio uninominale su righe orizzontali ripartite in due rettangoli. 4. La larghezza del rettangolo contenente il nome e il cognome del candidato nel collegio uninominale è doppia rispetto alla larghezza dei rettangoli contenenti il contrassegno nonché i nomi e i cognomi dei candidati nel collegio plurinominale.*
Similarly to the use of the graphical language, one can resort to the algorithmic language to clarify the vote counting procedures expressed in the election law, so to increase the transparency of the law, and to enable better reasoning about the content of the rule, which is quite involved and has been subject of many modifications, specialist studies and appeals to the Constitutional Court. We focus on Article 83 and 83-bis, which state that the distribution of seats takes place in successive steps, proceeding from the national territorial level, to the district level, down to the multi-nominal constituency level. Each level predetermines the number of seats to which the level below it is entitled. Seats are then allocated by the proportional method, on the basis of the national proportion, the district proportion and the constituency proportion, respectively, following the so-called method of quotients and largest remainders. At each level, seats are also distributed first between coalitions of parties and single parties exceeding the electoral threshold, and then, for each coalition, between the parties belonging to the coalitions. However, the allocation procedure based on the method of quotients and largest remainders may lead to the allocation of a number of seats that does not correspond to the number of seats predetermined at the higher level. The law therefore regulates the adjustment procedure, both at the district and at the constituency levels.

What is relevant to our investigation is the fact that a careful analysis of the text of these articles shows that the mechanism of seats distribution is written in way that leaves open the choice between different possibile sequences of operations (i.e., different algorithms), leading to different final seats allocations, as detailed below. Additionally, although the official operating manual adopted by ballot scrutineers is not available, the examination of some documents provided by the Chamber of Deputies (2019 and 2022), points to the application of a very specific procedure, which is compatible with the text of the law, even if it is not its most immediate and natural interpretation.

In summary, the text of the law describes a seats allocation mechanism that can be actually executed in many different ways with different outcomes, whereas the practically adopted procedure is hidden in the vade mecum for ballot scrutineers, which is not publicly available nor has the status of law or official regulation. Therefore, in a "Text is Law" perspective, we can ask what is the the actual normative power of the Article 83 and 83-bis? Would any interpretation of their text have the same legal force, or the specific interpretation adopted by the vade mecum has a special status?

2.1 An under-specified procedure that may lead to non-deterministic results

The problem with the text of the law stems from the fact that the distribution of seats between districts involves repeating a certain number of operations in each district, but there is a lack of clarity as to which is the exact set of operations that are to be repeated, and in what order the repetitions are to be carried out. More precisely, the Article 83 item h)\footnote{L’Ufficio centrale nazionale, [...] h) procede quindi alla distribuzione nelle singole circoscrizioni dei seggi assegnati alle coalizioni di liste o singole liste di cui alla lettera e). A tale fine determina il numero di seggi da attribuire in ciascuna circoscrizione sottraendo dal numero dei seggi spettanti alla circoscrizione stessa ai sensi dell’articolo 3, comma 1, il numero dei collegi uninominali costituiti nella circoscrizione. Divide quindi la somma delle cifre elettorali circoscrizionali delle coalizioni di liste e delle singole liste ammesse al riparto per il numero di seggi da attribuire nella circoscrizione, ottenendo così il quoziente elettorale circoscrizionale. Ottiene quindi le maggiori partite decimali del quoziente così ottenuto. Divide poi la cifra elettorale circoscrizionale di ciascuna coalizione di liste o singola lista per il quoziente elettorale circoscrizionale, ottenendo così il quoziente di attribuzione. La parte intera del quoziente di attribuzione rappresenta il numero dei seggi da assegnare a ciascuna coalizione di liste o singola lista. I seggi che rimangono ancora da attribuire sono rispettivamente assegnati alle coalizioni di liste o singole liste per le quali queste ultime divisioni hanno dato le maggiori parti decimali e, in caso di parità, alle coalizioni di liste o singole liste che hanno conseguito la maggiore cifra elettorale nazionale; a parità di quest’ultima si procede a sorteggio. Esclude dall’attribuzione di cui al periodo precedente le coalizioni di liste o singole liste alle quali è stato già attribuito il numero di seggi ad esse assegnato a seguito delle operazioni di cui alla lettera f). Successivamente l’Ufficio accerta se il numero dei seggi assegnati} is the following, where we emphasize its textual
The national central office, [...]

shall then proceed to distribute in the individual districts the seats allocated to the coalitions of parties or individual parties referred to in paragraph e).

To this end it shall determine the number of seats to be allocated in each district by [...] obtaining the district’s electoral quotient.[...]

It then divides the sum of the [...] obtaining the district’s electoral quotient, thus obtaining the allocation quotient.

The integer part of the allocation quotient represents the number of seats to be allocated to each coalition or individual party.

The seats remaining to be allocated shall be allocated respectively to the coalitions of parties or individual parties for which these last divisions have given the largest decimal parts and, in the event of equal numbers, to the [...].

It shall exclude from the allocation referred to in the preceding sentence those coalitions of parties or individual parties [which have already been allocated the number of seats to which they were entitled according to the previous higher-level allocation].

The Office shall then ascertain whether the number of seats allocated in all the districts to each coalition of parties or individual parties corresponds to the number of seats determined in accordance with [the previous higher-level allocation]. If not, it shall carry out the following operations [...].

Observe that the text identifies the following set of operations to be carried out for each district:

Step A: compute the number of seats to be allocated in the district (line 4), the district’s electoral quotient (line 5), and the district’s allocation quotient of each coalition or single party (lines 6-7);

Step B: allocate to each coalition or party the number of seats equal to the integer part of its district’s allocation quotient (lines 8-9);

Step C: the remaining district’s seats are allocated to the parties or coalitions with the largest decimal parts of the district’s allocation quotients (lines 10-12);

Step D: coalitions or parties that have already been allocated the number of seats to which they are entitled at national level are excluded from Step C, that is they are excluded from the allocation of seats due to the largest remainders of their district’s allocation quotients (lines 13-15).

Then there is a final step, to be executed only once, that checks the correctness of the seats allocation performed so far, and possibly triggers the adjustment procedure. It is not trivial to understand why the allocation mechanism prescribed by lines 4-15 can possibly lead to the assignment of the wrong number of seats with respect to the pre-determined national allocation. Full understanding would require a mathematical modelling that is out of scope: here we aim to focus not on the meaning of the election mechanism, but on the way it is described.

We remark that the textual description of Article 83 does not specify how these operations are to be repeated for each district. Then we could either repeat the entire sequence of Steps A-D for each district, or we could repeat a single step for all districts before moving to the next step. This can be rephrased in the algorithmic language by saying that the two Algorithms A and B depicted in the following figure both comply with the text of Article 83 above:

As an example, given the decimal number 31.425, its integer part is 31 and its decimal part is 0.452.
Algorithm A

1. Consider the list of districts
2. **for each** district in the list:
3.   execute Step A
4.   execute Step B
5.   execute Step C + Step D
6. Check the need for seats adjustment
7.   ....

Algorithm B

1. Consider the list of districts
2. **for each** district in the list:
3.   execute Step A
4. **for each** district in the list:
5.   execute Step B
6. **for each** district in the list:
7.   execute Step C + Step D
8. Check the need for seats adjustment
9.   ....

Using the language of computer science, the law’s text is said to be the **specification**, that is what we have to do, whereas the **algorithm** establishes how to fulfil the specification. A natural question is whether the two algorithms are equivalent, that is, even if they both comply with the same specification, do they produce the same output, i.e., the same seats allocation?

In general, two algorithms like those above are equivalent only if the execution of each step is independent of the execution of the previous steps. This is indeed the case of Steps A and B, which can be executed in any district in any order, therefore, lines 1-4 of Algorithm A are equivalent to lines 1-5 of Algorithm B. Instead, in a given district, the seats allocation according to the largest decimal parts of the district’s allocation quotients (Step C) depends on whether a coalition or a party has already reached the number of seats it is entitled to at national level (Step D), which in turn depends on how many seats has been already assigned to that coalition or party in the districts where the Steps B and C+D has been already executed. In other terms, the effect of executing the Step C+D in a district depends on the effects of the executions of Steps B and C+D in the districts already considered.

A detailed analysis of the different impact of these algorithms on the seats distribution has been studied (Crafa 2023), we just mention here that by executing the Article 83 according to Algorithm A we have that the seats allocation depends on the order in which districts are considered. Hence, given the fixed set of votes expressed by citizens, the allocation of seats would change depending on the order in which the districts are considered, which is not determined by the law, thus causing a dangerous indeterminacy. The same non-determinism due to districts’ ordering affects also Algorithm B, but in this case we can additionally prove that there is no need of the final check for seats adjustment, since the execution of lines 1-7 of Algorithm B always assign the expected number of seats. Interestingly, the format of Algorithm B corresponds to a former Italian election law (law n. 277/1993 aka Legge Mattarella), that also prescribed a specific ordering for districts, which should be considered from the least populated to the most populated.

However, by analysing the documentation available at the Chamber of Deputies (see Crafa, 2023 for details), it seems to emerge that the actual execution of the procedure defined in Article 83 corresponds to an even different algorithm, which corresponds to the Algorithm C given in the next figure. Algorithm C first allocates in all districts all the seats due to the integer parts of the allocation quotients, then uses the decimal parts to allocate all the remaining seats of every district only to coalitions or parties that have not already reached their expected number of seats just with the allocations due to the integer parts.

The relevant aspect of this algorithm is that, by using in this way the exclusion clause expressed in Step D, the allocation of residual seats (Step C) is no longer dependent on the order in which the
Algorithm C

1 Consider the list of districts, and the list L of coalitions or single parties
2 for each district in the list:
3   execute Step A
2 for each district in the list:
4   execute Step B, i.e., allocate seats according to the integer parts
5   execute Step D, i.e., excludes form the list L those who have already
   reached the number of seats they are entitled to
6 for each district in the list:
7   execute Step C only for those still in the list L,
   i.e., allocate seats according to the decimal parts
8 Check the need for seats adjustment
9 ....

districts are considered, then provides deterministic outcomes. Thus Algorithm C, which seems to
 correspond to the undocumented vade mecum for ballot scrutineers, does not suffer from the indeter-
 minacy problem highlighted in Algorithm A and B.

In conclusion, since all three algorithms are compatible with the legal text, the question remains
whether the interpretations corresponding to these algorithms all have the same legal force, or whether
the fact that only one avoids a serious problem of indeterminacy of results excludes the other two as
possible interpretations.

2.2 An incomplete specification

In the previous subsection we focused on the procedure for allocating seats within districts, but the
same holds for the text describing the distribution of the seats between the parties belonging to the
coalitions (Article 83 item i), and for the distribution of the district’s seats between the multi-nominal
constituencies of the district (Article 83-bis). In all these cases, the text of the law, by not specifying
the order in which the operations are to be carried out, defines an under-specified (if not incorrect)
procedure, which does not unambiguously identify the allocation of seats in the national territory.

The analysis of the law (Crafa, 2023) revealed a further problem, that can also be attributed to
the way the law is written. In the final part of Article 83 item h), and similarly in Article 83 item
i), the description of the seats adjustment procedure prescribes a specific ordering to adjust the seats
distribution, but fails to cover all possible scenarios. There are in fact two (rather rare) cases, due to
situations of equal decimal numbers, where the law does not indicate how to proceed, paving the way
for an illegitimate choice made by the actual executors of the operations.

In the algorithmic language in this case the specification is said to be incomplete, leaving room
for the executor of the procedure to decide what to do in the non-covered cases, which is clearly
problematic for an election law.

3 Algorithmic normativity: the interface between text and code

Choosing a specific electoral mechanism is a complex matter, that has a primary political nature.
But this is precisely why it is important to have a full understanding of the mechanism in order to
choose. Indeed, experts and institutions base their decisions on in-depth analyses, which also take advantage of mathematical studies. We think that in this context it is fruitful to draw attention to the distinction between the mathematical and the algorithmic approach, which are often confused and identified by non-experts. Intuitively, mathematics focus on defining a suitable formula, e.g., for votes aggregation or seats allocation, whereas computational thinking\(^6\) is characterized by simultaneously defining a formula and the procedure/algorithm to compute it. Moreover, computer science provides for a number of conceptual and practical tools to properly define a problem (i.e., a well formed and complete specification) and to check whether an algorithm correctly and satisfactorily solves the given problem in all possible cases.

To exemplify, Figure 1 illustrates the use of three different languages –mathematical, textual and algorithmic– to represent the same content, namely the procedure for distributing the votes of a coalition among its parties (D.P.R. 361/1957, Article 77). The law textually describes how to distribute the votes according to the so-called proportional method of integer quotients and greatest reminders. The adoption of the mathematical symbolic language leads to formulas involving indexed summations and arithmetic operations expressing the final distribution of votes that must be reached. On the other hand, the algorithmic perspective focuses on the sequence of operations to be carried over to reach the target distribution of votes.

We omit the detailed explanation of the content of Figure 1, but we highlight the intermediate status of the algorithmic description, which belongs to an abstraction level that is between the textual description and the mathematical formal modelling. Reasoning at this abstraction level can be useful to clarify ideas, to compare solutions, to test possibilities and to verify the correctness of the established procedures, as we have done in Section 2.1.

Additionally, besides corroborating the reasoning, drafting the procedural part of a law in algorithmic style is useful also to get back to the natural language and write a clearer legal text that avoids problematic interpretations that the law intends to exclude. For instance, if the intended interpretation of Article 83 was actually that of Algorithm C, the legislator had better write a different text, providing for a specification of the seat allocation procedure that would have excluded the other two problematic algorithms by making them non-compliant. Such a satisfactory specification could be obtained starting from a textual description of the definition of Algorithm C, and then rephrasing it according to the style of the legal text, as illustrated in Appendix B.

In our analysis we haven’t used the word software yet, rather insisting on using the term algorithm or logical algorithm. Indeed, even if non-experts tend to use them as synonyms, the difference between algorithm and software is crucial also in the context of the legal writing. Both the algorithm and the software express the operating logic of a programme, that is the sequence of steps leading to the solution of the problem. But they differ in the adopted language: the software is written as code of a specific programming language so to be fully unambiguous (as maths) and machine-readable, while the algorithm can adopt any sufficiently precise language without conforming to strict syntactic rules. The algorithms’ language usually lies between natural language, mathematics and programming languages, so to be human-readable and at the same time to scientifically support the reasoning on its content.

To illustrate, Figure 2 shows how the logical algorithm in Figure 1 can be first rephrased using a more technically precise language, called pseudo-code\(^7\), and then implemented as an executable software written in the Python programming language. Without going into technical explanations, we

---

\(^6\)for simplicity we use here the terms computational thinking and algorithmic thinking as synonyms, even if the first one is a broader concept (Nardelli, 2019 [?]).

\(^7\)the pseudo-code command \texttt{X := exp} stands for computing the value of the expression \texttt{exp} and assigning, i.e., naming, this value to \texttt{X}. The symbol \# denotes comments, providing for human-readable explanations.
**Mathematics:**
For each district $i$, for each coalition $j$ containing $k_j$ parties, the votes assigned to each party $p$ in the coalition $j$ are:

$$v_{i,j}^p = v_{i,j}^p + \left\lfloor \frac{v_{i,j}^p}{\sum_{p \in [1,k_j]} v_{i,j}^p} \cdot \text{vcl}_{i,j} \right\rfloor + \delta_{N_{i,j}^p}$$

where $N_{i,j}^p = v_{i,j}^p - \sum_{p \in [1,k_j]} \left\lfloor \frac{v_{i,j}^p}{\sum_{p \in [1,k_j]} v_{i,j}^p} \cdot \text{vcl}_{i,j} \right\rfloor$

(The votes assigned to the party are those that have been directly collected by the party ($v_{i,j}^p$), plus a proportion of the votes collected by the coalition ($\text{vcl}_{i,j}$), assigned according to the integer part of the proportion (the second addend of the sum) plus $1$ or $0$ residual vote due to the largest reminder of the proportion (the third addend $\delta_{N_{i,j}^p}$).)

**Legal Text:**
[Article 77 item c] "determina la cifra elettorale di collegio uninominale di ciascuna lista. Tale cifra e’ data dalla somma dei voti validi conseguiti dalla lista stessa nelle singole sezioni elettorali del collegio uninominale e dei voti espressi a favore dei soli candidati nei collegi uninominali collegati a più liste in coalizione di cui all’articolo 58, terzo comma, ultimo periodo, attribuiti alla lista a seguito delle seguenti operazioni: l’Ufficio divide il totale dei voti validi conseguiti da tutte le liste della coalizione nel collegio uninominale per il numero dei voti espressi a favore dei soli candidati nei collegi uninominali, ottenendo il quoziente di ripartizione. Divide poi il totale dei voti validi conseguiti da ciascuna lista per tale quoziente. La parte intera del quoziente così’ ottenuto rappresenta il numero dei voti da assegnare a ciascuna lista; i voti che rimangono ancora da attribuire sono rispettivamente assegnati alle liste per le quali queste ultime divisioni abbiano dato i maggiori resti, secondo l’ordine decrescente dei resti medesimi."

**Logical Algorithm:**

```
# Algorithm for distributing the votes of a coalition between its parties
1 Compute the distribution quotient $Q$ by dividing the sum of the votes of every party by the votes of the coalition
2 for each party $p$ in the coalition:
3    Divide the votes of the party $p$ by the distribution quotient $Q$
4    From that number take the integer part and the reminder; name them $I[p]$ and $R[p]$
5 for each party $p$ in the coalition:
6    Add to the direct votes of $p$ a number equal to its integer part $I[p]$
7 Compute the number $N$ of votes remaining to be distributed: subtract from the total votes of the coalition
8    the sum of the integer parts of every party, which have already been distributed
9 Take the list $L$ of parties sorted from that with largest reminder to that with the smaller reminder
10 until $N > 0$
11 Assign one additional vote to the next party in the ordered list $L$
```

Figure 1: Distribution of the coalition’s votes among its parties

highlight two interesting aspects emerging form the transition from algorithm to software: the role of data structures and the significance of machine executability.

**3.1 Algorithms are not just procedures: the power of data structures**

In computer science it is well known that both the definition and the efficiency of an algorithm are heavily influenced by the choice of the underlying data structures. For instance, an algorithm that operates on a number of items needs to work on a specific data structure that somehow collects these items. Therefore the algorithm designer has to choose between a list, a set, a queue, an indexed record, a key-value mapping, a graph, and many other computational structures representing (small, large or even infinite) collections, each one equipped with specific operations to manipulate the content. The most relevant differences due to data structures are in terms of the algorithm’s efficiency, i.e. the speed of its execution, and they are critical when the number of items is large. However, even in
Algorithm in pseudo-code:

# Algorithm for distributing the vcl votes of a coalition between its parties
# Data structures: Votes maps each party p to its direct votes Votes[p]. Int and Rest are initially empty mappings
1 Q := sum(values(Votes))/vcl  # Compute the distribution quotient
2 for each party p in the coalition:
3   Int[p] := (Votes[p]/Q).integer  # Compute the integer part
4   Rest[p] := (Votes[p]/Q).remainder  # Compute the reminder
5 for each party p in the coalition:
6   Votes[p] += Int[p]  # Add to its direct votes a number equal to its integer part
7 N := vcl - sum(values(Int))  # Compute the number of votes remaining to be distributed
8 L := sort_by_values(Rest)  # Take the list of parties sorted from the largest reminder to the smaller
9 until N > 0
10 Votes[L.next] += 1  # Assign one additional vote to the next party in the ordered list

Software in Python code:

# Function for distributing the votes of a coalition between its parties
# let vcl be the coalition’s votes, and let votes be a key-value map from a party to its direct votes
1 def distribution(vcl, votes):
2   q = sum(votes.values()) / vcl  # distribution quotient
3   integ = {}; rest ={}
4   for p in votes.keys():
5     (rest[p], integ[p]) = math.modf(votes[p] / q)  # integer part and reminder
6   for p in votes.keys():
7     votes[p] += integ[p]  # add votes due to the integer part
8   n = vcl - sum(integ.values())  # number of votes remaining to be distributed
9   l = list(sort_by_values(rest).keys())  # sort parties according to remainders
10  i = 0
11  while n > 0:
12     votes[l[i]] += 1  # assign 1 vote to the next party in the list
13     n -= 1
14     i = i + 1
15  return votes

Figure 2: From Algorithm to Software

simple situations, choosing specific data structures orients the algorithmic thinking and affects the design of the actual procedure to be executed, since data models entail specific operations on data. Indeed, computational thinking is not just about designing procedures, but also about organizing information into structured data. For instance, in the case of a collection of items, a main choice is that between an ordered or an unordered collection. This leads one to reason about the nature of these items, and the possible need to sort them in some way according to the goal to be achieved by means of the computation.

In the case of the Italian election law, consider again the Article 83 item h), where a number of operations had to be carried out for each local district. The computational perspective identifies the presence of a collection of districts guiding the number of repetitions of certain operations. Therefore, an actual algorithm requires this collection to be encoded as a suitable list of districts (see line 1 in all the three algorithms presented in Section 2.1). Reasoning on the ordering of such a list has been

---

8The algorithm in Figure 2 is based on three data structures: Votes, Int, and Rest. They are key-value mappings, that associate to each party p a specific value denoted by Votes[p], Int[p], and Rest[p]. They support operations like the sum of all the values (line 1 and 7) and sorting the keys (the parties) according to their associated values (line 8).
decisive to find the indeterminacy issue of Algorithm A and Algorithm B, as well as to match the format of Algorithm B with the former Italian election law (Law n. 277/1993) when a specific district ordering is adopted.

### 3.2 From code normativity to algorithmic normativity

The most common feature of algorithms is that they are amenable to automatic execution by means of their implementation in a machine-readable language, e.g., the software given in Figure 2 is an executable Python code. The power of using software in the legal field, also called code-driven law, has been increasingly investigated: from the blockchain-based smart contracts, up to the creation of machine-consumable versions of some types of rules issued by governments and public administrations like the tax office, student grant provision or social security agency (see for instance the projects Rules as Code, 2020, Catala, 2022, Akoma Ntoso, 2018 Crafa, 2022).

In Section 2 we have already mentioned the limits of the code-driven enforcement of rules hard-wired into the software, as well as the limits of the 'technology of text' to specifically prevent unintended interpretations. We think that the abstraction level provided by algorithms, being intermediate between the text and the software, conveys a specific kind of normativity, that we call algorithmic normativity, that differs from both the text normativity and the code normativity (Cohubicol project, 2019), and that deserves to be further investigated.

While the software code leads to a rigid closure to a specific procedure implementation, the algorithmic description guarantees a higher abstraction level, with conceptual constructs (like a coalition of parties or a multi-nominal constituency) that are not forced into specific programming constructs. Moreover, the algorithmic language being more precise than text and more human-readable of code, naturally supports reasoning about the defined procedures, as illustrated above. Thus it increases the transparency and keeps open the double play of contestability and predictability of law. At the same time, the fact that an algorithmic description can be implemented in an executable software leaves room for reasoning about the content of the legal rule not just through argumentation, but also by running simulations, so to (automatically) check the outcomes and test the consequences of the defined procedure in many different scenarios. For instance, the software in Figure 2 can be used to check the correctness of the ballot operations manually executed by scrutineers, or to test the fairness of the votes’ distribution according to the method of integer quotients and largest reminders in specific scenarios. Additionally, the implementations of the algorithms in Section 2.1 can be used as building blocks to study the political effects of changing the seats allocation procedure.

To conclude, having addressed the election law, it is worth mentioning a very different way of using algorithms in elections, that is the adoption of the electronic vote. Automatic procedures for casting votes as well as to automatically count votes are very different from what we have discussed in this article. An electoral system with electronic vote is based on the idea of running these procedures, that is executing specific software that implements the logical algorithms defined by the election law. This falls into what we called code normativity, bringing in the rigidity of the software-based rules, its non transparency to human reading and human verification, hence reducing the contestability of the outcome required by democratic elections. Instead of the electronic vote, we are suggesting an algorithmic election law, that uses digital tools as a safety belt for the definition and the enforcement of the election system.
4 Conclusions

We argued that computational thinking may offer a useful perspective to reason about juridical texts that intend to legally establish procedures. More precisely, we showed that the algorithmic language is keen to capture the procedural nature of a given process, promoting a proper writing style that improves non-ambiguity, clarity, transparency and (if needed) deterministic outcome. This approach proved to be effective in identifying a number of problematic indeterminacy issues in the seats allocation procedure defined in the Italian election law.

We acknowledge that any language entails a specific gap between what is written and what was intended, and we illustrated this gap in using the natural language, the mathematical language, the algorithmic language and the programming language to express the same procedural content. In general, the election law case study illustrates how the algorithmic language can productively interoperable with legal language, bypassing the limits of the text in preventing unintended interpretations, as well as the rigidity of software-based rules. We then put forward the algorithmic normativity, that is the power of the algorithmic language to express legislation at an appropriate abstraction level, balancing transparency with scientific precision.

This is very different from the legislative techniques know as ‘legimatics’ (Biagioli, Mercatali and Sartor, 1993), that started in the ‘80s with the so-called legal expert systems (nowadays related to modern systems of Artificial Intelligence), that aimed to translate legal texts into a machine readable language that supports automatic logical deductions. That approach contributes not only to making legal texts more accessible, but also and above all to make legislative design and drafting more responsive to the needs of computer applications (Cappello, 2023). Instead, in this article the purpose of using of computational thinking and algorithmic writing is not to meet the needs of computer applications, but to respond to the need for clarity in those parts of the legal texts that define procedural rules.

To conclude, in this article we focused just on algorithmic specification and the organization of information into data structures, but further research can be devoted to the study of a wider set of cognitive tools offered by the computational thinking. For instance, encapsulation, separation of concerns, modularity, reuse, testing, verification, versioning and refactoring, are concrete principles (coupled with practical tools) that can be useful in the legal writing, and deserve more investigation.

References


CoHuBiCoL research project - counting as a human being in the era of computational law. Three types of normativity.


P. De Filippi and S. Hassan. Blockchain technology as a regulatory technology: From code is law to law is code. First Monday, 21(12), Nov. 2016.

M. Hildebrandt et al. Research study on text-driven law. Cohubicol project.


M. Pietrangelo. La conoscibilità della legge per via informatica e telematica. Codice di drafting Libro VI – Produzione normativa e tecnologie dell’informazione.

Cracking the Code: Rulemaking for humans and machines.
A Ballot paper

Official model of the internal side of the ballot paper for the election of the Chamber of Deputies (D.P.R. 361/1957, Allegato 3, Tabella A-bis),
B A possible amendment to Article 83 item h)

The following simple amendment to Article 83 item h) corresponds to what seems to be the actual operations established by the vade mecum for the ballot scrutineers (i.e., Algorithm C of Section 2.1), and it is enough to exclude the two problematic and non-deterministic interpretations corresponding to Algorithm A and B:

1. The national central office, [...]
2. h) shall then proceed to distribute in the individual districts the seats allocated to the coalitions of parties or individual parties referred to in paragraph e).
3. To this end it shall determine the number of seats to be allocated in each district by [...].
4. It then In every district, it divides the sum of the [...] obtaining the district’s electoral quotient.[...]
5. It then divides the district’s votes of each coalition of parties or individual parties by the district’s electoral quotient, thus obtaining the allocation quotient.
6. The integer part of the allocation quotient represents the number of seats to be allocated to each coalition or individual party. In every district, it then allocates to each coalition or individual party the number of seats equal to the integer part of the corresponding allocation quotient.
7. The seats remaining to be allocated shall be allocated respectively to the coalitions of parties or individual parties for which these last divisions have given the largest decimal parts and, in the event of equal numbers, to the [...].
8. It shall belong to the allocation referred to in the preceding sentence those coalitions of parties or individual parties which have already been allocated the number of seats to which they were entitled according to the previous higher-level allocation.
9. The seats remaining to be allocated shall be distributed by the office among the coalitions or individual parties to which the allocation referred to in the preceding sentence has not already attributed the number of seats to which they were entitled according to the previous higher-level allocation.
10. To this end, in each district the remaining seats shall be allocated respectively to the coalitions or individual parties with the largest decimal parts of the allocation quotient and, in the event of equal numbers, to the [...].
11. The Office shall then ascertain whether the number of seats [...]. If not, it shall carry out [...] 9

9 L’Ufficio centrale nazionale, [...] h) procede quindi alla distribuzione delle singole circoscrizioni dei seggi assegnati alle coalizioni di liste o singole liste di cui alla lettera e). A tale fine determina il numero di seggi da attribuire in ciascuna circoscrizione sottraendo dal numero dei seggi spettanti alla circoscrizione stessa ai sensi dell’articolo 3, comma 1, il numero dei collegi uninominali costituiti nella circoscrizione. In ogni circoscrizione, divide quindi la somma delle cifre elettorali circoscrizionali delle coalizioni di liste e delle singole liste ammesse al riparto per il numero di seggi da attribuire nella circoscrizione, ottenendo così il quoziente elettorale circoscrizionale. In caso di pari con l’eventuale parte frazionaria del quoziente così ottenuto. Divide poi la cifra elettorale circoscrizionale di ciascuna coalizione di liste o singola lista per la somma delle cifre elettorali di ciascuna coalizione di liste o singola lista. Assegna quindi a ciascuna coalizione di liste o singola lista il numero di seggi pari alla parte intera del corrispondente quoziente di attribuzione. Successivamente l’Ufficio procede a ripartire i seggi che rimangono ancora da attribuire tra le sole coalizioni di liste o singole liste alle quali l’attribuzione di cui al punto precedente non ha già attribuito il numero di seggi ad esse assegnato a seguito delle operazioni di cui alla lettera f). A tale fine, in ogni circoscrizione, successivamente l’Ufficio procede a ripartire i seggi che rimangono ancora da attribuire tra le sole coalizioni di liste o singole liste alle quali l’attribuzione di cui al punto precedente non ha già attribuito il numero di seggi ad esse assegnato a seguito delle operazioni di cui alla lettera f). Successivamente l’Ufficio procede a ripartire i seggi che rimangono ancora da attribuire tra le sole coalizioni di liste o singole liste alle quali l’attribuzione di cui al punto precedente non ha già attribuito il numero di seggi ad esse assegnato a seguito delle operazioni di cui alla lettera f).