Digital Society
https://doi.org/10.1007/s44206-024-00103-4

ORIGINAL PAPER

Algorithmic Thinking for the Legal Writing: The Case of Italian Election Law

Silvia Crafa[1]

Received: 3 July 2023 / Accepted: 20 March 2024

Abstract
We examine the Italian election law as a case study to illustrate how the algorithmic 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 may lead to legal uncertainty in a critical section of the election law that one would expect to be fully determined. We then discuss the difference between algorithm and software in the legal context, illustrating how the algorithmic language acts as an interface between the textual description of a legal procedure and its mathematical or computational formalization. Hence we put forward the concept of 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.

Keywords Computational law · Algorithmic thinking · Italian election law · Legal drafting

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 (Diver et al., 2023). On the other hand,

✉ Silvia Crafa
silvia.crafa@unipd.it

1 Dipartimento di Matematica, Università degli Studi di Padova, Padua, Italy

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 in democratic elections the procedure for seats allocation is expected to be deterministic (*i.e.*, the seats allocation be *univocally* determined by the set of citizen's votes), unambiguous and transparent.

In this article we reflect on which *language* is the most appropriate to juridically define election procedures. On the one hand, 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. On the other hand, 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. In between natural language and maths there is algorithmic thinking, which is characterized by simultaneously defining a formula (*i.e.*, a problems's solution) together with the procedure (*i.e.*, the algorithm) to compute it, bringing in useful conceptual tools and concrete practices. We refer to the *algorithmic language* as a broad set of specification languages, ranging from simple but clearly ordered lists of textual instructions, like those of a cooking recipe, to the sophisticated formalism of UML[1] diagrams, including graphical representations such as flowcharts. Therefore the algorithmic language is the most natural language to express procedures. Being more precise than text and more human-readable than machine code, it naturally interoperates with both the legal (and political) argumentation 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 law[2], which establishes the vote counting procedures in Articles 77, 83, 83-bis, 84, 85.

[1]Unified Modeling Language. http://uml.org/.

[2]Testo Unico delle leggi recanti norme per la elezione della Camera dei Deputati, di cui al decreto del Presidente della Repubblica 30 marzo 1957, n. 361, and subsequent modifications. (D.P.R. 361/1957).

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 remainders[3], and the subsequent compensation procedure (D.P.R. 361/1957, Article 83, comma 1, items h) and 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 Italian legislative elections in September 2022 (Ansa, 2022; Sala, 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.

As mentioned above, the openness to multiple interpretations is a distinctive feature of legal systems, and legal scholarship as well as legal practice have developed means to resolve competing interpretations. Hence the meaning of a law does not depend only on its text, and the promises of transparency and predictability of the law always depend on being able to consult all legal sources, like inner-legal debates or court decisions, not only the text of the law itself. However, 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 (Pietrangelo, 2023). Information technology offers wide-ranging facilities for drafting, managing and retrieving normative acts. In this article we overlook the technical tools and we stick to a textualistic approach, emphasizing the conceptual contributions that the algorithmic language can offer to legal writing. We argue that algorithmic thinking provides for a useful perspective to reason about laws that intend to establish a procedure, and that the abstraction level of algorithms (differently from that of software) is best suited to capture the procedural nature of a given process.

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 & Hassan, 2016; Artosi, 2021; Huttner & Merigoux, 2022; Rosengrün, 2022). By rephrasing the

[3]In Italian: metodo proporzionale dei quozienti interi e dei maggiori resti.

same argument, we can ask how instead the legislators can fix their intended interpretation in a text written in natural language, leaving no room for interpretations they meant to exclude. That is, what is the actual normative power of the "technology of text" when the legislators want 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 (Diver et al., 2023). However, a fruitful ambiguity might turn into a confused text that generates uncertainty. According to Renato Borruso, a former judge of the Italian court of Cassation, the need to find an agreement based on compromise often pushes the legislators to formulate laws in an evasive or elastic way, and this problematically increases the importance of the jurisprudential activity of legal interpretation, giving judges enormous power that, to a certain extent, becomes pathologic and undermines the tripartition of the fundamental powers of the State (Borruso et al., 2004). Borruso, as well as other legal scholars linked to the Legal Cybernetics perspective (see Meldman & Holt, 1971; Sergot, 1991; Biagioli et al., 1993; Contissa et al., 2021 for an overview of the subject) goes so far as to propose drafting laws in a formalised language, so that legal reasoning can be computed by a Legal Expert System based on the rules of formal logic. Instead, we call attention to the idea that problematic textual formulations of the law happens in particular when *using the natural language to describe a content that would rather be better expressed in a different language*.

As an example, the Italian election law (D.P.R. 361/1957, Article 31[4]) 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

[4]Art. 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.*

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 1.

The role of visual-nonlinguistic normativity and its relation with other legal elements has been widely studied (Dudek, 2015; Aguilera, 2019; Siniscalchi, 2019; Lorini & Moroni, 2020, among others). This line of research addresses so-called "drawn norms" or "graphical norms", which are visualizations prepared not by the legal scholars or practitioners but by the legislators themselves, and included by them right in the texts of legal acts with no (fully equivalent) linguistic encoding. This is the case, for instance, of traffic signs and land-use planning regulations, which are not just descriptive but normative graphic representations that directly state prescriptions. Graphical norms are acknowledged not simply as illustrative or supplementary instruments, but as outright norms, that can only partially translated in a linguistic description (Dudek, 2015; Lorini & Moroni, 2020). However, Lorini and Moroni suggest that the philosophical and legal problems of normative drawings should be further discussed in order to give greater ground to graphical norms, which may shed new critical light also on the more widely discussed written norms.

Similarly to the case of the format of the ballot paper, which is better expressed adopting the graphical language, another prominent aspect of the election law that fails to be properly described using the natural language is the seats allocation procedure. The way it is expressed in the law is quite involved and has been subject of many modifications, specialist studies (Lodato et al., 2014; Ricca & Scozzari, 2019) and appeals to the Constitutional Court[5] (Camera dei Deputati, 2022). Therefore one could resort to the algorithmic language to clarify the procedure, so to increase the transparency of the law and to enable better reasoning about the content of the rule.

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

[5]Corte Costituzionale, sentenza n.1 del 2014 e sentenza n.35 del 2017.

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 Camera dei Deputati (2020 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 officially provided by Ministero dell'Interno, which is not publicly available. 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? The consultation of other legal sources did not reveal any indication on this specific aspect of the election law[6], hence we leave this question open for discussion by legal scholars.

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 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)[7] is the following, where we emphasize its textual structure:

[6]Source: author's private discussions with legal experts in Italian election law. The details of the technical analysis described in this article have been submitted to the Italian public institutions.

[7]*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. Nell'effettuare tale divisione non tiene conto dell'eventuale parte frazionaria 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 in tutte le circoscrizioni a ciascuna coalizione di liste o singola lista corrisponda al numero di seggi determinato ai sensi della lettera f). In caso negativo, procede alle seguenti operazioni, [...].*

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 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.

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 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].

9 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 3), the district's electoral quotient (line 4), and the district's allocation quotient of each coalition or single party (line 5);
Step B: allocate to each coalition or party the number of seats equal to the integer part[8] of its district's allocation quotient (line 6);
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 (line 7);
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 (line 8).

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 3–8 can possibly lead to the assignment of the wrong number of seats with

[8]As an example, given the decimal number 31,425, its integer part is 31 and its decimal part is 0,452.

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 I and II depicted below both comply with the text of Article 83 above:

Algorithm I

```
Consider the list of districts
for each district in the list:
        execute Step A
        execute Step B
        execute Step C + Step D
Check the need for seats adjustment
  ....
```

Algorithm II

```
Consider the list of districts
for each district in the list:
        execute Step A
for each district in the list:
        execute Step B
for each district in the list:
        execute Step C + Step D
Check the need for seats adjustment
  ....
```

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 (Silberschatz et al., 2012). 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 I are equivalent to lines 1–5 of Algorithm II. 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 I 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 II, 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 II always assign the expected number of seats. Interestingly, the format of Algorithm II 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 III given below.

Algorithm III

```
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   ....
```

Algorithm III 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 districts are considered, then provides deterministic outcomes. Thus Algorithm III, which seems to correspond to the undocumented vade mecum for ballot scrutineers, does not suffer from the indeterminacy problem highlighted in Algorithm I and II.

In conclusion, since all three algorithms are compatible with the legal text, we leave to legal scholars the open legal question 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 (*e.g.* Lodato et al., 2014; Ricca & Scozzari, 2019, in the case of Italian election law). We think that in this context it is fruitful to draw attention to the distinction between the *mathematical*, the *algorithmic* and the *software*-based approach, which are sometimes 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[9] is characterized by simultaneously defining a formula and the procedure/algorithm to compute it. Moreover, the algorithm focuses on the logical sequence of steps leading to the solution of a problem, while the software rigidly instructs a machine about how to execute those steps. Accordingly, the three approaches adopt three different languages: the mathematical-symbolic language, the algorithmic language—intended as the broad set of specification languages—and the programming languages. Both mathematics and software's code are unambiguous[10], while the algorithm can adopt any sufficiently precise language without conforming to strict syntactic rules. Therefore, the algorithmic description keeps part of the fruitful flexibility provided by the natural language, and at the same time scientifically supports the reasoning on its content,

[9]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).

[10]Meaning that each symbol and code instruction have a unique interpretation. In particular, the machine always knows how to interpret (*i.e.*, to execute) the code, which is different from a software being fully intelligible for programmers.

thus acting as an interface between the textual description of a law and its formal modelling through either maths of software.

To exemplify, Figs. 1 and 2 illustrate the use of different languages –textual, mathematical, algorithmic (in a human-friendly style and in pseudo-code), and Python software– 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 item c[11]). 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 distribution of votes corresponding to those formulas. The logical algorithm can then be refined into a more technically precise language, called *pseudo-code*, which directly leads to an executable Python software.

We omit the detailed explanation of the content of Figs. 1 and 2, but we call attention to the intermediate status of the algorithmic description, between the textual description and the full encoding either in maths or in software. As an example, what Article 77 textually defines as "distribution quotient", mathematically corresponds to the formula $(\Sigma_{p\in[1,k]}\ \underline{v}_p)/v_{cl}$, which is better understood by reading its algorithmic variant given in line 1 of the logical algorithm in Fig. 1. Similarly, the text "the votes remaining to be allocated" corresponds to the number N given by the right-most mathematical formula, but it is better clarified by the algorithmic instruction in line 7 that explains how to actually compute that number.

The transition form algorithm to software brings in another useful insight: notice that the lines 1 and 7 of the logical algorithm in Fig. 1 have a direct correspondence with lines 1 and 7 of both the pseudo-code[12] and the Python software in Fig. 2, the only difference is the use of data structures with related operations.[13] In computer science it is well known that the definition of an algorithm is heavily influenced

[11] *"[...] 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 piu' 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 cosi' 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."[Article 77 item c]*

[12] The pseudo-code command `X: = exp` stands for computing the value of the expression `exp` and assigning, *i.e.*, naming, this value to `X`. The symbol # denotes *comments*, providing for human-readable explanations.

[13] The pseudo-code is based on three data structures: `Votes`, `I`, and `R`. They are key-value mappings, that associate to each party `p` a specific value denoted by `Votes [p]`, `I [p]`, and `R [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) (Cormen et al., 2009). The same holds for the software, where data structures are renamed as `votes`, `integ`, and `rest` to comply with Python's syntax rules.

Legal Text:

[Article 77 item c] "[...] determines the uninominal constituency's votes of each party. This number is the sum of the valid votes obtained by the party itself in the individual electoral sections of the uninominal constituency and the votes cast in favour of the candidates linked to more than one party in a coalition as referred to in Article 58, third paragraph, last sentence, attributed to the party with the following operations: the Office shall divide the total of the valid votes obtained by all the parties in the coalition in the uninominal constituency by the number of votes cast for the coalition's candidates in the uninominal constituencies, obtaining the distribution quotient. It then divides the total number of valid votes obtained by each party by this quotient. The integer part of the quotient obtained in this way represents the number of votes to be allocated to each party; the votes remaining to be allocated are assigned to the parties for which the latter divisions have the largest remainders, in descending order of the remainders.

Mathematics:

For a coalition containing k parties, the votes assigned to each party p in the coalition are:

$$v_p = \underline{v}_p + \left\lfloor \frac{\underline{v}_p}{\Sigma_{p\in[1,k]}\,\underline{v}_p} \cdot v_{cl} \right\rfloor + \delta_{N,p} \quad \text{where} \quad N = v_{cl} - \Sigma_{p\in[1,k]} \left\lfloor \frac{\underline{v}_p}{\Sigma_{p\in[1,k]}\,\underline{v}_p} \cdot v_{cl} \right\rfloor$$

(The votes assigned to the party are those that has been directly collected by the party ($\underline{v}_p$), plus a proportion of the votes collected by the coalition (v_{cl}), 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 $\delta_{N,p}$).)

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 by subtracting from the total votes of the coalition
    the sum of the integer parts $I[p]$ of every party, which have already been distributed
8   Take the list $L$ of parties, sorted from that with largest reminder to that with the smaller reminder
9   while $N > 0$:
10      Assign one additional vote to the next party in the ordered list $L$, and decrease $N$
```

Fig. 1 Distribution of the coalition's votes among its parties

by the choice of the underlying data model (Cormen et al., 2009). The most relevant differences are in terms of the algorithm's efficiency, *i.e.*, the speed of its execution; however, even in simple situations, choosing how to organize information into specific data structures orients the algorithmic thinking, and affects the design of the actual procedure to be executed (Cormen et al., 2009). For instance, in the case of a procedure working on a collection of items, a main choice is that between working with either an ordered or an unordered list. 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 Sect. 2). Reasoning on the ordering of such a list has been decisive to find the indeterminacy issue of Algorithm I and Algorithm II, as well as to match the format of Algorithm II with the former Italian election law (Law n. 277/1993) when a specific district ordering is adopted.

Algorithm in pseudo-code:

```
#   Algorithm for distributing the votes of a coalition (named v_cl) between its parties
#   Data structures: Votes maps each party p to its direct votes Votes[p]. I and R are initially empty mappings
Q := (sum_values(Votes))/v_cl     # Compute the distribution quotient
for each party p in the coalition:
    I[p]:=(Votes[p]/Q).integer    # Compute the integer part
    R[p]:=(Votes[p]/Q).remainder # Compute the reminder
for each party p in the coalition:
    Votes[p]+=I[p]    # Add to its direct votes a number equal to its integer part
N := v_cl - sum_values(I)    # Compute the number of votes remaining to be distributed
L:=sort_by_values(R)    # Take the list of parties sorted from the largest reminder to the smaller
while N>0
    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 v_cl be the coalition's votes, and let votes be a key-value map from a party to its direct votes
def distribution(v_cl,votes):
   q = sum(votes.values())/v_cl #distribution quotient
   integ = {} ; rest ={}
   for p in votes.keys():
      (rest[p],integ[p])=math.modf(votes[p]/q) #integer part and reminder
   for p in votes.keys():
      votes[p]=votes[p]+integ[p] #add votes due to the integer part
   n = v_cl - sum(integ.values()) #number of votes remaining to be distributed
   l=list(sort_by_values(rest).keys()) #sort parties according to remainders
  i=0
   while n > 0:
      votes[l[i]]=votes[l[i]]+1 #assign 1 vote to the next party in the list
      n=n-1
      i = i+1
   return votes
```

Fig. 2 From Algorithm to Software

In summary, we have seen that reasoning at the abstraction level of the algorithmic description 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 Sect. 2. Additionally, besides corroborating the reasoning about the *legal content*, 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 III, the legislators 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 by writing a textual description of Algorithm III according to the style of the legal text, as illustrated in Appendix 2.

3.1 From Code Normativity to Algorithmic Normativity

Having stressed the difference between algorithm and software, we now examine that difference in the legal context. The power of using software in the legal field has been increasingly investigated; adopting the classification put forward by Hildebrandt (2018) and the Cohubicol project (2019), a major distinction is the usage of software that either operates according to an explicitly programmed decision logic, or that is informed by the data on which it has been trained through machine learning algorithms. Here we focus on the first case, called *code-driven law*, that aims to translate or to directly write legislation into computer code. Recent applications ranges from 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 Palmirani & Vitali, 2011; Mohun & Roberts, 2020; Corsius et al., 2021; Crafa, 2022; Huttner & Merigoux, 2022; Casolari et al., 2023).

In Sect. 2 we have already mentioned the limits of the code-driven enforcement of rules hardwired 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 by legal scholars. Indeed, while the software code leads to a rigid closure to a (possibly fallacious) implementation, the algorithmic language guarantees a more flexible description of legal provisions, that also admits the usage of domain-specific terms (like a coalition of parties or a multi-nominal constituency) that are not forced into specific programming or data constructs. Therefore, the algorithmic normativity, being based on a language more precise than text and more flexible and human-readable than code, increases the transparency and keeps open the double play of contestability and predictability of law.

Finally, we remark that even the software can be used flexibly as a *tool for reasoning* about legal content, instead of simply encoding that content. Indeed, the fact that a legal procedure can be implemented in an executable software leaves room for reasoning about the legal content of that procedure 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 Fig. 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 Sect. 2 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 (for a discussion of the risks accompanying the use of electronic voting in political elections, see the motion approved in 2021 by the Italian IT community represented by the two academic associations GRIN and GII[14]). 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 algorithmic 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 interoperate 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.

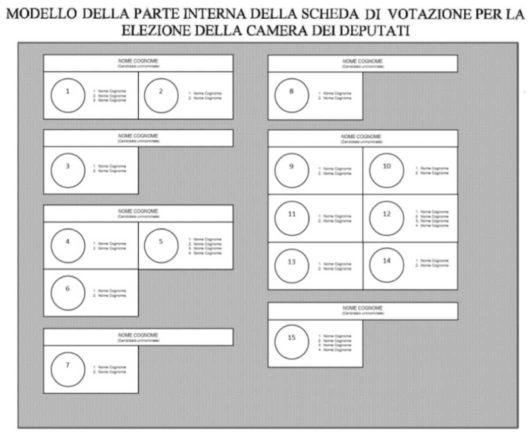
In this work we focused on legal drafting and the difficulty of satisfactorily describing complex legal procedures. In this sense it is related to the field of legislative techniques, and the advantages that technology can bring to legal drafting. The case of the Italian legal system is particularly awkward, since the Italian legal language is very convoluted, redundant and often difficult to parse (Mercatali, 2011). National recommendations for the drafting of regulatory texts also contain the indication of the use of software tools for the automatic analysis of the quality of the text, providing indications on the comprehensibility of texts by means of data and statistical indices (Consiglio Regione FVG 2007; Zaccaria, 2011; Tecniche Normative, 2023). These recommendations contributes not only to making legal texts more accessible, but also to make legislative design and drafting more responsive to the needs of computer applications so to support web-based access and document retrieval. Instead, in this article we suggest using the algorithmic

[14] http://www.grin-informatica.it/opencms/export/sites/default/grin/files/mozione-voto-elettronico-finale-per-pubblicazione.pdf.

language 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. In general, we envision the adoption of a legal drafting that relies on a richer set of technologies that goes beyond text, to include algorithmic pseudo-code, diagrams, images and other specification languages, productively interoperating by choosing each time the most suitable language to express a specific content and a specific type of normativity.

To conclude, in this work we focused just on algorithmic specification, 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 future investigation.

Appendix 1: Ballot Paper


MODELLO DELLA PARTE INTERNA DELLA SCHEDA DI VOTAZIONE PER LA ELEZIONE DELLA CAMERA DEI DEPUTATI
NOME COGNOME (Candidato uninominale)
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
1. Nome Cognome
2. Nome Cognome

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)

Appendix 2: 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 Section2.1), and it is enough to exclude the two problematic and non-deterministic interpretations corresponding to Algorithm A and B:

The national central office, [....]
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).
To this end it shall determine the number of seats to be allocated in each district by [...].

~~It then~~ In every district, it divides the sum of the [...] obtaining the district's electoral quotient.[...]

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.

~~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.

~~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 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].

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 [...].

The Office shall then ascertain whether the number of seats [...]. If not, it shall carry out [...] [15]

[15]*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. 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 cosi' il quoziente elettorale circoscrizionale. Nell'effettuare tale divisione non tiene conto dell'eventuale parte frazionaria del quoziente cosi' ottenuto. Divide poi la cifra elettorale circoscrizionale di ciascuna coalizione di liste o singola lista per il quoziente elettorale circoscrizionale, ottenendo cosi' 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.~~ Assegna quindi a ciascuna coalizione di liste o singola lista il numero dei 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 i seggi che rimangono ancora da attribuire sono rispettivamente assegnati alle coalizioni di liste o singole liste per le quali ~~queste ultime divisioni hanno~~ il calcolo dei quozienti di attribuzione ha dato le maggiori parti decimali e, in caso di parita', alle coalizioni di liste o singole liste che hanno conseguito la maggiore cifra elettorale nazionale; a paritaa' di quest'ultima si procede a sorteggio. ~~Esclude dall'attribuzione di cui al periodo precedente le coalizioni di liste o singole liste alle quali e' stato gia' 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 in tutte le circoscrizioni a ciascuna coalizione di liste o singola lista corrisponda al numero di seggi determinato ai sensi della lettera f). In caso negativo, [...].*

Acknowledgements The author acknowledges the support of CINI National Laboratory “Informatics & Society”, and dedicates this work to Presidente della Repubblica Italiana, Sergio Mattarella.

Funding This work was supported by Università degli Studi di Padova. Open access funding provided by Università degli Studi di Padova within the CRUI-CARE Agreement.

Data Availability All data generated or analysed during this study are included in this article.

Declarations

Consent Not applicable.

Materials and Method Not applicable.

Competing Interests On behalf of all authors, the corresponding author states that there is no conflict of interest, nor financial or non-financial interests that are directly or indirectly related to the work submitted for publication.

References

Aguilera, M. (2019). Pictures, content, and normativity: The semantic of graphic rules. *Phenomenology and Mind*, *17*, 136–149. https://doi.org/10.3128/pam-8032

Ansa. (2022). Elezioni, gli errori del Viminale e il riconteggio dei voti: Cos’è successo. Retrieved September 29, 2022, form https://tg24.sky.it/politica/approfondimenti/riconteggio-voti-elezioni-errori-viminale#00

Artosi, A. (2021). Technical Normativity. In S. Chiodo & V. Schiaffonati (Eds.), *Italian philosophy of technology, philosophy of engineering and technology* (pp. 149–160). Springer Nature. https://doi.org/10.1007/978-3-030-54522-2_7

Biagioli, C., Mercatali, P., & Sartor, G. (1993). *Elementi di legimatica*. Cedam

Borruso, R., Di Giorgi, R. M., Mattioli, L., & Ragona, M. (2004). *L’informatica del diritto*. Giuffrè

Camera dei Deputati, XVIII Legislatura, Giunta delle Elezioni (2020). Relazione nazionale sull’attribuzione dei seggi nei collegi plurinominali. Resoconto della seduta del 14 luglio 2020. Retrieved from https://documenti.camera.it/leg18/resoconti/commissioni/bollettini/pdf/2020/07/14/leg.18.bol0408.data20200714.com16.pdf

Camera dei Deputati, XVIII LegislaturaServizio Studi (2022). Il sistema di elezione del parlamento nazionale. L’evoluzione normativa e la disciplina vigente. Retrieved from https://documenti.camera.it/leg18/dossier/pdf/AC0337.pdf

Casolari, F., Taddeo, M., Turillazzi, A., & Floridi, L. (2023). How to improve smart contracts in the European Union Data Act. *Digital Society*, *2*(9). https://doi.org/10.1007/s44206-023-00038-2

CoHuBiCoL research project - counting as a human being in the era of computational law (2019). Three types of normativity. Funded by the European Research Council (ERC) under the HORIZON2020 Excellence of Science program ERC-2017-ADG No 788734 (2019-2024). Retrieved form https://www.cohubicol.com/about/three-types-of-normativity/

Consiglio Regione FVG (2007) Regole e suggerimenti per la redazione dei testi normativi - manuale per le Regioni promosso dalla Conferenza dei Presidenti delle Assemblee legislative delle Regioni e delle Province autonome con il supporto scientifico dell'Osservatorio legislativo interregionale. Online document, Retrieved from https://www.consiglio.regione.fvg.it/pagineinterne/Portale/drafting/drafting.pdf

Contissa, G., Godano, F., & Sartor, G. (2021). Computation, Cybernetics and the Law at the Origins of Legal Informatics. In S. Chiodo & V. Schiaffonati (Eds.), *Italian philosophy of technology, philosophy of engineering and technology* (pp. 91–110). Springer Nature. https://doi.org/10.1007/978-3-030-54522-2_7

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms*. MIT Press

Corsius, M., Hoppenbrouwers, S., Lokin, M., Baars, E., Sangers-Van Cappellen, G., & Wilmont, I. (2021). RegelSpraak: A CNL for executable tax rules specification. In *Proceedings of the seventh international workshop on controlled natural language (CNL 2020/21)*. Retrieved from https://aclanthology.org/2021.cnl-1.6

Crafa, S. (2022). From legal contracts to legal calculi: The code-driven normativity. In *Proceedings international workshop on expressiveness in concurrency and structural operational semantics, EXPRESS/SOS* (Vol. 368, pp. 23–42). EPTCS. https://doi.org/10.4204/EPTCS.368.2

Crafa, S. (2023). *Sull'indeterminatezza della procedura di attribuzione dei seggi nella legge elettorale italiana*. Technical report. https://doi.org/10.5281/zenodo.8038610

De Filippi, P., & Hassan, S. (2016). Blockchain technology as a regulatory technology: From code is law to law is code. *First Monday*, *21*(12), https://ssrn.com/abstract=3097430

Diver, L., Duarte, T., Gori, G., van den Hoven, E., & Hildebrandt, M. (2023). Research study on Text-Driven Law (Brussels 2023), funded by the ERC Advanced Grant 'Counting as a Human Being in the Era of Computational Law' (COHUBICOL project). Retrieved from https://publications.cohubicol.com/research-studies/text-driven-law/

Dudek, M. (2015). Why are words not enough? Or a few remarks on traffic signs. In M. Araszkiewicz, P. Banaś, T. Gizbert-Studnicki, K. Płeszka (Ed.), *Problems of normativity, rules and rule-following* (Vol. 111 of Law and Philosophy Library). https://doi.org/10.1007/978-3-319-09375-8_27

Hildebrandt, M. (2018). Algorithmic regulation and the rule of law. *Philosophical Transactions of the Royal Society A*, *376*, 20170355. https://doi.org/10.1098/rsta.2017.0355

Huttner, L., & Merigoux, D. (2022). Catala: Moving towards the future of legal expert systems. *Artificial Intelligence and Law*. https://doi.org/10.1007/s10506-022-09328-5

Lessig, L. (1999). *Code and other laws of cyberspace*. Basic Books, Inc

Lodato, G., Pajno, S., & Scaccia, G. (2014). Quanto può essere distorsivo il premio di maggioranza? Considerazioni costituzionalistico-matematiche a partire dalla sentenza n.1 del 2014. *Federalismi.it* n.9

Lorini, G., & Moroni, S. (2020). How to make norms with drawings: An investigation of normativity beyond the realm of words. *Semiotica*, *233*, 55–76. https://doi.org/10.1515/sem-2018-0062

Meldman, J. A., & Holt, A. W. (1971). Petri nets and legal systems. *Jurimetrics Journal*, *12*(2), 65–75. http://www.jstor.org/stable/29761228

Mercatali, P. (2011). Linguistica, informatica, scienza e tecniche della comunicazione nella formazione del giurista. In R. Zaccaria (Ed.), *La buona scrittura delle leggi*. Camera dei Deputati. Retrieved form https://www.camera.it/temiap/temi16/La_buona_scrittura_delle_leggi.pdf

Mohun, J., & Roberts, A.(2020). Cracking the code: Rulemaking for humans and machines. Organisation for Economic Co-operation and Development (OECD) Working Papers on Public Governance No. 42. https://doi.org/10.1787/3afe6ba5-en

Nardelli, E. (2019). Do we really need computational thinking? *Communication of the ACM*, *62*(2), 32–35. https://doi.org/10.1145/3231587

Palmirani, M., & Vitali, F. (2011). Akoma-Ntoso for legal documents. In *Legislative XML for the semantic web. Law, governance and technology series* (Vol. 4). Springer. https://doi.org/10.1007/978-94-007-1887-6_6

Pietrangelo, M. (2023). Codice di drafting Libro VI - La conoscibilità della legge per via informatica e telematica. Website *Tecniche normative - il portale del drafting normativo*. Retrieved June, 2023, from https://www.tecnichenormative.it

Ricca, F., & Scozzari, A. (2019). L'algoritmo elettorale tra rappresentanza politica e rappresentanza territoriale. Una nuova procedura di allocazione proporzionale dei seggi. Camera dei Deputati, XVIII Legislatura, Servizio Studi. Retrieved from https://www.camera.it/temiap/2019/06/25/OCD177-4075.pdf

Rosengrün, S. (2022). Why AI is a threat to the rule of law. *Digital Society*, *1*, 10. https://doi.org/10.1007/s44206-022-00011-5

Sala, A. (2022). Riconteggio voti, cambiano gli eletti. Non solo Bossi: Chi sono i deputati "ripescati" (e gli esclusi). Corriere della Sera. Retrieved September 28, 2022, form https://www.corriere.it/elezioni/22_settembre_28/riconteggio-voti-cambiano-eletti-non-solo-bossi-tutti-deputati-ripescati-esclusi-9754440c-3f4f-11ed-b6e3-34338f1c69a0.shtml

Sergot, M. J. (1991). The representation of law in computer programs: A survey and comparison. In *Knowledge based systems and legal applications* (Vol. 36, pp. 3–67). Academic Press. https://doi.org/10.1016/B978-0-12-086441-6.50006-4

Silberschatz, A., Galvin, P. B., & Gagne, G. (2012). *Operating system concepts*. John Wiley & Sons, Inc

Siniscalchi, G. (2019). Deontic visual signs. Between normative force and constitutive power. *Phenomenology and Mind*, *17*, 150–159. https://doi.org/10.13128/pam-8033

Tecniche Normative. (2023). Il Portale del Drafting normativo. Retrieved June, 2023, from https://www.tecnichenormative.it

Zaccaria, R. (2011). La buona scrittura delle leggi. Camera dei Deputati, 2011. Retrieved form https://www.camera.it/temiap/temi16/La_buona_scrittura_delle_leggi.pdf