3

A Legal Philosophy Perspective: Code is *Less* than Law

> The alternative to legality is not anarchism, it is legalism . . . '[N]ot thinking about it', if left to its own devices, tends to take over the entire social world, or at least cyberspace.[1]

In the previous chapter, I used design theory and the philosophy of technology to describe how code constitutes and regulates end-user behaviour. Demonstrating the material directness of code-based regulation was the first task in setting out the theoretical grounding of digisprudence. This chapter complements that analysis, turning towards legal philosophy to develop an account of 'computational legalism'. This idea is borne of the parallel between code's ruleishness – its reliance on strict, binary logic instead of interpretable standards – and its conceptual equivalent in the juridical realm, known as *legalism*. The latter is a perspective that disavows the holistic interpretation of legal norms, instead requiring that citizens merely follow rules as they are presented to them, without enquiring as to their efficacy or their legitimacy beyond the question of where they came from. Code's characteristics exemplify a particularly strong form of 'legalism', and therein lies the problem of unlegitimated code-based regulation and the claim that it is 'less' than law. As Wintgens puts it, '[r]ule creation is a matter of choice, and this choice is legitimated because it is based upon the democratic character of the regulating process.'[2] In very few cases are such aspirations reflected in the production of code. End-users are 'induced, habituated and, if necessary, compelled, to accept the norms of commercialized cyberspace',[3] all of which taking place

[1] Z Bańkowski and B Schafer, 'Double-click justice: Legalism in the computer age' (2007) 1 *Legisprudence* 31, 47–8.

[2] L Wintgens, *Legisprudence: A New Theoretical Approach to Legislation* (Hart 2002) 2.

[3] G Longford, 'Pedagogies of digital citizenship and the politics of code' (2005) 9 *Techné: Research in Philosophy and Technology* 68, 71.

outwith democratic debate and legal processes of interpretation, contest, and remediation. As Longford suggests,

> whereas the terms and conditions of political citizenship in liberal democratic states are, relatively speaking, subject to free, open and transparent deliberation and negotiation, the codes governing the citizen in the digital era are invisible and opaque, thanks to certain features of the technologies themselves and to the proprietary nature of many of the codes increasingly mediating our lives.[4]

To be clear, my aim is not to suggest that designers harbour a legalistic ideology; instead, I want to demonstrate how aspects of a legalistic mentality are closely reproduced in the material architectures of code, with the result that the ideological 'ought' of legalism becomes the technological 'is' of code. If we proceed from the starting point that legalism is an undesirable thing in a democracy, then the mechanisms for mitigating it in the traditional legal sphere might also have an ameliorating effect in the analogous context of code-based 'legislation'.

Drawing a parallel between legalism in the contexts of legal and technological normativity sets the stage for an analysis of the ways in which its mitigation in the former can be imported into the sphere of the latter. The aim is to investigate the 'new forms of interaction' that Bańkowski and Schafer suggest are necessary to 'promote the benefits of legality, and to prevent the disadvantages of legalism' in the code context.[5]

This chapter first sets out the notion of legalism, before demonstrating how it is that code, when it operates as an enforcer, mediator, and constitutor of behavioural reality, can be a particularly extreme incarnation of this ideological perspective. Wintgens suggests that 'long decades of legalism in legal reasoning [have meant that] the dominant views in legal theory . . . have barred the way for questioning the position of the legislator'.[6] In the parallel between code and law that I aim to construct, questioning the position of the designer *qua* legislator becomes a pressing concern and precisely what digisprudence seeks to do.

3.1 What is Legalism?

There are conflicting conceptions of legalism in the literature, it being occasionally confused with related concepts such as legality and the rule of law.[7]

[4] Ibid.

[5] Bańkowski and Schafer (n 1) 46.

[6] Wintgens, *Legisprudence: A New Theoretical Approach to Legislation* (n 2) 2.

[7] Bańkowski and Schafer note for example that legalism is 'often confused with legality, an altogether more reflexive and rational concept'. See Bańkowski and Schafer (n 1) 31–2.

MacCormick contended early on that legalism is 'a prerequisite of free government',[8] but this seemed for him to amount essentially to the ex post doctrine of *nulla poena sine lege* ('acts of government however desirable teleologically must be subordinated to respect for rules and rights'[9]), and is therefore different from the 'stronger' version of legalism with which I am concerned. Indeed, MacCormick explicitly distinguishes between that conception of legalism and the stronger conception identified in the literature.[10] MacCormick's conception of legalism is akin to Wintgens's idea of 'weak' legalism, which forms a part, rather than the whole, of the legisprudential conception of legitimacy that I draw on in Chapter 4. At any rate, this intermediate position, later adopted by MacCormick himself in work with Bańkowski, views some measure of legalistic rule-following as a necessary element, but not the whole, of a functioning legal order. This accords with the normative position developed here, where rules are an appropriate basis for regulating behaviour, but the process of their development is constrained so as not to be arbitrary. This idea is embodied in the Greek myth of Odysseus, in which the eponymous captain orders his crew both to tie him to the mast of his ship, so he cannot succumb to the enticement of the Sirens, and to block their ears with beeswax, so his orders to untie him in the face of that temptation will fall on deaf ears. The metaphor is of a sovereign limiting itself in order to avoid the temptation of iniquity (being bound to the mast) whilst also submitting to checks and balances that prevent that power being exercised should the sovereign's scruples change (the wax in the ears of the crew). Various scholars have considered the myth in their discussions of constitutionalism in the computational context.[11] Hildebrandt contrasts legality and legalism thus:

> Legality, in this sense, refers to justice (proportionality), to legal certainty (the legal ground in positive law, with the necessary safeguards) and purposiveness or expediency (the legitimate aim of the intervention, the requirement of effective remedies). Legalism, instead, reduces all this to the correctly enacted legal ground, which may or may not offer any protection, leaving the subject of government interventions dependent on a rule *by* law instead of the Rule of Law. Even if the sovereign that rules *by* law is the

[8] N MacCormick, 'The ethics of legalism' (1989) 2 *Ratio Juris* 184.

[9] Ibid. 184.

[10] Ibid.

[11] See for example M Hildebrandt, *Smart Technologies and the End(s) of Law: Novel Entanglements of Law and Technology* (Edward Elgar Publishing 2015) 156; L Lessig, *Code: Version 2.0* (Basic Books 2006) 313–14.

> *nation* or the *Parliament*, legalism leaves individual subjects without effective remedies against arbitrary rule.[12]

One can see here the implication in the concept of legality that the rules promulgated must be designed to reflect certain ideals (proportionality, safeguards, the substantive legitimacy of the norm itself). Legalism, by contrast, is concerned only that the rule has been promulgated by a legitimate institution, and cares not what its content or substantive effects are.

Hildebrandt's characterisation of legalism, where there is an absence of protection against arbitrary rule, matches the stronger variant of the concept, described by Shklar as 'an ethical attitude that holds moral conduct to be a matter of rule following, and moral relationships to consist of duties and rights determined by rules'.[13] This deontological outlook – termed the 'morality of duty' by Fuller[14] – of course has a long pedigree in moral philosophy, exemplified in Kant's categorical imperative, he being the 'high priest for a rule based morality'.[15] Such an approach has moral force because it results in a normalisation and systematisation of behaviour across society, which in turn begets the kind of behavioural predictability that has been argued is a desirable goal in the development of a stable (capitalist[16]) society.[17]

Heteronomy, the condition of being dominated by an external sovereign, is antithetical to aspirations of reasoned interpretation and action, and thus to autonomy.[18] It is exemplified in what Wintgens calls the 'strong' variant of legalism, which he describes as a normative 'strategy', used historically to avoid contingency and promote legal certainty.[19] On the other hand, it is also

[12] M Hildebrandt, 'Radbruch's Rechtsstaat and Schmitt's legal order: Legalism, legality, and the institution of law' (2015) 2 *Critical Analysis of Law* 42, 56 (emphasis supplied). For the antinomian conception of law that Hildebrandt draws on, see G Radbruch, 'Legal philosophy' in K Wilk (ed.), *The Legal Philosophies of Lask, Radbruch, and Dabin* (Harvard University Press 1950).

[13] JN Shklar, *Legalism* (Harvard University Press 1964) 1.

[14] LL Fuller, *The Morality of Law* (Yale University Press 1977) chapter 1.

[15] Z Bańkowski and N MacCormick, 'Legality without legalism' in W Krawietz et al. (eds), *The Reasonable as Rational? On Legal Argumentation and Justification; Festschrift for Aulis Aarnio* (Duncker & Humblot 2000) 183. See also Z Bańkowski, 'Don't think about it: Legalism and legality' in MM Karlsson, Ó Páll Jónsson and EM Brynjarsdóttir (eds), *Rechtstheorie: Zeitschrift für Logik, Methodenlehre, Kybernetik und Soziologie des Rechts* (Duncker & Humblot 1993) 45; Bańkowski and Schafer (n 1) 33.

[16] Bańkowski (n 15) 48. See also L Wintgens, 'The rational legislator revisited. Bounded rationality and legisprudence' in *The Rationality and Justification of Legislation* (Springer 2013) 4.

[17] Shklar (n 13) 64.

[18] See, for example, Bańkowski and Schafer (n 1); Bańkowski and MacCormick (n 15) 194; Bańkowski (n 15) 56; MacCormick, 'The ethics of legalism' (n 8) 192.

[19] L Wintgens, *Legisprudence: Practical Reason in Legislation* (Routledge 2012) 159.

often thought that some measure of legalism (that is, respect for rules *qua* rules, or 'law as law') is necessary for a society to operate well, and indeed that legalism should be understood normatively not as being in opposition to legality but rather as a necessary element of it.[20] This is 'weak legalism', a viewpoint central to legisprudence, according to which rules remain the proper mechanism for regulating action, but the potential for their arbitrary definition is simultaneously constrained. Strong legalism undermines legality, whereas weak legalism is a necessary (although insufficient) component of it. Legality, properly understood, requires a complementary combination of adherence to rules with thoughtful interpretation of what is being commanded by the rule, with the appropriate response varying depending on the particular circumstances.[21] For Bańkowski and Schafer, it is sometimes appropriate for citizens mindlessly to follow a rule – to 'act like automata' – while at other times it is necessary to act autonomously, considering what the rule asks of us before deciding how to act. Strong legalism implies only the former approach, whereas weak legalism is the rule-based element of the broader concept of legality.

This strong conception of legalism is extremely relevant to a descriptive analysis of code, because as we shall see the latter not only exemplifies its characteristics but indeed amplifies them far beyond what is envisaged in most of the legal literature: '[code's] unrestricted anarchism in the absence of the state has indeed resulted in the most absolute form of legalism possible'.[22]

Subsequent references in this chapter to legalism are to this strong variant, unless otherwise specified. In the rest of this section I set out the theory of legalism and its approach to law-making in more detail, setting the stage for a comparison between it and code in the latter part of the chapter.

(a) Solipsism and Positivism

Legalism is rooted in a solipsistic view of law as a system of rules and practices that operates separately from the societal contexts within which it is embedded and which it serves. Law is a 'clean' system, 'self contained and autogenerative', subsuming the outputs of the 'dirty business' of politics (that is, legislation) and applying them according to its own *sui generis* processes, institutions, and vocabulary.[23] Already we have a glimpse of the parallel with code.

[20] Bańkowski (n 15); MacCormick, 'The ethics of legalism' (n 8).

[21] Bańkowski and Schafer (n 1) 48. I will return to legality in Chapter 4.

[22] Ibid.

[23] Bańkowski (n 15) 46. For a nineteenth-century expression of this perspective in Scots constitutional law, see *Edinburgh & Dalkeith Railway Company v Wauchope* (1842) 1 Bell 278,

'Legislation is a matter of politics, and politics is a matter of choice', and so the 'truthiness' of law requires that it remain separate from anything so contingent.[24] Law is viewed as a scientific practice that identifies and works with 'truths', which are the product of sovereign legislators. The content of those truths is not to be questioned: from the perspective of the law and legal practice the truth 'just is', it is handed down from the political realm where it is the sole province of the legislator to debate the substance of the norm. The jurist has no valid interest in what goes on there; politics is about choice, and therefore it does not deal in 'truth' because competing choices can never be objects of true knowledge.[25] Once the legislature chooses between the various possible options and crystallises one of them into a law, it becomes an item of 'true' knowledge within the science of law, whose objective (extra-legal) quality is irrelevant to the legal 'scientists' who, from that point onward, take it as a datum for application within their field.

In this way, legal thinking becomes 'fenced off' from 'all contact with the rest of historical thought and experience'.[26] The result is a positivistic view that the law is 'just there', and it is not the task of citizens or practitioners to enquire as to how it got 'there'.[27] What matters is whether it is a valid law, and not whether we agree with its substance and relevance to the present circumstances. The 'truth' ('is-ness') of a given legal norm derives from the validity of its creation vis-à-vis authorised actors and processes, and the question of whether its substance is desirable or not (its 'ought-ness') is properly to be viewed as separate from this.[28] The preoccupation with the ex post examination of what should and should not be considered 'law' is of course a core characteristic of Anglo-American analytical legal positivism. Strong legalism is connected with this outlook in its drive to classify rules according to those that are internal to the legal system and those that are external.[29]

---

per Lord Campbell: 'All that a court of justice can look to is the parliamentary roll; they see that an act has passed both Houses of Parliament, and that it has received the royal assent, and no court of justice can enquire into the manner in which it was introduced into parliament, what was done previously to its being introduced, or what passed in parliament during the various stages of its progress through both Houses of Parliament.'

[24] L Wintgens, 'Legisprudence as a new theory of legislation' (2006) 19 *Ratio Juris* 1, 5.

[25] Ibid.

[26] Shklar (n 13) 3. See also Bańkowski and MacCormick (n 15) 182, and Fuller, in his response to Hart in their classic debate: LL Fuller, 'Positivism and fidelity to law: A reply to Professor Hart' (1958) 71 *Harvard Law Review* 630, 635.

[27] Bańkowski (n 15).

[28] Bańkowski and MacCormick (n 15) 186.

[29] L Wintgens, 'Legislation as an object of study of legal theory: Legisprudence' in *Legisprudence: A New Theoretical Approach to Legislation* (Hart 2002) 20.

Thus, from a legalistic perspective, 'what ought to be done is confined to the knowledge of the rules that contain rights and duties. Following rules is a matter of knowledge, while their enforcement is a matter of application.'[30] Legal practitioners take that knowledge, provided from somewhere 'out there', and use it as a tool to achieve a given legal aim. Their practice is 'neutral' as to the substance of these materials (rules), and they become technicians whose task it is to manipulate those rules according to the mechanisms of legal reasoning.[31]

(b) Legalism According to Legisprudence

Strong legalism is thus concerned with the application rather than the design of rules.[32] In his comprehensive historical discussion of the origins of legalism, Wintgens discusses the theoretical mechanisms of legitimation in both natural law and analytical legal positivism, before identifying a set of specific characteristics of which the phenomenon of strong legalism is a 'conjugation', namely representationalism, a-temporality, concealed instrumentalism, *etatism* (the belief that the only true source of law is the state), and the scientific study of law.[33]

The orthodox source of a rule's legitimacy differs depending on the source of sovereignty – broadly, natural law or the social contract: respectively, that source is either a transcendent set of natural law norms, or a social contract which founds a sovereign law-making institution. In the case of a natural law perspective, this is because the source of its substantive content is the 'background' knowledge of natural law principles which are inherently true: such representational laws are 'a concretisation of natural law, or reflect a natural law conception that in its turn legitimises positive law'.[34] In the case of the sovereign, this is so because the social contract legitimises such pronouncements as a descendent of some original founding contractual act of the people that set up the institution to represent them[35] (perhaps a document with constitutional status, although the social contract can also be a hypothetical moment rather than a real instrument). That the state defines what is legal is in itself enough to legitimise the substance of the legal norms it chooses to

[30] Wintgens, 'Legisprudence as a new theory of legislation' (n 24) 5.

[31] Bańkowski and Schafer (n 1) 34.

[32] Wintgens, *Legisprudence: Practical Reason in Legislation* (n 19) 139.

[33] Wintgens, 'Legisprudence as a new theory of legislation' (n 24) 5. For an in-depth philosophical and historical discussion of these characteristics, see Wintgens, *Legisprudence: Practical Reason in Legislation* (n 19) chapter 5.

[34] Wintgens, 'Legislation as an object of study of legal theory' (n 29) 10–11. See also Wintgens, *Legisprudence: Practical Reason in Legislation* (n 19) 147 *et seq*.

[35] Wintgens, *Legisprudence: Practical Reason in Legislation* (n 19) 195.

declare; in constituting the field of play (the legal system), the state legitimises *de facto* that which it consequently promulgates as the rules of the game. One can detect in this hierarchical idea of legitimacy Kelsen's 'Grundnorm' and Hart's 'rule of recognition'.[36] The outcome in either case is the same, namely a legitimated foundational source from which laws can be promulgated that are themselves *de facto* legitimate as a result of the *a priori* legitimacy of the source, and that therefore ought to be followed.[37]

Wintgens describes this as 'one-shot' legitimation, operating continually thereafter to validate prospectively any norms promulgated within the ex ante framework that it sets up. Drawing on Hobbes and Rousseau, this is what he describes as the 'proxy model' of legitimation,[38] according to which the initial legitimation of the external decision-maker permits it to act on behalf of the people (that is, as a proxy) from that moment onward, despite the inability of either the sovereign itself or the people it represents to foresee all the rules, or 'limitations on freedom', that will in the future be imposed. The citizen is given the imperative 'not to think about it'; she need only act in accordance with the rule as it is given to her,[39] since by virtue of those constitutive facts the pronouncement of the sovereign is 'imputed to [the citizenry], as if they were its author'.[40] This legalistic idea of minimal interpretation is connected with Hart's discussion of the 'core' and 'penumbra' in the meaning of individual words, the former being deemed to be settled and uncontested, and the latter being where controversies of interpretation arise.[41] I discuss this further in the section on rules in computational legalism below.

In the commercial realm, society essentially gives the designer of code a one-shot 'legitimation' of this kind when (1) we endow her with the plasticity of code to create a near-infinite number of conditions which enable and constrain behaviour through technological normativity, (2) we protect her privatised practices through (legally sanctioned) commercial secrecy and a general absence of scrutiny, and (3) we submit to the *sui generis* opacity of code. I discuss each of these characteristics in more detail below.

From a computational perspective, perhaps the most relevant element of strong legalism is representationalism. This is the view of law as a

[36] Wintgens explicitly identifies the connection between the Rousseauian 'act of will' that creates the social contract, and the Hartian system of a founding rule of recognition that is followed by emergence of a combination of primary and secondary rules (ibid. 170).

[37] Ibid. 196 *et seq*.

[38] Ibid. chapter 6 generally.

[39] Bańkowski (n 15).

[40] Wintgens, *Legisprudence: Practical Reason in Legislation* (n 19) 208.

[41] HLA Hart, 'Positivism and the separation of law and morals' (1958) 71 *Harvard Law Review* 593.

representation of reality, either through the latter's reproduction (in the case of natural laws that need to be given force by positive law) or its construction (in the case of laws based on a founding social contract).[42] Wintgens embarks on a rich analysis of the philosophies of Hobbes and Descartes, discussing the relationship between realism and nominalism and how these, despite their seemingly fundamental differences, can both result in the 'naturalisation of positive law', according to which law is deemed to be a representation of objective reality.[43] What the law states is therefore held to be true, either because natural laws are hypostatically true or because the social contract is true and therefore so too are the rules that are based upon it.

The salient connection with the computational context is that there representationalism is even more concrete than in the ideology of strong legalism: whereas proponents of the latter hold the belief that the rule presents reality, in the computational context this is much more than mere belief because, as we saw in Chapter 2, code does not just represent reality but actively constitutes it (or, at least, a part of it). I have already talked about constitutive versus regulative technological normativity, a theme I will return to below.

The next salient component of strong legalism is 'a-temporality', which flows from the belief in law as a representation of reality. Because either the social contract or natural law represents reality ex ante, anything that flows from them is believed also to be true, since they are the genuine and true foundation of political space. That foundation is a-temporal because it is believed to be the universal principled basis for public law, something that is valid independently of human recognition.[44] Contingent laws built upon this foundation are deemed to 'uncover' the general will, rather than proactively to reflect it (those who disagree with a particular legislative proposition are in error as to what the general will is, rather than in disagreement *per se*). The general will exists at all times, ready to be uncovered and recognised by contingent legislative acts. Thus, 'acts of will then take on the appearance of timelessness'.[45] The notion of the norms' timelessness connects with the immutable character of code discussed below, and the approach that Wintgens develops

[42] Wintgens, 'Legisprudence as a new theory of legislation' (n 24) 4. See also CM Campbell, 'Legal thought and juristic values' (1974) 1 *British Journal of Law and Society* 13.

[43] Wintgens, *Legisprudence: Practical Reason in Legislation* (n 19) 147–53. This metaphysical discussion centres around the distinction between the view of law as directly reproducing reality (realism), or the view that the creation of law constructs reality through the description of what is otherwise 'semantically empty' (nominalism). In either case, the result is a belief that law is representative of reality, of the world as it is.

[44] Ibid. 155.

[45] Ibid. 156–7.

to cope with timelessness in law-making – the legisprudential principle of temporality – also becomes relevant as a practical consideration.[46]

The final relevant aspect of strong legalism is 'concealed instrumentalism'. This is the idea of a 'veil of sovereignty', behind which the values or ends of the legislator are hidden. It is evidenced in textualism, where the policy goal of the legislator is less important than formalistic reasoning from the text. The fiction of timelessness previously discussed combines with this instrumentalism to form a strategy for converting the messy and contingent political into the clean and scientific legal.[47] This concealment finds its analogue in the legal and economic veils that protect code: enterprise is protected by trade secrets and anti-circumvention laws, while faith is placed in the market to curb any excesses.

As previously mentioned, whereas strong legalism is concerned primarily with the validity of a norm's source, legisprudence suggests that this is a necessary but insufficient condition for legitimacy. Not only must the sovereign be 'bound to the mast', but so too must it proactively legitimate the norms that it proposes. This is a type of validity that to an extent crosses the line between formal and substantive legitimacy – the substantive content of the norm is constrained according to certain formal qualities embodied in the principles of legisprudence. As Wintgens argues,

> [t]he basic idea of the rule of law or the *Rechtsstaat*, that both the ruler and the ruled are bound to rules, can be interpreted in two ways. The first interpretation is the path to strong legalism. According to this approach, the ruler's being bound to rules is tantamount to his 'not violating' them. This is both a necessary and a sufficient condition for rules to be valid and legitimate. Under the second interpretation – which is adopted by legisprudence – the idea of following rules by a sovereign counts only as a necessary and not as a sufficient condition for rules to be valid. Legal validity on this view is distinct from legitimacy. Legitimacy for its part can only be obtained through legitimation.[48]

The achievement of legitimacy thus requires an additional active step of legitimation, which means not just that the sovereign is subject to the same rules as everyone else (the rule of law; being 'bound to the mast'), but also that the rules which it seeks to promulgate reflect certain required formal

[46] This will be considered later in Chapters 4 and 5.

[47] Wintgens, 'Legislation as an object of study of legal theory' (n 29) 158; Bańkowski (n 15) 46.

[48] Wintgens, *Legisprudence: Practical Reason in Legislation* (n 19) 145.

characteristics, which in turn limit the breadth of possible content that those norms can legitimately have.

Strong legalism is clearly open to abuse: prioritising heteronomy undermines critical reflection and the application of principles of legality that are characteristic (and constitutive) of democracies. Without spaces for interpretation, rules become 'implements of tyranny' and legalism a 'vice of narrow governance'.[49]

By contrast, in addition to a formally valid source of the rule, legisprudence requires the legitimation of the proposed legislative act through its proactive justification. This justification is achieved according to the principles that legisprudence sets out, which guide the conduct of the ruler regardless of the political content of the norm she is making: 'through his ruling activity, while following rules, the ruler must supply reasons for his choices'.[50] One can thus see how legisprudence represents a form of constitutionalism. I will discuss the principles of legisprudence in greater detail in Chapter 4; the goal here has been to set out the problems of strong legalism that they seek to ameliorate. Those problems reach their apex in code, to which we can now turn.

3.2 Computational Legalism

Like strong legalism, code also requires citizens 'not to think about it' and simply to follow the rule as handed down. There is a difference of degree, however, since the legalistic mindset is at least something that can notionally be challenged by the citizen or rejected by the values of a given society. Even without such resistance, the interpretative or hermeneutic gap by definition creates space between the promulgation of the norm and any acquiescence to its requirements. Code, by contrast, admits of no opportunity for challenge: as we saw in Chapter 2, some measure of technological normativity is inherent in its very existence – the boundaries of the field of play, as well as the rules of the game, are determined from the outset, and there is little or nothing the end-user can do to change them, if she is even aware of what they are to begin with (which is far from a given, as we will see below). Not only is she made to 'not think about it', in many cases she is not given the opportunity to apprehend what it is that she is not thinking about.

Returning to the characteristics of legalism set out above, we can think about how they apply in the context of code to create what I call 'computational legalism', the particular species of legalistic obedience that flows from the *sui generis* nature of code as a regulator. First, we saw how legalism

[49] Bańkowski and MacCormick (n 15) 194.

[50] Wintgens, *Legisprudence: Practical Reason in Legislation* (n 19) 145.

concerns itself with rules that are to be followed as written. Code presents us with an extreme form of 'ruleishness', where conditions are hard-edged and admit of no latitude for interpretation. Second, legalism views those rules as a representation of reality. We saw in Chapter 2 how the code of digital architectures actively constitutes, rather than merely represents, the empirical and legal realities that the end-user is presented with (as well as those she cannot perceive). Third, under legalism rules are seen as a-temporal, or timeless – they reflect background truths, and they 'just are'. Similarly, code in a sense 'collapses' time, through a combination of the immediacy of its execution, its immutability at the point of execution, and the cumulative normativity of its pervasiveness. Fourth, the source of the sovereign's power is concealed so that the policy reasons behind the rules it promulgates are ignored, those norms being treated as simply 'there', to be followed without question by the end-user. The opacity of code and the privileging of commercial practices and trade secrets set up a similar concealment in the computational context – the 'sovereignty' of those who produce code is concealed by veils of both technical and legal-economic opacity.

The remainder of this chapter considers these characteristics in turn, demonstrating how the computational form of legalism is much stronger than even the strongest notion of its legal counterpart, in turn underlining the need for its ex ante mitigation through design.

(a) Ruleishness

Hildebrandt observes that textual norms have lives of their own beyond the author:

> Absent ostensive reference, the author is never sure how her text will be understood, while the reader cannot take for granted what the author meant to say. This provides for an inevitable latitude in the use of texts and turns law-making (enactment of legal codes as well as their application) into a creative process rather than a mechanical application.[51]

This is a historical consequence of the embodiment of law in text, implying the choice that text as a medium affords us as to whether or not to adopt a legalistic perspective.[52] This idea of law as a creative process reflects the aspirational view of legality, which I discuss in more detail in Chapter 4,

[51] M Hildebrandt, 'Legal and technological normativity: More (and less) than twin sisters' (2008) 12 *Techné: Research in Philosophy and Technology* 169, 172. This echoes R Barthes, 'The death of the author' in S Heath (ed.), *Image – Music – Text* (Fontana 1977).

[52] L Diver, 'Law as a user: Design, affordance, and the technological mediation of norms' (2018) 15 *SCRIPTed* 4, 30–2 ('4.3 *Operation* versus *formation* of law').

according to which the slow iteration of normative interpretation across heterogeneous circumstances builds towards a body of law that is simultaneously stable and flexible.[53] As we have seen, this is at odds with strong legalism's binary application of rules *qua* rules.[54] This 'ruleishness'[55] is one of the most salient connections between juridical and computational legalism. Taking the orthodox spectrum that at one end has absolute rules that admit of no interpretation, and at the other has more flexible standards that specify broader outcomes but not the detailed means by which they should be achieved, code is very much located towards the rule end.[56] Code execution represents the mechanical application of rules *par excellence*; as Zittrain puts it, 'execution is exquisite'.[57] Simultaneously, code's 'enactment' does not by default (or even generally) admit of the latitude of interpretation that Hildebrandt refers to. The rule as laid down by the designer is the rule that will be followed (not just by the end-user, but also, as we shall see, by the machine). Whereas text-based institutional laws are created in the knowledge that the passage of time and the ambiguity of language will permit consideration of exceptional circumstances or evolving social norms, code 'requires extreme precision and rigor not resident in analog law'.[58] Without that precision, the code will simply fail to execute.

As a result of this, three profound consequences flow from code's ruleishness, namely (1) its mindless execution wherever the conditions it requires are met, (2) the total absence of performance in circumstances where exactly those conditions are not met, leaving no possibility of sensitivity to edge cases, and (3) its inability to respond to changes in the world that lie outside of its predetermined and necessarily limited ontology. Each of these is explored below, followed by a discussion of the difference between the 'rules' the machine follows versus the rules *qua* technological normativity that they in turn create, and to which the end-user is subjected.

Mindless Execution

First, the rules specified ex ante in the code will be applied in all instances where the conditions they require are present, regardless of any ex post

[53] Hildebrandt, 'Legal and technological normativity' (n 51) 171–2.

[54] Bańkowski and Schafer (n 1) 34.

[55] I adopt this term from J Grimmelmann, 'Regulation by software' (2005) 114 *The Yale Law Journal* 1719.

[56] Ibid. 1723. See also K Yeung, 'Can we employ design-based regulation while avoiding *Brave New World*?' (2011) 3 *Law, Innovation & Technology* 1.

[57] J Zittrain, *The Future of the Internet and How to Stop It* (Yale University Press 2008) 107.

[58] LA Shay et al., 'Do robots dream of electric laws? An experiment in the law as algorithm' in R Calo, A Froomkin and I Kerr (eds), *Robot Law* (Edward Elgar Publishing 2016) 274.

considerations (although of course inputs at run-time, from the end-user or some other source such as a sensor or an oracle, will in many cases be required for the ex ante specifications of the rules to be met). In the technical context this is of course a major benefit: even the most complex body of rules can be expected to execute in predetermined ways under precisely defined conditions, giving a notional predictability that has facilitated the rapid innovation that society has observed over the past several decades of the silicon era.

This is connected with the automated and immediate nature of code, which means that once released into a production environment it can repeat the same set of operations millions or billions of times with little or no marginal cost, and with no human intervention required beyond the maintenance of computing and energy infrastructure and any input necessary for its operation.[59]

Again, provided the conditions specified in the code are met, the code will execute automatically, regardless of any other consideration, provided that it is formally valid. Indeed, this point about formal validity is one of the problems of code's instrumentalism: the machine will execute semantically correct commands faithfully and with no regard to their consequences, which, depending on the behaviour and the pervasiveness of the code in question, can be catastrophic.[60] Back in the legal realm, this is quite evidently undesirable. Even the most 'ruleish' of textual legal norms requires interpretation in order to move from the page to behavioural instantiation, and even where the rule is one of strict liability (for example a speed limit for drivers), enforcement still requires an active process of interpretation, in the course of which justificatory or excusatory reasons may come to light which modulate a strongly legalistic application of the original rule (for example the driver was rushing to get her injured passenger to hospital).[61]

[59] Grimmelmann (n 55) 1729. See also Yeung (n 56). The low marginal cost is part of the orthodox story, of course – focus is now beginning to turn to the vast amount of energy required to run the ballooning infrastructures necessary to meet demands for computation, and the contribution this is making to climate breakdown.

[60] H Surden, 'Values embedded in legal artificial intelligence' (University of Colorado Legal Studies Research Papers 2017) 5. For a recent example that continues to affect both hardware and software at the most fundamental level of execution, see 'Meltdown and Spectre' (2018) <https://meltdownattack.com/> last accessed 4 March 2021.

[61] Shay et al. use the example of speed cameras in their discussion of the practical difficulties inherent in transposing a textual norm into code. See Shay et al., 'Do robots dream of electric laws?' (n 58). For a fascinating discussion of techno-regulation of speeding that bridges Hartian and Latourian notions of law, see K de Vries and N van Dijk, 'A bump in the road. Ruling out law from technology' in M Hildebrandt and J Gaakeer (eds), *Human Law and Computer Law: Comparative Perspectives* (Springer 2013).

Hard Edges

Second, as long as the precise conditions specified in the rule do not exist, it will never be executed. No matter how closely the code-based rule matches the circumstances that arise in operation, if the two do not match then the code will remain inert. Taken together with the first characteristic, there is therefore in code an emphatic absence – and indeed impossibility – of Hart's concept of the 'penumbra of doubt':[62] there can only be the core of meaning, except that the core expressed in the code reflects the subjective understanding of the designer, and not necessarily the settled meaning understood by the legislature, courts, or society more generally.[63] As Grimmelmann notes, the 'hard edges' of software rules are not susceptible to blurring, no matter how complex the set of rules is that is being applied[64] (which, in the modern computing context, will invariably be staggeringly complex). Whereas a human's ability to apply simultaneous rules might result in less precision as the set increases, there is for practical purposes no such limitation for code (except that speed of execution might suffer).

Limited Ontology

Third, and as a corollary of the two characteristics above, code's ruleishness limits by definition the conditions that it will respond to. This limiting of possibilities is put in place by the designer, who of course is interested in solving a particular problem by a particular set of technical means, each of which is considered from the perspective of the underlying business model and the norms and assumptions about formalisation that are a part of whatever computing 'discourse' she inhabits.[65] In so doing, she may fail to consider the

[62] Hart (n 41).

[63] Work on fuzzy logic seeks to map the indeterminacy of language onto the determinacy of numbers, while defeasible and non-monotonic logic are aimed at countering the otherwise brittle logic of code rules. See, for example, R Binns, 'Analogies and disanalogies between machine-driven and human-driven legal judgement' (2021) 1 *Journal of Cross-disciplinary Research in Computational Law* <https://journalcrcl.org/crcl/article/view/5> last accessed 19 April 2021; G Governatori and S Sadiq, 'The journey to business process compliance' in J Cardoso and W van der Aalst (eds), *Handbook of Research on Business Process Modeling* (IGI Global 2009); L Philipps and G Sartor, 'Introduction: From legal theories to neural networks and fuzzy reasoning' (1999) 7 *Artificial Intelligence and Law* 115, 122 *et seq*. It remains to be seen how far such approaches extend into mainstream code development practice, however.

[64] Grimmelmann (n 55) 1733.

[65] PE Agre, *Computation and Human Experience* (Cambridge University Press 1997) 44–8; F Flores et al., 'Computer systems and the design of organizational interaction' (1988) 6 *ACM Transactions on Information Systems (TOIS)* 153. For a critique, in this vein, of machine learning, see D McQuillan, 'Data science as machinic neoplatonism' (2018) 31 *Philosophy & Technology* 253. All of this is mediated by the affordances of the tools the

other possibilities that were relevant to the situation, thus reducing the world to an inappropriately limited set of conditions and responses. From her perspective, she intends that in the operation of the system conditions *A*, *B*, or *C* will arise, and the system should respond with one or a combination of *X*, *Y*, or *Z*. These conditions comprise the entirety of the closed world that is assumed by the code's 'ontology', which once it is defined is rigid and cannot, without the code being altered, be made to be sensitive to conditions *D* and *G*, or responses *W* and *Q*. The designer's predetermined view of the code's operation (conditions *A*, *B*, and *C*, and potential responses *X*, *Y*, and *Z*) is thus reified, and although that reification will not reflect the empirical reality of the world,[66] or the requirements of substantive law, or some other relevant normative value such as legitimacy or the rule of law, this fact will pose no barrier whatsoever to the execution of the code on the basis of the ontology the designer builds her artefact around.[67] As Grimmelmann puts it, '[w]hen a programmer creates a program, she predetermines its responses to every possible input – to every possible "case" it may adjudicate. The algorithm is the rule.'[68] Once compiled into commands executable by the machine the code is 'closed', and no information that has not somehow been represented there can make its way in post hoc to alter the nature of that execution.[69]

There is a connection here with Hart's notion of the open texture of language, and his argument against attempts to regulate unambiguously in advance:

> If the world in which we live were characterized by only a finite number of features, and these together with all the modes in which they could combine were known to us, then provision could be made in advance for every possibility . . . Everything could be known, and for everything, since it could be known, something could be done and specified in advance by rule. This would be a world fit for 'mechanical' jurisprudence. Plainly this world is not our world.[70]

---

designer uses, from the programming language to the integrated design environment. I return to this important theme in Chapter 7.

66 This is sometimes termed the 'map–territory relation'.

67 Agre (n 65) 48.

68 Grimmelmann (n 55) 1732.

69 M Krajewski, 'Against the power of algorithms closing, literate programming, and source code critique' (2019) 23 *Law Text Culture* 119; P Swartz, 'How do programs mean?' in *Division III: Essays in Programs as Literature* (Hampshire College 2007) 78–80. Strictly speaking, compiled and interpreted languages differ on this point, but pragmatically they are the same, at least from the perspective of the end-user.

70 HLA Hart, *The Concept of Law* (2nd edn, Clarendon Press 1994) 128.

Code makes this vision a reality, although not in the way Hart imagined. It goes further by *imposing* such 'mechanical jurisprudence' on a contingent and complex world. It reduces that complexity to the set of features that the designer thought to include, whether or not those features are sufficient in terms of their number or the appropriateness of the types of formalisation used, and whether or not they capture the necessary features of whatever context(s) the code will ultimately operate in.[71] This solipsistic ontology allows code to operate at great speed with immense predictability, but this is inherently limited in scope to only what has been anticipated (and will include the inevitable bugs that accompany its implementation).[72] By contrast, law must be capable of contingently accommodating, referencing, and interacting with systems external to itself (that is, the society it is intended to serve).[73]

Bańkowski made a connection early on between legalism and code-based regulation in his description of a hypothetical system for borrowing ebooks from a library.[74] His description shows how the transition from an 'offline' manual library system to an automated system affects the rules that governed the former. The rules state:

1. borrowers must complete a separate form for each volume borrowed,
2. books should be returned before the due date,
3. borrowers have a limited number of loans that must not be exceeded,
4. no further books will be loaned to borrowers who have overdue loans.[75]

These regulations are transposed into code, governing the 'borrowing' of an ebook that will 'self-destruct' after the appropriate borrowing period.[76] (This is of course an archetypal DRM system, where ex ante constraints on media use define its availability and are embedded in and enforced by the artefact itself.) The computational legalism of such a system becomes evident when we consider what happens to the textual rules listed above, which previously under the manual system were interpreted and applied by the human librarian. The rules are bright lines that admit of no interpretation – once the

[71] On this general theme, see GC Bowker and SL Star, *Sorting Things Out: Classification and Its Consequences* (MIT Press 2000). There is a significant overlap here with the problems of attempting to formalise 'ground truths' in machine learning research design.

[72] An example of the combined problem of bugs and code's limited ontology is the Post Office Horizon scandal, which I discuss in Chapter 6.

[73] M Hildebrandt, 'Code-driven law: Freezing the future and scaling the past' in SF Deakin and C Markou (eds), *Is Law Computable? Critical Perspectives on Law and Artificial Intelligence* (Hart 2020).

[74] Bańkowski (n 15) 54 ('Norms and Machines') *et seq*.

[75] Ibid. 55.

[76] Ibid.

borrowing limit is reached, if no appeal process is built into the code then no further books can be borrowed, regardless of any extenuating circumstances (recall the third consequence of code's ruleishness discussed above). Once the borrowing period is reached, the book 'self-destructs', again regardless of any extenuating circumstance that might have moved a human librarian to make an exception (for example a combination of illness and exams).

The Absence of Interpretation

In sum, these characteristics highlight the absence of interpretative possibilities in code. There is no tolerance of ambiguity, no possibility of discretion or subversion, and little space to reason separately from either interpreting or even, in the first instance, identifying the rule.[77] Digital systems are thus 'crude and inflexible, often brutal and not open to critical reason'.[78] Returning to the example above, the speed camera detects one of three conditions (the car is travelling below, at, or above the speed limit) and it has two responses (do not take photo; take and process photo). In spite of the legalism of such a system, there is nevertheless an ex post buffer to enable some interpretation – a human may interpret the photo and, upon realising the vehicle was an ambulance, override the automated decision. The decision to include such oversight (that is, a 'human in the loop'[79]) is a design choice and is by no means a given – there is no technological barrier preventing a speed camera and penalty system being fully automated with no ex post adjudication[80] but we choose not to do so because there are other important values that must be represented.[81] Such scenarios demonstrate how the three elements of ruleishness can come together to amplify legalism in the computational context, especially when they are combined with the automation and immediacy of code.

[77] P Swartz, 'A tower of languages' in *Division III: Essays in Programs as Literature* (Hampshire College 2007) 96–7; Grimmelmann (n 55) 1723. On reason as an element of decision-making that is distinct from the rule, see N MacCormick, *Rhetoric and the Rule of Law: A Theory of Legal Reasoning* (Oxford University Press 2005) chapter 2.

[78] Bańkowski and Schafer (n 1) 46.

[79] W Hartzog et al., 'Inefficiently automated law enforcement' (2015) *Michigan State Law Review* 1763, 1780 *et seq*. I return to this theme in Chapter 6, discussing the digisprudential affordance of delay.

[80] LA Shay et al., 'Confronting automated law enforcement' in R Calo, A Froomkin and I Kerr (eds), *Robot Law* (Edward Elgar Publishing 2016); C Gavaghan, 'Lex machina: Techno-regulatory mechanisms and rules by design' (2017) 15 *Otago Law Review* 123, 130–1.

[81] This is reflected in Art. 22 of the GDPR, regarding automated decision-making and the difference made by having a human involved.

(b) Representationalism

Representationalism is a key aspect of strong legalism: in both its *jusnaturalistic* and positivistic accounts of the origins of law, legal norms are held to represent reality. As discussed above, for the former legal validity is derived from the underlying truth of nature, while for the latter it comes from the founding social contract, rule of recognition, et cetera. If the underlying natural norms or the founding political act are true, then the rules which flow from them must also be true.[82]

The way in which code constructs a rule-based normativity is of course not the same as law's approach. The regulative force of law is (and indeed ought to be) limited by its instantiation in the technology of the script (that is, text), which creates an interpretative or hermeneutic gap between what the text requests and how the addressee(s) of that text interpret its terms and choose to reflect them in their behaviour.[83] The instantiation of code rules, on the other hand, is not so limited: technological normativity, as we saw in Chapter 2, can have a direct effect in a way that legal normativity – necessarily constrained by its textual embodiment – does not.[84]

Does Code Contain Rules per se*?*

One crucial way in which code can be said to be less than law is that it does not promulgate rules in the basic sense of providing citizens with a set of guidelines they can find, interpret, and adapt their behaviour to follow. Nor does an appreciation of code's representationalism require any metaphysical gymnastics to connect its rules with empirical reality or some contested notion of 'truth'. The rules of code work in a different way – they are not Austinian commands to be followed,[85] nor are they the ex post representations of the norms of a community, against which standards of conduct can be evaluated.[86] Instead, they are instrumental tools that crystallise behavioural possibilities and limits from the outset, with varying levels of normative 'force' (recall the

[82] Wintgens, 'Legisprudence as a new theory of legislation' (n 24) 4.

[83] Diver (n 52); Hildebrandt, 'Legal and technological normativity' (n 51) 175.

[84] Hildebrandt, 'Legal and technological normativity' (n 51) 176. For more on the consequences for law of text as a medium, see J Goody, *The Logic of Writing and the Organization of Society* (Cambridge University Press 1986) chapter 4 and, more generally, WJ Ong, *Orality and Literacy: The Technologizing of the Word* (3rd edn, Routledge 2012).

[85] J Austin, *The Province of Jurisprudence Determined*, ed. WE Rumble (Cambridge University Press 1995), *passim*.

[86] See P Winch, *The Idea of a Social Science and Its Relation to Philosophy* (Routledge & Kegan Paul; Humanities Press 1990) 24 *et seq.*, discussing L Wittgenstein, *Philosophical Investigations*, trans. GEM Anscombe (Blackwell 1968).

discussion of the spectrum of technological normativity in Chapter 2). The result is that the gap that otherwise exists between the text of a legal rule and its effect on behaviour in the physical world collapses. Whereas the law has used legal fictions to maintain its position as ultimate arbiter of the social world, it now has to compete with code that constitutes both empirical and normative landscapes.[87] More and more, it is losing this competition: 'the virtual is a mode of reality that evades the space-time categories of the law'.[88]

The extent to which the hermeneutic gap collapses is profound; its effacement in code is not only easy but is entirely normal, and not necessarily through malice or intentional obfuscation (although these are a significant concern), but simply by the very nature of the medium. The gap can so easily be collapsed because the 'text' of the rule (the code) constitutes directly the geography of the artefact: they are not just isomorphic, they are one and the same, at least at the level of computation. Unlike law, whose 'carrier' has hitherto been the inherently passive medium of text, software code allows us to, in Latour's words, 'conceive of a text (a programming language) that is at once words and actions'.[89]

In code we find the collision of rules and reality, where what was 'ought' becomes simply 'is' (or, at least, a categorical 'will be'). In this way, representationalism finds its apex: no appeal to metaphysical belief is required to see how computer code not just represents reality, but actively constitutes it (or at least a part of it). Behavioural possibilities are constituted, and not merely regulated, by code rules. If legal rules can ultimately be decomposed into an 'if this, then that' structure,[90] code rules exemplify and amplify this reality, given that is in practice exactly how they are expressed from the outset.

Returning to Bańkowski's library borrowing system, we can appreciate how in their translation into code the rules become simply descriptive rather than regulative. The tracking of library users and their loans is obviated by means of swipe-card authentication (rule 1 collapses); the end of loans and the 'return' of ebooks is automated by code, thus rendering rule 2 descriptive; and rule 3 merely describes the state of the system that rule 4 enforces (again,

[87] What Vismann and Krajewski call 'digital virtuality'. See C Vismann and M Krajewski, 'Computer juridisms' (2007) *Grey Room* 90, 92.

[88] Ibid.

[89] B Latour, 'Where are the missing masses? The sociology of a few mundane artifacts' in WE Bijker and J Law (eds), *Shaping Technology/Building Society: Studies in Sociotechnical Change* (MIT Press 1992) n 1.

[90] N MacCormick, *Institutions of Law: An Essay in Legal Theory* (Oxford University Press 2007) 24 *et seq*. Importantly, this point is only about the structure of the rule and not the reasoning which interprets and applies it – which of course is precisely the point of comparison with code.

automatically). The 'if this, then that' instantiation of rule 4's logic simply does not allow the function in the code that issues loans to be executed. As Bańkowski puts it, '[w]hat we see then, is how the normative has become the descriptive. This gives us an example of rule following which has the machine-like quality of heteronomy: we "don't think about it".'[91] The library's rules, which were once normative, have become descriptive – they are simply how the system, and the end-user within it, will inevitably operate. There is no gap or space between the rule and its imposition; the code rule constitutes reality. If the stored number of books is at a certain threshold, then no further loans will be permitted, regardless of any extenuating circumstances. The code rule means that the computer will simply say no.

There is more to the story, however. A full analysis of code rules requires us to think at more than one level of abstraction. In that vein, Asscher draws a distinction between rules on the 'conceptual level' and the 'technical commands within a certain computer language',[92] concluding that it is the former that is ultimately what matters. This is correct insofar as it is necessary to think abstractly when comparing instantiations of rules in code with the textual legal rules from which they may be derived (this would be the techno-regulation level, which I discuss in Chapter 5). However, we must not ignore the concrete materiality of the technical commands that are the building blocks of the normativity that ultimately implements those rules. Although it might quickly become cumbersome to focus on the minutiae of the individual commands in source code, the materiality of the system in operation is precisely where the action happens, and so it is necessary to focus our attention at that level, at least to some degree. The challenge is to find an appropriate abstraction threshold between individual commands and the technological normativities that collectively they bring into being. In the end, it is code that performs this translation of normativities, and for better or worse that code is necessarily designed, even if not all of its effects can be anticipated in advance.

At this more abstract level, Leenes and Koops suggest that the negligent production of privacy-eroding code can be viewed as akin to a rule stating that privacy is not important, or is less important than other values:

> Although this is perhaps stretching the term 'rule' rather far, we are inclined to think that the development and application of code that negligently fails to take privacy effects into account can indeed be seen as the embedding

[91] Bańkowski (n 15) 56.

[92] L Asscher, '"Code" as law: Using Fuller to assess code rules' in E Dommering and L Asscher (eds), *Coding Regulation: Essays on the Normative Role of Information Technology* (TMC Asser Press 2006) 83.

> of a 'rule' in the technology, namely that privacy is unimportant and secondary to other values that the code primarily serves. Such technology does indeed serve to guide or control (what is perceived as) proper and acceptable behaviour, since privacy-infringement is considered an acceptable outcome of its use.[93]

One can appreciate how abstracted this perspective is from the idea of viewing individual code instructions as rules. In this case, the broader functionality of the code and how it guides behaviour according to a particular stance on a given value (in this case privacy) is interpreted as a kind of rule. Here we can see a connection with the design theories discussed in Chapter 2: we could frame the above in terms of the (dis)affordance of privacy-protecting functionality as being a *de facto* rule embodied in that particular code.

Adapting Leenes and Koops's formulation, one might say that code rules that fail to embody standards of legitimacy in effect represent 'rules' stating that those standards are not important and need not be valued. The (dis)affordances and inscriptions that the code embodies can be seen as 'rules' of this sort, the corollary being that code can and ought to be made compatible with those standards by providing certain affordances. If it does so, the likelihood of the code artefact's normativity being illegitimate is accordingly reduced.

From this level of abstraction, we can appreciate the normativity the code imposes without having to look directly at the underlying commands, which is one useful strategy for connecting orthodox notions of what makes a rule and the normativities that code rules in fact bring into being. Maintaining a holistic sensitivity to code's effects in this way can help us avoid too narrow a focus on just its purposive effects (that is, what its designers purport its functionality to be), a perspective that much of the literature adopts implicitly.[94] We saw in Chapter 1 how digisprudence is concerned with not just the intended regulatory effects of code, but also those wider 'techno-effects' that may not have a legal underpinning, whether public or private.[95] It is important to consider not just the intended normative effects of a system but also its potential unintended effects, and if those undermine the legitimacy of the code's normativity, this becomes a real cause for concern. The

[93] R Leenes and B-J Koops, '"Code" and privacy or how technology is slowly eroding privacy' in E Dommering and L Asscher (eds), *Coding Regulation: Essays on the Normative Role of Information Technology* (TMC Asser Press 2006) 191.

[94] Cf. B van den Berg and RE Leenes, 'Abort, retry, fail: Scoping techno-regulation and other techno-effects' in M Hildebrandt and J Gaakeer (eds), *Human Law and Computer Law: Comparative Perspectives* (Springer 2013).

[95] Ibid. 81.

relational focus of both affordance theory and postphenomenology requires us to think about what the code is actually doing, as opposed to merely what it is intended by its designer to do. By viewing code rules at these different levels of abstraction, we can conceptualise the role they play in constituting and regulating behaviour.

Constitutive and Regulative Rules

We saw in Chapter 2 the distinction between constitutive and regulative technological normativity.[96] Hildebrandt discusses the difference between legal rules that are constitutive of other (institutional) facts or rules, and those that aim to regulate behaviours which can take place independently of the rule's existence.[97] For example, the institutional fact of marriage cannot exist independently of a constitutive rule which creates or institutes it,[98] while it is possible to drive at 100 miles per hour even though there is a regulative rule which prohibits it.[99] Regulative rules are therefore aimed at regulating existing activities, while constitutive rules 'create the very possibility of certain activities'.[100] In a sense, then, constitutive rules are creative, or generative, while regulative rules are limiting. Searle discusses the example of chess: the rules of the game do not regulate what was already happening (that is, we do not tend idly to push around on a chequered board miniature figurines representing kings, queens, knights, etc.); rather, the rules in fact constitute the game. The game of chess does not exist outside of its constitutive rules – if people ignore those rules, they may be playing something, but it is not chess. The constitutive rules are thus creative in their bringing about (1) the general institution of 'chess', and (2) the contingent institutional fact of any given game of chess.[101]

This idea of an 'institutional fact' stems from the distinction between facts that are socially constructed, and 'brute facts' which exist 'out there' in

[96] See 'A Spectrum of Technological Normativity' in Section 2.2.

[97] Hildebrandt, 'Legal and technological normativity' (n 51) 172 *et seq*.

[98] MacCormick's suggested sub-division of constitutive rules into institutive, consequential, and terminative rules adds helpful pragmatic granularity to the broader scope of the former, but we can continue for present purposes to use the term as-is. See MacCormick, *Institutions of Law* (n 90) 36–7.

[99] Hildebrandt, 'Legal and technological normativity' (n 51) 172.

[100] JR Searle, *The Construction of Social Reality* (Free Press 1995) 27.

[101] Ibid. 27–8. One can also appreciate here the contrast drawn between 'broad' and 'narrow' normativity in M Piekarski and W Wachowski, 'Artefacts as social things: Design-based approach to normativity' (2018) 22 *Techné: Research in Philosophy and Technology* 400.

empirical reality.[102] Obvious examples of institutional facts include a 'university' and a 'doctoral degree'. Examples of brute facts include the distance between the earth and the sun at this very moment, or the amount of force I am currently exerting on my keyboard.[103] An 'institution' in this sense refers to an arrangement recognised within the relevant community or form of life, as in the game of chess, as opposed to an agency. (Having said that, the two concepts can overlap, potentially symbiotically, within the same entity. For example, a university is both an 'institution-arrangement' – like a game of chess, it is something whose character is borne of certain features being observed and maintained through time by those with the relevant capacities committing to doing so – and it is also an 'institution-agency', that is an organisation empowered *inter alia* to confer doctoral degrees.[104])

Because institutional facts are 'thought-objects'[105] that we create as part of our shared institutional world, they do not exist or make sense independently of that world and can therefore be brought into being only by following the criteria agreed to by the members of the relevant community. This is the creative work done by constitutive rules. One can appreciate the tension here between legalism on the one hand, which holds that the constitutive rules of law (quintessentially a system of institutional facts) are 'out there', and the viewpoint I am advancing, which seeks to question the design of those constitutive rules.

From a legalistic perspective, constitutive rules can be arranged in a hierarchy which creates the underlying framework (itself an institutional fact or set of facts, sometimes termed a 'constitution') within which other rules can be made. This is Wintgens's proxy model, discussed earlier, where the legitimacy of a given legal rule flows from some founding act which operates in the

[102] Searle (n 100).

[103] It should be noted, however, that the units we use to conceptualise these two facts are in a sense themselves institutional (that is, part of a shared social understanding), because the scientific practices that have resulted in them are not objective, even if the physical reality they seek to represent is. See for example Ihde's discussion of the 'hermeneutic relation' that mediates our experience of the world (or the universe) via the reading of scientific instruments in *Technology and the Lifeworld: From Garden to Earth* (Indiana University Press 1990) 80 *et seq.*

[104] MacCormick, *Institutions of Law* (n 90) 35–7.

[105] O Weinberger, 'The norm as thought and as reality' in N MacCormick and O Weinberger, *An Institutional Theory of Law: New Approaches to Legal Positivism* (Springer 1986). From an external perspective, a legal-institutional fact might be treated as brute: 'depending on one's perspective, any brute fact can be rearticulated as an institutional fact, while institutional facts can be "used" as brute facts to be regulated' (Hildebrandt, 'Legal and technological normativity' (n 51) 172).

background to validate those subsequently promulgated norms. As previously noted, the idea is connected with various accounts of law that culminate in a notional foundational legal rule.

As with the institution of chess, in law constitutive rules are necessary to inaugurate valid instances of what MacCormick calls institution-arrangements (for example a contract or marriage), institution-agencies (for example a university or local authority), and intangible institution-things (for example a patent or a stock portfolio).[106] The legal institution of marriage, for example, is defined by the requirements laid down in certain constitutive rules, and a particular institution-arrangement of marriage is an institutional fact brought into being by the following of those rules. To speak of a couple being 'married' outside the institutional framework (in both the universal and particular senses of 'institution') does not make sense, or at least does not point to a legally recognised institutional fact.[107]

How does all of this relate to code? Whereas the institutional facts created by legal constitutive rules are always only 'real' within the law's own 'regime of veridiction'[108] or institutional order, code-based rules are 'brute' in the sense that they are 'just there' and part of the 'physical' fabric of the system. Recalling the discussion of rules above, this is true in every case where the notion of 'rule' refers to the individual instructions given to the machine – the instructions are brute facts in the sense of manipulating physical reality at the level of the machine's hardware. Moving up towards more abstract notions of rules, code-based rules become potentially less brute, in the sense that spaces can be opened up in which more than one course of action is open to the end-user. The extent to which she is accorded this 'freedom' is down to the particular mix of intentional and unintentional (dis)affordance embodied in the design, but, as previously stated, whatever the level of flexibility the design provides, this too is necessarily bounded from the outset.

Code-based rules, then, are 'constitutive of our behaviour'.[109] Just as our notions of legal rules have moved beyond the earlier Austinian command model to encompass the broader 'creative' or empowering aspect of secondary or institutive rules,[110] so too does code represent an example of this 'creative', constitutive form of normativity-generation. A crucial difference in the code context, however, is that it is not the citizen who is so empowered by those constitutive rules to create the relevant normative constructs, such as

[106] MacCormick, *Institutions of Law* (n 90) 35–6.
[107] Hildebrandt, *Smart Technologies* (n 11) 145.
[108] Ibid.
[109] Hildebrandt, 'Legal and technological normativity' (n 51) 174.
[110] See, respectively, Hart (n 70) 91 *et seq*.; MacCormick, *Institutions of Law* (n 90) 36–7.

a contract, will, or whatever. Instead, the creative 'instituting' of the relevant form of normativity (technical, not legal) is performed by the designer of the code, who constitutes the terms of behaviour of the citizen as if the latter were a subject of the former's sovereign power. In including, excluding, and defining the limits of behaviour within the ambit of the system, code 'rules' limit end-user freedom in material ways. The qualitative difference is that, with code, these limits are not merely regulative, where the end-user, already engaging in whatever activity, is 'requested' to amend her behaviour this way or that (this is the equivalent of the speed limit which *ceteris paribus* can only ever 'request' that the driver travel at less than 70 miles per hour). Rather, the limits are constitutive, in that the salient features of the behaviour are themselves defined by code. Whereas the chess player can use the architecture of the artefact (the board and its pieces) in ways that are outside the rules of the game, for example to play something less complex like draughts, the architecture of code sets the rules in place ex ante, and does not by default allow them to be so adapted at the discretion of the end-user. To quote Lessig, 'one obeys these laws as code not because one should; one obeys these laws as code because one can do nothing else'.[111] The constraints and enablements of code are 'like laws of nature, like the world as we find it';[112] they are simply 'there', with the crucial difference between code and law being that rather than suggesting an 'ought', as law in its regulative capacity always does, code can simply impose an 'is'.[113] The three traditional phases of regulation (direction, detection, and correction)[114] are thus collapsed into a single step.[115]

We will see later that it is possible for code rules to be regulative so that the end-user is in fact invited to behave in certain ways, and of course many digital systems do allow a range of behaviours in end-users' interactions with them. Social networks, for example, give end-users fairly wide latitude on the volume and content of the text, images, and videos they can upload. But even within this seemingly unlimited freedom to upload there are nevertheless

[111] L Lessig, 'The zones of cyberspace' (1995) 48 *Stanford Law Review* 1403, 1408.

[112] Grimmelmann (n 55) 1740. Similarly, Bamberger notes that 'the fact that architectural technology embodies normative choices at all can escape notice, as the perfect constraints code places on behavioral possibility can seem as natural, immutable, and invisible as the laws of physics'. See KA Bamberger, 'Technologies of compliance: Risk and regulation in a digital age' (2010) 88 *Texas Law Review* 669, 724–5.

[113] R Brownsword, 'Lost in translation: Legality, regulatory margins, and technological management' (2011) 26 *Berkeley Technology Law Journal* 1321.

[114] C Scott, 'Regulation in the age of governance: The rise of the post-regulatory state' in J Jordana and D Levi-Faur (eds), *The Politics of Regulation: Institutions and Regulatory Reforms for the Age of Governance* (Edward Elgar Publishing 2004) 147.

[115] Brownsword, 'Lost in translation' (n 113) 1344.

constitutive limits – the code will accept only text, images, or video files (not PDFs, executables, or images/videos in formats the platform does not recognise), and only a certain quantity of data can be uploaded. For most ordinary usage these boundaries will never be approached, and so the end-user may be unaware of their existence, but they are nevertheless always and necessarily there.

In many cases, then, the normativity embodied in code is more constitutive than regulative of the practice in question. The more constitutive a digital system is of a given behaviour, the more control the designer has exercised in defining the nature of that behaviour. In many cases it will be more profitable to limit the 'regulative latitude' given to the end-user, channelling her behaviour in ways that are commercially beneficial (some of these were discussed in the previous chapter). Even on a practical level, building in or enlarging the contingent regulative space at the expense of constitutive certainty requires anticipation of more possible conditions, which in turn requires more code and therefore more expense in its creation, maintenance, and support. This will lend further commercial impetus towards taking a constitutive rather than a regulative approach to design.

Rules for Humans; Rules for Machines

One might detect an ambiguity in the discussion above as to who the addressee is of a rule found in code. In the abstract sense discussed above and in the previous chapter, the end-user's interactions with the system are constituted and delimited by the design of that system's code, facilitating some acts while preventing others. Continuing the analogy with legislation, the end-user is the 'addressee' of these 'rules',[116] what she can do being structured in all the ways previously discussed. This is the level that Asscher suggests we ought to focus on. But that level of rule is itself necessarily constituted by a further, deeper level of 'rule-following', where the machine is the 'addressee' of the instructions contained in the code, those instructions being intended in turn to produce that end-user-facing technological normativity at the higher level. We might conceptualise these relationships as shown in Figure 3.1, where source code directs the machine, creating the technological normativity that directs the end-user.

Without the 'internal', absolute ruleishness of the source code that directs the machine, the 'external', abstract ruleishness of the artefact that directs the end-user will never come to be. As we saw in Chapter 2, even where there are

[116] Assuming that is possible, bearing in mind the discussion in Chapter 2 of signifiers and the distinction between real and perceived affordances.

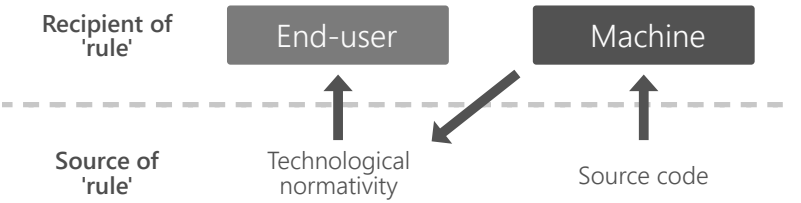

Figure 3.1 Sources of 'rules' in code

spaces in the latter that allow for contingent behaviours, these are themselves ruleishly delimited, simply because they must be – whatever nature those interactional possibilities might have, they could not exist without code that reflects the three characteristics of ruleishness described above. So, whereas technological normativity might be resist-able by the end-user, the machine has no choice or latitude as to whether or not it follows the instructions it is given – and so the configuration of technological normativity experienced by the end-user is itself necessarily structured by the absolutism of machinic rule-following.

These multiple roles that code plays – a text describing what its execution will do, a set of rules for the machine to follow, and, ultimately, the framework for the experience that the end-user will have – suggest some important sites at which practical approaches to ensuring legitimacy can be introduced. I will return to this theme of code deconstruction in Part III of the book, where I discuss how the text of code provides us with potential avenues for both interrogating and facilitating its legitimacy.

(c) Immediacy

Code's immediacy refers to the temporal aspect of execution. As discussed above, the hermeneutic gap between text and behaviour is collapsed, but not only does the text of code constitute both rule and reality, but the imposition of it is arranged prior to, and imposed immediately at the point of, execution. The conditions specified in that prior arrangement are imposed without delay and without consideration of alternative actions that might have been appropriate. There is no scope to 'hesitate well', as in Latour's description of judicial practice;[117] linear time is in a sense compressed, further distancing the character of code's operation from the role that law's more measured pace plays

[117] B Latour, *The Making of Law: An Ethnography of the Conseil d'Etat* (rev. edn, Polity 2009) 193–4.

in the stabilisation of societal expectations.[118] After the point of 'closure',[119] the compilation of whatever configuration of normativity is contained in the code, the machine will execute it faithfully and as quickly as it is physically capable of.

Grimmelmann refers to this characteristic as code's immediacy, where the ex ante nature of code means that '[s]oftware cannot – as law can – adapt its response in light of later-available information or a later determination that such information is relevant.'[120] As we have seen, whereas law as a prospective regulator is inert in the absence of the will to reflect its terms in real-world behaviour, the enablements and constraints of code, put in place by the designer, have latent efficacy even before the system is operational. Recalling the discussion of the library above, the end-user is simply subjected to the ex ante 'ruling' of the code: she may not borrow any more ebooks, the overdue ebook ceases to be accessible, et cetera. The code's swiftness is brutal and entirely impervious to external reason or extenuating circumstances.[121]

Default Configurations

Immediacy is embodied in code even where the design includes the possibility of altering its configuration. End-users tend to accept as a 'natural and immutable fact'[122] the configuration an artefact has when it is supplied to them. This may be due to automation bias, whereby the configuration and responses of a machine are assumed to be more appropriate or 'correct' than a human equivalent.[123] Clearly, the designer has significant power in choosing one starting configuration over another. As Tien notes, 'default settings may seem "normal" because the equipment is common, or have become "legitimate" as people have grown accustomed to the situation presented

[118] M Hildebrandt, 'A vision of ambient law' in R Brownsword and K Yeung (eds), *Regulating Technologies: Legal Futures, Regulatory Frames and Technological Fixes* (Hart 2008) 186–7. See also N Luhmann, *Law as a Social System*, ed. F Kastner et al., trans. KA Ziegert (Oxford University Press 2004) 205 *et seq*.

[119] Krajewski (n 69).

[120] Grimmelmann (n 55) 1730.

[121] Bańkowski and Schafer (n 1) 46.

[122] M Goldoni, 'The politics of code as law: Toward input reasons' in J Reichel and AS Lind (eds), *Freedom of Expression, the Internet and Democracy* (Brill 2015) 128. See also JP Kesan and RC Shah, 'Setting software defaults: Perspectives from law, computer science and behavioral economics' (2006) 82 *Notre Dame Law Review* 583, 591 *et seq*.

[123] DK Citron, 'Technological due process' (2008) 85 *Washington University Law Review* 1249, 1271–2.

by the equipment'.[124] Simultaneously, the 'is-ness' of default settings militates against enquiring as to whether other configurations might be more suitable for the end-user – instead, the defaults are accepted as immutable facts, and alternatives (should they even imagine them) as impossible or even unreasonable.[125] For many end-users, the assumption will in many cases be that the designer knows best.[126] Furthermore, they might lack the technical sophistication to investigate all the possible customisation options,[127] or they might not have the time to do so,[128] much less to think critically about what incentives might have motivated the designer to select a certain configuration of defaults or why these occasionally change in ways users do not necessarily support.[129] The behaviour-guiding quality of default configurations is often stripped away by the perception that they are 'mere design features';[130] as with legalism, they are taken to be part of what is 'just there'. Even where end-users do care about the underlying values that are impacted by the default configuration of the design, unless they are aware of the possibility of choice they will accept the default, even if it is injurious to the value.[131]

To further complicate matters, the designer must make a choice as to how to balance the number of defaults (that is, options which the end-user can change) against the amount of 'pre-wired' functionality: too many options or a complex interface can confuse, undermining the benefit of providing a choice.[132] Indeed, providing configurable options within interfaces that are antagonistic to exercising that choice is one means by which some unscrupulous (or perhaps just negligent) enterprises can argue that they are respecting end-users' autonomy whilst simultaneously undermining their interests in

[124] L Tien, 'Architectural regulation and the evolution of social norms' (2004) 7 *Yale Journal of Law & Technology* 1, 16.

[125] Kesan and Shah (n 122) 596, 601; Diver (n 52) 11. See also I Kerr, 'The devil is in the defaults' (2017) 4 *Critical Analysis of Law* 91, 98 *et seq*.

[126] This is known as the 'legitimating effect'. See Kesan and Shah (n 122) 603. For a study showing empirical evidence of the effect, see RC Shah and C Sandvig, 'Software defaults as de facto regulation: The case of the wireless internet' (2008) 11 *Information, Communication & Society* 25.

[127] Kesan and Shah (n 122) 611–12.

[128] Ibid. 598 *et seq*.

[129] Tien (n 124) 16. An example of this was Facebook's controversial move to the 'news feed' layout as the default for all end-users. See J Leyden, 'Users protest over "creepy" Facebook update' *The Register* (7 September 2006) <https://www.theregister.co.uk/2006/09/07/facebook_update_controversy/> last accessed 4 March 2021.

[130] Tien (n 124) 12.

[131] Kesan and Shah (n 122) 601–2.

[132] Ibid. 627. See also Brownsword's discussion of 'prudential choice' in 'Lost in translation' (n 113) 1345.

favour of commercial expediency. A recent example is the design change in Google's Chrome browser that obfuscates the circumstances in which end-users are logged into Google's services, even when they have enabled the 'block third-party cookies' preference that would normally prevent this kind of behaviour (and indeed still does in other browsers).[133] It is worth noting, too, that the 'block third-party cookies' preference itself is not the default setting on any mainstream browser, and thus its privacy-enhancing mechanism is something end-users must manually enable, which requires first that they are aware of the option and what it means.[134]

Pervasiveness

The pervasiveness of code is not difficult to appreciate. Code is all around us, its presence at both infrastructural and artefactual levels increasing at what seems like an inexorable rate. The market analysis firm IDC, for example, expects there to be 41.6 billion Internet of Things (IoT) devices connected by 2025, a compound annual growth rate of almost 29 per cent.[135] Earlier this decade, Cisco estimated that by 2020 there would be 250 'things' connecting to the Internet every second, up from 80 per second in 2013.[136] IoT devices are being integrated further into daily life through the development of more sophisticated low-power infrastructure[137] and the diversification of connected

[133] L Olejnik, 'Am I logged in or not? GDPR case study on the example of Chrome browser change' <http://blog.lukaszolejnik.com/am-i-logged-in-or-not-gdpr-case-study-on-the-example-of-chrome-browser-change/> last accessed 4 March 2021.

[134] This is in line with the evolution of the relevant IETF standards, which have changed from requiring a higher standard of privacy (RFCs 2109 (1997) and 2965 (2000) suggest that the default should be to reject persistent (cross-session) cookies) to permitting browsers to implement whatever default standard they wish, albeit while noting the 'worrisome' nature of third-party cookies (RFC 6265 (2011), section 7.1). At time of writing, it appears that third-party cookies may be in the process of being phased out, at least in relation to behavioural advertising, the efficacy of which is beginning to be questioned.

[135] IDC Media Center, 'The growth in connected IoT devices is expected to generate 79.4ZB of data in 2025, according to a new IDC forecast' *IDC Media Center* (18 June 2019) <https://www.businesswire.com/news/home/20190618005012/en/Growth-Connected-IoT-Devices-Expected-Generate-79.4ZB> last accessed 4 March 2021.

[136] K Tillman, 'How many Internet connections are in the world? Right. Now' *Cisco Blogs* (29 July 2013) <https://blogs.cisco.com/news/cisco-connections-counter> last accessed 4 March 2021.

[137] J Twentyman, 'IoT drives progress towards low-power technology' *Financial Times* (8 January 2018) <https://www.ft.com/content/f2b4de5a-d8ee-11e7-9504-59efdb70e12f> last accessed 4 March 2021. A salient example at time of writing is the use of Bluetooth Low-Energy (BLE) in the tracing applications that have become a central element in tackling the COVID-19 pandemic.

applications.[138] Similar trends can also be seen in blockchain adoption and machine learning, themselves also so often amalgamated with the data that flows from IoT devices.

Regardless of the truth of such numbers, what seems intuitively clear is that the hype around the 'fourth revolution' is resulting in an ever-greater reliance on code and data infrastructure that seems unlikely to abate any time soon.[139] In the end, as Ihde puts it, our 'technologies invent us as we invent them',[140] and given the accelerated blurring of the line between the off- and the on-line it becomes ever more pressing to protect the ability of citizens to question the 'inventive' normativities that code imposes. If the pervasiveness of code means we are entering a new phase of the 'onlife', we must not in the process surrender our capacity to have a say in the ways that code is reinventing us. As we find our behaviour being channelled this way and that by the many code artefacts that play a role in our lives, we might in our more sober moments find the aggregate of all this technological normativity somewhat troubling. The point, however, is not to push from one extreme to the other, but rather to mandate checks and balances that can facilitate our autonomy in amongst both the good and the bad that this brings.

Immutability

Surden notes how the subjective value judgements of designers, and the resulting effects of the rules embodied in their code, can be magnified when the systems are distributed and adopted widely.[141] The choice of rules becomes fixed in the code, enabling its exponential magnification as its effects compound with concurrent and successive execution.[142] Bamberger notes similar risks in his discussion of 'systemic effects': whereas at production time the designer has great freedom to choose how the code should behave, this

[138] Forbes Technology Council, '14 predictions for the future of smart home technology' *Forbes* (12 January 2018) <https://www.forbes.com/sites/forbestechcouncil/2018/01/12/14-predictions-for-the-future-of-smart-home-technology/> last accessed 4 March 2021.

[139] Even a global pandemic has done nothing to halt this trend – in fact, at time of writing the worldwide lockdowns due to COVID-19 have significantly increased the role played by code infrastructures in our lives. See for example BBC News, 'UK's internet use surges to new highs in lockdown' *BBC News* (24 June 2020) <https://www.bbc.com/news/technology-53149268> last accessed 4 March 2021.

[140] D Ihde, *Ironic Technics* (Automatic Press/VIP 2008) vi.

[141] Surden (n 60) 5.

[142] Ibid.

plasticity is to a great extent 'locked down' once production has ceased.[143] This is what Winner terms the 'initial commitments'; for him,

> technological innovations are similar to legislative acts or political foundings that establish a framework for public order that will endure over many generations. For that reason, the same careful attention one would give to the rules, roles, and relationships of politics must also be given to such things as the . . . tailoring of seemingly insignificant features on new machines.[144]

Without some ready means of altering the code after it is produced, the normative importance of those 'initial commitments' is all the greater. These observations strengthen the impetus to focus on ex ante production of code rather than ex post assessments of it. Goldoni summarises why this is so:

> Given the nature and the logic of architectural regulation, the emphasis on output legitimacy is misplaced for several reasons. First, since technology is often irreversible – once it is developed and applied in society, it is difficult to change it or remove it from society in those applications – the process which develops code as law becomes a key concern when normativity is at stake. In fact, it may well be too late when a particular version of a technology appears or is adopted . . . The difficulty of reversing embedded code is often evident and makes it fundamental to focus on the procedure and the actors involved in [its] development.[145]

Even where the code can be updated, its immediacy means its normative effect is in place before this happens, and so it is important that the design is produced in a legitimate fashion from the outset. Code is of course updated all the time, but this does not alter the reality that at the point of compilation it is immutable, pending some change at some point in the future. The technological normativity imposed upon the end-user is precisely that which was defined in the latest update, and it will not change until the next one comes along. In that sense, code is immutable insofar as its normative configuration is fixed for whatever length of time. The ability to update it is contingent on

[143] Bamberger (n 112) 710–11. Krajewski refers to this as a moment of 'closure' (n 69).

[144] L Winner, 'Do artifacts have politics?' (1980) *Daedalus* 121, 128.

[145] Goldoni (n 122) 128. Koops makes a similar argument, although he focuses nevertheless on ex post assessment. See B-J Koops, 'Criteria for normative technology: The acceptability of "code as law" in light of democratic and constitutional values' in R Brownsword and K Yeung (eds), *Regulating Technologies: Legal Futures, Regulatory Frames and Technological Fixes* (Hart 2008) 166. I consider both his contribution and the important difference between input and output legitimacy in Chapter 5.

the design – it is not a given – but it also requires oversight by its creator, who may not have any commercial incentive to provide updates (so-called 'planned obsolescence' means the trade-off between creating profitable new features and maintaining older code is one that usually favours the former at the expense of the end-user's interests).

(d) Opacity

We saw in Chapter 2 how design operates in ways which are not always within the conscious apprehension of the end-user. This opacity forms another fundamental way in which code *de facto* requires end-users to 'not think about it': if the end-user cannot comprehend the rules to which her behaviour is subject, she cannot possibly consider whether and how to respond to them. This foundational issue is problematic in traditional processes of democratic law-making. Waldron, for example, notes that 'those interested in democracy will have a direct interest also in this opacity itself – that is, in the sheep-like ignorance of the nature of the law one is ruled by'.[146] So too in the computational context, except that there the extent to which end-users *qua* citizens are rendered 'sheep-like' is qualitatively and quantitatively greater.[147] As Goldoni notes, 'given the opacity of architectural regulation, to be aware of how technology is directly or indirectly impacting upon agents' behaviours may prove to be too difficult in many cases'.[148] Longford's observations in relation to web technologies are apposite:

> A central feature of new media design, in fact, is that the source code for any particular application or program which structures an end-user's experience is hidden from them . . . HTML, IP addresses, and web browser software are exemplary of code's self-concealing character. HTML conceals the textual information which is ultimately responsible for the graphical web pages presented to surfers.[149]

Whereas most browsers have a 'view source' option that makes HTML relatively accessible (if not necessarily comprehensible, despite its human-readability[150]) to the end-user, the compiled code that implements specific rules in other digital artefacts is generally both inaccessible and inscrutable

[146] J Waldron, 'Can there be a democratic jurisprudence?' (2009) 58 *Emory Law Journal* 675, 696 *et seq*., discussing the problematic nature of analytical positivism from a democratic perspective.

[147] Citron (n 123) 1254–5.

[148] Goldoni (n 122) 128.

[149] Longford (n 3) 82.

[150] This is the heart of the transparency fallacy, discussed in Chapter 6.

because of its translation (compilation) into machine-readable 'object code'. Regardless of the programming language used, however, end-users face difficulty in comprehending the totality of the system before them, and are in essence forced to accept a great deal on faith.[151] The user interfaces of digital artefacts are inherently limited in their communication of the myriad operations taking place behind-the-scenes. Even the apparently simplest of operations, for example clicking a hyperlink on a webpage, involves a host of unseen technical processes. Most of the time obscuring all of this is beneficial, in terms of avoiding the overwhelming cognitive load that trying to comprehend all that is really going on would entail. As mentioned above, it would be undesirable to enquire into the detail of every rule being followed in every computational operation, since the burden of comprehension is too great. The task, then, is to give information and control 'over the right things at the right time'.[152] This obfuscation of actual behaviour can, however, be used both for good and bad; the ability to hide the complexity of standard technical behaviours for the sake of the end-user can also be used to obfuscate technical behaviours that are antagonistic to her interests.[153]

We have seen on the one hand how end-users tend to accept defaults as-is, while on the other the immutable aspects of a system's architecture situate the end-user within an assemblage of behaviour-constraining rules that might admit of only minimal, if any, interpretation – a fait accompli, 'achieving compliance by default rather than through active enforcement'.[154] As Hildebrandt notes, whereas '[l]egal norms do not rule out disobedience, contestation of the technological defaults that regulate our lives may be impossible because they are often invisible and because most of the time there is no jurisdiction and no court.'[155]

The normativity of code is thus in no way contingent on it being intelligible to those whose behaviour is regulated by it (or indeed even those who created it). There is no requirement that its rules be made public, much less

[151] L Winner, *Autonomous Technology: Technics-Out-of-Control as a Theme in Political Thought* (2nd edn, MIT Press 1977) 284.

[152] B Friedman, 'Value-sensitive design' (1996) 3 *interactions* 16, 18. See also A Pols and A Spahn, 'Designing for the values of democracy and justice' in J van den Hoven, PE Vermaas and I van de Poel (eds), *Handbook of Ethics, Values, and Technological Design: Sources, Theory, Values and Application Domains* (Springer 2015). For a recent discussion of design sensitive to 'kairological time', see T Bucher, 'The right-time web: Theorizing the kairologic of algorithmic media' (2020) 22 *New Media & Society* 1699.

[153] See generally F Pasquale, *The Black Box Society: The Secret Algorithms that Control Money and Information* (Harvard University Press 2015).

[154] Tien (n 124) 12.

[155] Hildebrandt, *Smart Technologies* (n 11) 12.

in a form that can be understood by humans. Indeed, the complexity of code rules is such that they rapidly become unintelligible, even to the experts who created them. As discussed in Chapter 2, it thus becomes extremely difficult (and in most cases impossible) for the end-user to scrutinise the rules to which her behaviour is subject. We therefore have the blindest of legalistic 'blind rule following'.[156]

Code as a-legal 'Positivism'

What can hopefully be appreciated from the preceding discussion is the sheer 'is-ness' of code; its 'positivity' is of a form that is deeply challenging to a vision of law as a reasoned enterprise that reflects the democratically expressed outlook of a society, from and according to which its norms are articulated and subsequently interpreted and applied. As Vismann and Krajewski note, code's 'virtuality challenges the law's core concepts: corporeality, finitude, and authentication, concepts that are fundamental to any claim of territorial sovereignty as well as to imputations and rules of evidence'.[157]

This 'positivism'[158] removes the possibility of deliberation on the part of the end-user, resulting in a kind of instrumentalism that strips individuals of their ability to take part in the moral community, even where they disagree with what the code rule is making them do. End-users have no choice but to obey the rules, but simultaneously they have no standing in their formulation.[159] As Brownsword notes, design simply 'by-passes practical reason to eliminate all options other than the desired pattern of behaviour'.[160] One of the effects of this is to de-moralise citizens, blunting their sensitivity to social norms and thus their capacity for self-control and for doing good.[161] This latter point evokes Fuller's discussion of the morality of aspiration, and how it conflicts with a legalistic morality of duty according to which, as we have seen, the rule constitutes the entirety of what is required of regulatees,

[156] Bańkowski and Schafer (n 1) 31.

[157] Vismann and Krajewski (n 87) 92.

[158] Hoffmann-Riem for example refers to 'digital neo-positivism' in the context of techno-regulation. See W Hoffmann-Riem, 'Legal technology/computational law: Preconditions, opportunities and risks' (2021) 1 *Journal of Cross-disciplinary Research in Computational Law* <https://journalcrcl.org/crcl/article/view/7> last accessed 19 April 2021.

[159] Bańkowski and Schafer (n 1) 48. See also Longford (n 3).

[160] R Brownsword, 'Code, control, and choice: Why east is east and west is west' (2005) 25 *Legal Studies* 1, 4.

[161] Ibid. 19, quoting DJ Smith, 'Changing situations and changing people' in A von Hirsch, D Garland and A Wakefield (eds), *Ethical and Social Perspectives on Situational Crime Prevention* (Bloomsbury Publishing 2004). See also Brownsword, 'Lost in translation' (n 113) *passim*.

with no further expectation.[162] For Bańkowski, this minimal expectation is not enough; legalism and positivism collapse the aspirational aspect of legality and reduce the guiding normative value of social practices, because the presence of a 'critical morality' that invites self-scrutiny becomes displaced by a rote approach to the direction of citizens' conduct.[163] By removing the need to consider the proper course of action, our ability or habit of raising such questions becomes atrophied. Brownsword suggests that a community fully reliant on such forms of (state) regulation which obviate the possibility of moral deliberation is 'no longer an operative moral community'.[164] We require the opportunity to choose to do good, in the face of at least the possibility of doing otherwise, if we are to continue to exercise practical reason as moral actors.

I return to the topic of legality in Chapter 4, but for now, we can observe how computational legalism demonstrates the most certain of certainties, simultaneously hiding this from the end-user's comprehension under the veil of opacity. As we will see in Part III, the provision of spaces for the exercise of such reason and choice is not just about providing more choice, but is about providing the right quality of choice at the right time.

(e) The Veiling of Code's Production

Wintgens discusses the 'veil of sovereignty' that shrouds the work of the legislator, shielding it and her from the gaze of the legal philosopher and the citizen.[165] Legislative sovereignty is thus perceived as a black box.[166] The ultimate source of sovereignty is not in question and the mechanisms by which its outputs are arrived at are not to be questioned by jurists. Boyle noted this obscuring function early on when he suggested that '[t]he technology appears to be "just the way things are"; its origins are concealed, whether those origins lie in state-sponsored scheme or market-structured order, and its effects are obscured because it is hard to imagine the alternative.'[167]

[162] Fuller, *The Morality of Law* (n 14) chapter 1.

[163] Bańkowski (n 15) 49–50. Gardner resists this characterisation of legal positivism, although such arguments seem predicated on conflicting framings of the question – what counts as law versus what one ought to do with it. See J Gardner, 'Legal positivism: 5½ myths' in *Law as a Leap of Faith: Essays on Law in General* (Oxford University Press 2012).

[164] Brownsword, 'Code, control, and choice' (n 160) 19. See also Brownsword, 'Lost in translation' (n 113) 1355–6.

[165] Wintgens, *Legisprudence: Practical Reason in Legislation* (n 19) 2; Wintgens, 'Legislation as an object of study of legal theory' (n 29) 2.

[166] Wintgens, *Legisprudence: Practical Reason in Legislation* (n 19) 212 *et seq*.

[167] J Boyle, 'Foucault in cyberspace: Surveillance, sovereignty, and hardwired censors' (1997) 66 *University of Cincinnati Law Review* 177, 205.

In the computational context, the 'sovereignty' of the designer is protected by two 'veils' – one technical and the other legal. The first is the code-based opacity discussed in the previous section: this veil is one of technical inscrutability, which the ordinary end-user is usually ill-equipped to lift, if it is even possible to do so. The second veil protects enterprise by means of, for example, trade secrecy and anti-circumvention laws, which limit the scrutiny their code production practices can be put under, thus strengthening their quasi-sovereignty.[168] The prevailing neoliberal economic outlook sustains this second veil by shifting sovereignty away from the state and onto the market,[169] while simultaneously prioritising unfettered technological 'innovation' as a good in itself.[170]

The private 'sovereignty' of the profit-seeking enterprise is thus black-boxed unless and until a real-world harm is detected, which because of the technical veil might never happen. It is the task of the market and not the state to respond to the enterprise's designs in order to ascertain their value; meanwhile, the commercial entity is free to exercise its imperative for profit, while the market is trusted to curb any excesses. As Schulz and Dankert put it, 'code is essentially a resource through which the ones designing the code can pursue their interests'.[171] Similarly, Bańkowski and Schafer note that '[f]or the individual, more often than not, the absence of government is not experienced as liberating, but as subjugation to commercial interests which effortless [sic] project, and indeed magnify, their offline powers into cyberspace.'[172] Herein lies a paradox of commercial computational legalism: as discussed above, legalism is ideologically attractive in part because it helps establish a baseline of legal certainty which is advantageous to capitalist enterprise.[173] As those enterprises have developed into promulgators of code-based rules, however, their need for certainty has circumscribed the liberty of the ordinary citizen by the development of an imbalance of regulative power

[168] Pasquale (n 153) 15.

[169] BH Bratton, *The Stack: On Software and Sovereignty* (MIT Press 2016) 21.

[170] JE Cohen, 'The regulatory state in the information age' (2016) 17 *Theoretical Inquiries in Law* 369, 387–8. See also D Harvey, *A Brief History of Neoliberalism* (Oxford University Press 2005) 157 *et seq*.

[171] W Schulz and K Dankert, '"Governance by things" as a challenge to regulation by law' (2016) 5 *Internet Policy Review* 5 <https://doi.org/10.14763/2016.2.409> last accessed 19 April 2021. Hildebrandt and Koops term this a 'provider-centric scenario'. See M Hildebrandt and B-J Koops, 'The challenges of ambient law and legal protection in the profiling era' (2010) 73 *The Modern Law Review* 428, 457.

[172] Bańkowski and Schafer (n 1) 47.

[173] Wintgens, 'The rational legislator revisited' (n 16) 4; Bańkowski (n 15) 48.

between government and code,[174] and by the lack of incentives to ensure that their design processes and their products embody the values of legitimacy and legal protection intrinsic to that liberty. Hildebrandt and Koops suggest that cost, convenience, the difficulty of controlling risk, and the power imbalance between government and commerce all combine in the context of privacy to 'favour privacy-threatening technology far more than privacy-friendly "code"'.[175] In the absence of incentives to create the latter, code which is supportive of commercial interests but detrimental to end-users' interests is likely to win out; the features of computational legalism make it easy to take that route (and indeed 'rational', from an orthodox economic perspective). The same is true for the more fundamental form of user-antagonistic code I am concerned with. As with the aim of legality in law-making, we cannot appeal to market fundamentalism to prevent the creators of code from exploiting computational legalism to further their own ends – checks and balances need to be put in place at the design level.

3.3 Conclusion

We have seen how the characteristics of code come together to demonstrate a form of legalism that is significantly stronger than anything envisaged in the legal literature. With code we have the apex of legalism: from the end-user's perspective, code's architecture is 'just there', while simultaneously its constitutive nature defines what practices are possible, by definition ruling out all the possible alternatives that the plasticity of the code might otherwise have allowed. This, indeed, is one of the great ironies of code – a near-infinite range of design possibilities mean there is nothing in principle stopping it from being designed in a non-legalistic way, there are just few incentives to do so. Ultimately, whatever the (de)merits of the design, code confronts us with not just representationalism, but realism: it does not just represent reality, it actively constitutes it. The behaviour of the end-user is to a great extent structured and bound ex ante, and since she will in most cases not be aware of that binding she is, through no oversight or mistake on her part, forced to acquiesce blindly to the rules that are inscribed in the code: she is deprived of even the notional possibility of choosing whether or not to 'think about it'.

Taken together with the analysis in Chapter 2, the discussion here has set up the concept of computational legalism, strengthening the theoretical parallel between code and law as normative orders whose rules can be created in ways that are to varying degrees legalistic. By developing this parallel, I set up

[174] Hildebrandt and Koops (n 171) 444.
[175] Ibid.

the foundation for an analysis that imports into the design sphere measures that are intended to reduce or avoid legalism in the domain of law-making. Before I embark on that synthesis, we first move on to Part II, which does two things: first, Chapter 4 sets out the two legal theories from which I develop the digisprudential framework, and second, Chapter 5 reviews the existing legal literature on criteria for the use of code as a regulator.