1

Introduction

> The computer programmer, however, is a creator of universes for which he alone is the lawgiver . . . No playwright, no stage director, no emperor, however powerful, has ever exercised such absolute authority to arrange a stage or a field of battle and to command such unswervingly dutiful actors or troops.[1]

The focus of this book is at the under-explored intersection between law and design. The theory I am advancing is about how the behaviour-shaping rules embodied in software code can be designed in a legitimate way. My central claim is this: those who exercise the power to shape the behaviour of citizens in a democracy ought to do so legitimately, which implies both minimal standards for and limits to such exercises of power. We expect this of laws, so why not of code? Software is everywhere, touching and structuring almost every aspect of our lives in ways that are often more effective and direct than what law can, or indeed ought to attempt to, achieve. Despite this, code is subjected to very little scrutiny as to whether or not this is acceptable, both in individual cases and as a whole. This book challenges that status quo, arguing that code-based norms ought to be subject to tests of their legitimacy, and providing concrete ways to achieve this. In a democratic society, the regulation of citizens' behaviour – of whatever kind and from whatever source – ought to meet minimal standards of legitimacy to be acceptable. Code that is not legitimate should not be released – full stop – and we should not be shy about asserting this. This is especially true in a world where myriad troubles are contributed to by code, developed within an economic orthodoxy that is less concerned with curbing abuses of design power than it is with facilitating 'innovation'.[2]

[1] J Weizenbaum, *Computer Power and Human Reason: From Judgment to Calculation* (Freeman 1976) 115.

[2] R von Schomberg, 'A vision of responsible research and innovation' in R Owen, J Bessant and M Heintz (eds), *Responsible Innovation* (John Wiley & Sons 2013) 58; K O'Hara and

Questioning the legitimacy of code is necessarily an ex ante concern – that is, it must be done at design time, before the code operates in the world. This kind of analysis therefore requires an internal focus on the production of code, rather than just its operation, viewed externally. If lawyers are properly to grapple with the realities of how code regulates, we must embrace an analytical shift that takes into account not just its effects but also the practical realities of its production. This means we must consider the processes and tools that make up the 'legislature' where code is 'enacted', including software development methodologies and the integrated development environments (IDEs) where the text of code is actually written.[3] They are the point at which 'constitutional' protections can be built into the very fabric of the code.

Why 'digisprudence'? As a portmanteau of 'digital' and 'jurisprudence', it mirrors the concept of *legisprudence*, according to which the creation of legislative rules should be seen not as a purely political concern, 'fenced off' from the view of the jurist, but instead as an appropriate subject of both jurisprudential analysis and tests of legitimacy.[4] As with legislative norms, if code regulates behaviour then its behaviour-enabling and behaviour-constraining 'rules' ought also to be subject to such scrutiny. Digisprudence is thus to software rules as legisprudence is to legal rules: it asks how they are created, and whether or not they meet specific formal standards that can render them legitimate, whatever their 'substantive' purpose might be. This raises the practical question: can the standards that make a legal rule legitimate be imported into the realm of design to make a computational rule legitimate? My answer is that they can, and they must. This is a significant challenge that requires novel theoretical and practical translations between domains, but in the face of the ever-greater presence in our lives of potentially illegitimate code, it is one that must be faced sooner or later.

1.1 The Structure of the Argument

To tackle the various elements of this challenge, the book is organised in a roughly dialectical structure. Part I problematises code as a regulator, first conceptualising its regulative characteristics in terms of design theory and the philosophy of technology (Chapter 2) before conceptualising why, from a legal-philosophical perspective, those characteristics are problematic (Chapter 3). From that analysis I posit the notion of *computational legalism*,

---

M Hildebrandt, 'Between the editors' in M Hildebrandt and K O'Hara (eds), *Life and the Law in the Era of Data-Driven Agency* (Edward Elgar Publishing 2020) 37–40.

[3] These will be discussed in more detail in Chapter 7.

[4] L Wintgens, 'Rationality in legislation – legal theory as legisprudence: An introduction' in *Legisprudence: A New Theoretical Approach to Legislation* (Hart 2002) 2.

an extreme species of unreflective rule-following that code can so easily impose upon citizens. Part II sets out existing literature on what constitutes legitimate rule-making, both from a legal perspective (Chapter 4) and in terms of what standards might render code an acceptable form of regulation (Chapter 5). Following that analysis we can appreciate first what formal standards rules ought to exhibit to be deemed legitimate, and second what is absent from the current literature on 'code as law' that deals with this point, the lack of analysis of code production being the primary issue. Finally, the synthesis of the book comes in Part III, where I propose a framework of design standards (the *digisprudential affordances* set out in Chapter 6) that can be used to critique and guide the production of digital artefacts such that they are formally legitimate, whatever their intended use or commercial purpose might be. Throughout that discussion I consider the implications of digisprudence for contemporary technologies, in particular (but not limited to) blockchain applications and the Internet of Things. The framework set out in Chapter 6 is complemented by a discussion in Chapter 7 of various ways in which those standards might be facilitated in a guiding, 'constitutional' manner by the tools and processes of the design environment, from IDEs to development paradigms to programming languages themselves.

In that vein, I take a pragmatic view of code, asking what it in fact does, and how it is in fact made (indeed, the genesis of this study lies in my own experience as a web developer in a small design firm). By looking directly at the processes of code production, they might lose some of their mystique, and we as lawyers might in turn be empowered to ask some difficult but necessary questions.

Speaking of pragmatism and empowerment, I use 'designer' throughout as shorthand for all those involved in the production of code, which will include graphic designers, requirements engineers, programmers, testers, et cetera. I also refer throughout to the citizen as an 'end-user', that being the term often used in technical communities for those who are at the receiving end of the code norms I am concerned with. The term also draws attention to the citizen's position at the *end* of the product design process, and her relative lack of agency in shaping its output.[5] One could interpret this as a somewhat defeatist perspective, but in acknowledging the diminished position of the citizen there is contained a seed of hope: 'end' implies a middle and a beginning – points at which things might be done differently, and better. That is precisely the goal of digisprudence.

[5] S Gürses and J van Hoboken, 'Privacy after the agile turn' in E Selinger, J Polonetsky and O Tene (eds), *The Cambridge Handbook of Consumer Privacy* (Cambridge University Press 2018) 581.

1.2 Rebooting 'Code as Law'

Asscher frames the starting point of the enquiry like this:

> Code can present constraints on human behaviour that can be compared with constraints by traditional laws. We have argued that even though code is not law, in some instances it can be useful to ask the same questions about code regulation as we do about traditional regulation. Code as law must be assessed by looking at the results of regulation in terms of freedom and individual autonomy and compared to the balance struck by traditional law.[6]

Code is like institutional law in that it possesses normative force, but is also different from it in fundamental ways. Code is like legislation, in that it is created to achieve some purposive end; it is 'enacted'. Code is capable of violating rights, whilst simultaneously resisting the aspirations and oversight of the rule of law. And finally, there are two pivotal moments at which assessments of code can be made: ex ante at the point of production, or ex post at the point of operation. Each of these elements plays an important role in the argument I am presenting, which runs as follows.

When commercial enterprises produce the software code of digital artefacts, they necessarily create alternative normative orders that can replace institutional law as a primary source of behavioural regulation. Importantly, the private commercial contexts within which this code is created are not subject to the legitimising formal and procedural standards of law-making found in constitutional democracies. This means that, in the move from public to private rule-making, the resulting normative force of that code on behaviour risks being illegitimate, whether or not this is intended.[7] As the quote above suggests, the question then arises of whether formal standards of law-making might be imported into the *sui generis* 'legislature' of the commercial design environment, in order to ensure that the code produced there is legitimate.

My purpose in framing code in terms of legal legitimacy derives from the point, made above, that it can so readily augment and even supplant law as a regulator. Those who create code, whose work is shielded by the private context of its production, ought to wield the power they hold legitimately.[8] If notionally sovereign legislatures are bound by constitutions so that they

[6] 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) 86.

[7] E Bayamlıoğlu and R Leenes, 'The "rule of law" implications of data-driven decision-making: A techno-regulatory perspective' (2018) *Law, Innovation and Technology* 1, 12.

[8] K Yeung, 'Why worry about decision-making by machine?' in K Yeung and M Lodge (eds), *Algorithmic Regulation* (Oxford University Press 2019) 38 *et seq*.

cannot arbitrarily impose regulations on citizens' behaviour, then neither should this be possible for private enterprise, especially given the characteristics of code that render it more problematic to comprehend and to control than is text-based legislation, both qualitatively and quantitatively. We will see below and in Part I what those characteristics are and what effects they have.

In this context, the term 'code' connects of course to Lessig's seminal work on 'code as law'.[9] His original argument was that individuals, represented by the 'pathetic dot', are regulated not just by law, but also by three further regulatory 'modalities', namely social norms, the market, and architecture. In 'cyberspace', architecture is constituted by software, or code, as opposed to the physical architecture of the 'real' world. The 'architects' of that code therefore have significant power in cyberspace, and because of the greater instrumental potency of code than the other modalities to shape what is and is not possible in that 'place', those architects therefore have disproportionate power within the digital realm. As Lessig puts it, '[a]rchitecture is a kind of law: it determines what people can and cannot do. When commercial interests determine the architecture, they create a kind of privatized law.'[10] Given the power of code to define the rules of behaviour in cyberspace, and given the inherent flexibility of designers to choose those rules, his fear was that they might be captured by state interests mandating backdoors and other measures antagonistic to civil liberties. His general prescription to avoid this was a culture of transparency, including actual transparency, in the form of open source code.[11]

The focus on transparency betrays a dependency on classic-liberal market orthodoxy that seems to obscure both the processes of code production and its ultimate embeddedness within society. The myriad effects of code – possibly good, possibly bad, but certainly never neutral[12] – stretch far beyond the relationship between the classical *homo economicus* and the 'trader' who sells or licenses the code. To fully appreciate this requires consideration of complementary insights deriving from scholarly fields including the philosophy of technology and science and technology studies (STS).[13] The Lessigian

[9] L Lessig, *Code: Version 2.0* (Basic Books 2006) *passim*.

[10] Ibid. 77.

[11] Ibid. chapter 5.

[12] M Kranzberg, 'Technology and history: "Kranzberg's Laws"' (1986) 27 *Technology and Culture* 544, 545–6.

[13] As Cohen suggests, without the latter 'one cannot explain how code regulates'. See JE Cohen, *Configuring the Networked Self: Law, Code, and the Play of Everyday Practice* (Yale University Press 2012) 27 *et passim*. See also V Mayer-Schönberger, 'Demystifying Lessig' (2008) *Wisconsin Law Review* 713.

framework does not include such perspectives, nor does it include the intimately connected topic of design practice – how code is actually made.

Lessig's analysis was extremely valuable in opening legal eyes to the potential roles played by technology in regulating individual behaviour, but it was skewed towards a particular view of law and was thus limited in the scope of normative effects and forms of technology it encompassed. By seeking to ensure that code provide market signals through transparency, Lessig's prescriptions can be boiled down to a call for this meta-technology (code) to be adapted to fit a particular market-oriented orthodoxy, driven by an instrumental view of regulation rather than a concern for legal protection and human flourishing. The assumption was that that economic orthodoxy was and is a given, this assumption in turn obscuring the deeper level to which Lessig's own mode of analysis might be extended – the nature of law and the rule of law as the necessary substrate for *any* kind of government and economic system within a constitutional state. Such a view would implicate a broader set of technologies whose mediations in turn affect (and effect) the relationships between the multifarious actors in society.[14]

Lessig's analysis has towered over the 'cyberlaw' landscape for over two decades, and as welcome as it has been in raising important questions about the nature of the relationship between law and code, there is now a need to reframe the topic away from his original assumptions. We are seeing no let-up in the deployment of code-driven systems, both as regulatory tools *per se* and as the building blocks of what is sometimes called the 'onlife'. There is little reason to suppose that this trend will reverse or even decelerate, and so the question is how best to respond. In that vein, in recent years we have seen a number of initiatives aimed at regulating (big) technology, from the European Commission's new proposed Regulations on artificial intelligence and the digital single market,[15] to the United States' proposed Deceptive Experiences To Online Users Reduction (DETOUR) Act, aimed at regulating 'dark patterns'

[14] For a pivotal discussion in this vein, see M Hildebrandt, *Smart Technologies and the End(s) of Law: Novel Entanglements of Law and Technology* (Edward Elgar Publishing 2015).

[15] The draft of the former is already creating waves in industry and academia. See European Commission, 'Proposal for a Regulation on a European approach for Artificial Intelligence' (European Commission, 2021) <https://digital-strategy.ec.europa.eu/en/library/proposal-regulation-european-approach-artificial-intelligence> last accessed 23 April 2021. This complements the Commission's two digital single market Regulations – the Digital Services Act and the Digital Markets Act – which are aimed at digital platforms and which will update the E-Commerce Directive (2000/31/EC). See European Commission, 'Shaping Europe's digital future' (European Commission) <https://digital-strategy.ec.europa.eu/en> last accessed 23 April 2021.

in digital product design,[16] to the EU General Data Protection Regulation (GDPR), whose reshaping of the digital landscape is evolving day by day.

Although my interest here is complementary to these sectoral responses, in some crucial respects my focus lies behind or before them. The goal here is to ask a fundamental question about the legitimacy of regulative code in the onlife, which means revisiting the notion of 'code as law' with a fresh willingness to embrace the insights of those other scholarly fields concerned with technology and its effects in the world. Pursuing new intellectual directions can be daunting, especially across disciplinary boundaries, but it can also be hugely valuable and, in this context at least, it is essential.

The nature of code necessarily implies a kind of temporal front-loading: code is designed and implemented before it is out in the world, its effects 'out there' being predetermined, at least in broad structure if not always in every atomic detail. This means we must engage directly with the practices of those who build it, understanding them from an internal perspective insofar as that is necessary to transplant our lawyerly critiques from the anodyne ex post into the incisive ex ante.

Lawyers cannot rely solely on calls for 'greater regulation', especially if the latter is uninformed by knowledge of or appreciation for design practice and the rich philosophical discussions of those things they are arguing ought to be regulated. Bringing all this together will be a crucial challenge for the twenty-first century, and so the time is ripe for a reboot of this twenty-year-old debate, clearing some of the dusty files that have been cluttering our memory so that we can get a renewed appreciation of what the fundamental questions are and how we might start to answer them.

(a) Code and/or Data?

It is true to say that the literature has evolved somewhat since the 'code as law' concepts were first introduced by Reidenberg in his analysis of *lex informatica*.[17] The literature was initially concerned primarily with the regulation of the amorphous cyberspace as a location that is 'out there', that is the Internet as a platform and a 'place'. The discussion has since evolved to consider on the one hand the code of individual and/or networked applications, and on the other code that facilitates data-driven services based on machine learning. Both forms of code are 'algorithmic', but the distinction between the two is

[16] See S.1084 – 116th Congress (2019–2020): Deceptive Experiences To Online Users Reduction (DETOUR) Act (4 September 2019) <https://www.congress.gov/bill/116th-congress/senate-bill/1084> last accessed 4 March 2021. I discuss dark patterns in Chapter 2.

[17] JR Reidenberg, 'Lex informatica: The formulation of information policy rules through technology' (1997) 76 *Texas Law Review* 553.

an important one,[18] albeit that they will often now be complementary, with the outputs of data-driven software feeding into the ordering imposed by code-driven architectures.

Broadly speaking, modern data-driven applications are concerned with the use of machine learning algorithms and 'big data' to facilitate automated classification and decision-making. Such systems are based on the processing of vast, often contingent datasets using mathematical models – compressed representations of the so-called 'ground truth' that the model is intended to identify, or 'predict', in any new data that it is provided with. By identifying a sufficiently similar pattern in that new data, such models can assist in classifying unforeseen examples, a function that has been employed across a wide range of applications, from distinguishing pictures of dogs to classifying offenders according to whether or not they are likely to reoffend.[19] Faith in the existence of underlying patterns in data that accurately represent human truths, and the ability of mathematical models to identify those patterns, has led to the use of machine learning in domains, including the law, where one might have thought mathematical reduction was anathema;[20] the unintelligibility of machine learning models, even to experts,[21] has not deterred some of

[18] M Hildebrandt, 'Algorithmic regulation and the rule of law' (2018) 376 *Philosophical Transactions of the Royal Society A* 20170355, 2–4.

[19] On the latter, see the influential J Angwin et al., 'Machine bias' *ProPublica* (23 May 2016) <https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing> last accessed 4 March 2021. The dog example highlights one of the fundamental problems at the centre of machine learning applications: MT Ribeiro, S Singh and C Guestrin, '"Why should I trust you?": Explaining the predictions of any classifier' (2016) arXiv:1602.04938 [cs, stat] <http://arxiv.org/abs/1602.04938> last accessed 4 March 2021. While the misidentification of a husky as a wolf might at first blush seem a rather insignificant 'mistake', the implications are profound for other contexts where the same or similar machine learning approaches are used, for example credit scoring, facial recognition, and the aforementioned prediction of recidivism (to name only a few examples). The literature on these implications is large and growing all the time; see for example J Buolamwini and T Gebru, 'Gender shades: Intersectional accuracy disparities in commercial gender classification' (2018) 81 *Proceedings of Machine Learning Research* 1; SU Noble, *Algorithms of Oppression: How Search Engines Reinforce Racism* (New York University Press 2018); C O'Neil, *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy* (Crown 2016).

[20] D McQuillan, 'Data science as machinic neoplatonism' (2018) 31 *Philosophy & Technology* 253. For various examples in the legal domain, see for example MA Livermore and DN Rockmore (eds), *Law as Data: Computation, Text, and the Future of Legal Analysis* (SFI Press 2019).

[21] The literature on algorithmic opacity is significant. See for example E Bayamlıoğlu, 'On the possibility of normative contestation of automated data-driven decisions' in I Baraliuc et al. (eds), *Being Profiled: Cogitas Ergo Sum – 10 Years of Profiling the European Citizen*

a more computationalist bent from pursuing these applications, despite the many risks involved.

A contrast is sometimes drawn between the contingent mathematical algorithms of machine learning applications and the predetermined, logical 'if–then' structure of code-driven software. This may be a false dichotomy, given that data-driven applications are precisely that – applications – operating at a level above code and requiring its foundation for their very existence. The general-purpose computing infrastructure that gathers, transmits, and stores the data that machine learning algorithms process is ultimately based entirely on that general-purpose computing infrastructure, and the algorithms themselves are expressed in code. The issues raised by data-driven applications will in many cases be distinct from those that are purely code-based, but in many cases the implications for the mediated experience of the end-user will be the same, and will raise the same questions of legitimacy, even if the ways of answering those questions will differ in terms of design practice.[22]

As is common in the literature, I use 'code' interchangeably with 'software' and 'architecture', to refer generally to digital systems that have a regulating effect on action and behaviour. This applies whether or not the artefact in question is built around machine learning. Throughout the book, the term 'code' is intended to be contrasted with law as a competing regulator. In the later chapters, however, I do shift to consider code *per se* – its text, rather than the architectures that that text brings into being – as one crucial site of digisprudential enquiry.

(b) Regulation

'Regulation' in this context straddles two of the definitions identified by Black, namely (1) the promulgation of rules by government (posited laws and regulatory instruments), and (2) all mechanisms of social control affecting behaviour, of whatever kind from whatever source, whether intentional or not.[23] The phrase 'social control' in the second of these might be viewed as including 'technical control' or 'commercial control'. This notion of control is connected with the definition of 'normativity' that I adopt, which is to say any mechanism, legal or otherwise, through which action and behaviour are

---

(Amsterdam University Press 2018); J Burrell, 'How the machine "thinks": Understanding opacity in machine learning algorithms' (2016) 3 *Big Data & Society*.

[22] The question of affording transparency, for example, arises whether or not the application is code- or data-driven; the emphasis may change but the requirement does not. I discuss this in more detail in Chapter 6.

[23] J Black, 'Critical reflections on regulation' (2002) 27 *Australian Journal of Legal Philosophy* 1, 11.

enabled or constrained. As Goldoni puts it, 'code as law is normative in the sense that it regulates and guides human behaviour'.[24] Of course, the technological species of normativity is different from legal normativity in crucial ways,[25] which is precisely what makes the present analysis necessary. I set out those mechanisms in greater detail in Part I of the book.

(c) Legitimacy

By 'legitimacy', I refer to the idea that rules that govern behaviour ought to be created according to pre-existing standards that embody values of accountability, transparency, and contestability.[26] Despite a large literature on software as both a target and a conduit of regulation, the question of legitimacy is one that has received only minimal attention with regard to the normative standards to which the designers who create code might be held.[27] Very few scholars have considered the question directly, and the treatment so far has focused more on ex post assessments of code regulation (the effects of its operation in the world) rather than on ex ante normative standards (questioning how it was produced and whether formal standards have been met). Although the former are an important and necessary element of oversight, the characteristics of computational legalism make necessary an additional focus on the application of ex ante standards during the production process. The reasons for this are discussed below and in detail in Chapter 3.

The legal-theoretical analysis of legitimacy that I adopt builds in particular on Fuller's *internal morality of law* and Wintgens's *legisprudence*. The latter is less well-known, but provides us with a serious, historically grounded theoretical inroad into the formal qualities that legislative rules ought to have, a topic so often bracketed by legal theorists concerned mainly with adjudication. I briefly summarise both theories later in this introductory chapter, before returning to them in more detail later in the book. These ideas help in conceptualising the ex ante assessment of other forms of normative rule-making, my contention being that many of the factors that render

[24] 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) 119. This meaning is similar to the concept of 'governance' in the regulatory literature. See C Reed and A Murray, *Rethinking the Jurisprudence of Cyberspace* (Edward Elgar Publishing 2018) 140.

[25] See for example N MacCormick, *Institutions of Law: An Essay in Legal Theory* (Oxford University Press 2007) chapter 1. On the essential distinction between orthodox and technological normativity, see M Hildebrandt, 'Legal and technological normativity: More (and less) than twin sisters' (2008) 12 *Techné: Research in Philosophy and Technology* 169, 173–5.

[26] See for example J Waldron, 'Can there be a democratic jurisprudence?' (2009) 58 *Emory Law Journal* 675.

[27] Goldoni (n 24) 123–5.

legislative rules illegitimate according to those theories can also be found in privately ordered code as law. By adapting and importing their principles into the design process, the illegitimacies of computational legalism can thus be mitigated.

(d) Code is both More*, and* Less*, than Law*

One of the main criticisms of Lessig was that code is not law, and therefore the perceived attempt to equate them was in some sense fallacious. For my part I agree that they are not the same, but my conviction is that we cannot simply stop there – to dismiss code as being 'not law' is to blind oneself to the central importance of its role in structuring society and our individual interactions, and thus to the fundamental questions of legitimacy, legal and political, that this raises.

To be sure, there are indeed overlaps and 'structural homologies'[28] between code and law, but so too are there significant differences. Code instantiates law, legal effect, and constellations of legally relevant fact, and while code-based artefacts are themselves constituted to some extent by legal reality (contracts, intellectual property rights, etc.), the relationship is lopsided. There exists an inherent 'hermeneutic gap' between the legal norm printed on the page and its instantiation in the physical world via interpretation and behavioural change.[29] In the computational context, law is not nearly as powerful as we might suppose, because it is dependent upon the very medium it is attempting to regulate, and the immediacy and instrumental power of that medium and the 'sovereignty' of the designer in shaping its effects tip the balance against law as the 'apex' regulator. The written law is rendered 'a paper dragon in the age of the "digital tsunami"',[30] with the social and rhetorical power of legal fictions making way for the representationalism of 'digital virtuality', whereby reality is constituted by and through the machine.[31] Adjudication is thus collapsed into obedience,[32] since the rule in the code also represents reality for the end-user.

[28] C Vismann and M Krajewski, 'Computer juridisms' (2007) *Grey Room* 90, 92.

[29] A foundational discussion of this temporal gap in hermeneutics can be found in H-G Gadamer, *Truth and Method*, trans. J Weinsheimer and DG Marshall (Bloomsbury 2013) chapter 4. This ultimately translates into the affordance of delay, discussed in Part III.

[30] 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, 440.

[31] Vismann and Krajewski (n 28) 92.

[32] Z Bańkowski and B Schafer, 'Double-click justice: Legalism in the computer age' (2007) 1 *Legisprudence* 31, 48.

Fuller defines law widely as 'the enterprise of subjecting human conduct to the governance of rules'.[33] As Chapter 2 will demonstrate, the governing rules that code subjects human conduct to increasingly constitute the very 'terms and conditions of existence and action'.[34] They may not be rules as commonly understood,[35] but they are designed by humans with a purpose in mind, and should therefore be subject to scrutiny as to their legitimacy. The power to decide those purposes is significant:

> The quasi-sovereign power of the computer engineer's code stems from the ease by which posing, implementing, and applying a norm are achieved in technology compared with the cumbersome procedures that legal code must pass through. The swift effectiveness of a technological code, which cannot, when seen through legal eyes, appear as anything other than uncanny, renders any possible competition between law and computer pointless.[36]

Architectural constitutions supplant legal constitutions; code is not just law-like, rather it is both more, and less, than law. As Chapter 2 will demonstrate, it is more than law because of the instrumental power of design to constitute and regulate end-user action and behaviour. But it is simultaneously less than law because, as Chapter 3 explains, it lacks the normative mechanisms designed to keep its textually bound sister in check. This is what Hildebrandt points to when she says that 'technologies that are constitutive for [sic] our interactions may enforce compliance beyond anything that a written law can achieve'.[37] It is precisely because code is not law *per se*, but nevertheless has a power to regulate that is more direct and effective than that of law, that it is necessary to instantiate the sorts of constitutional protections I will be discussing. While code constitutions are not law under any orthodox definition, if we adopt a pluralist perspective[38] we can identify, through a comparison of the regulative aspects of institutional law and code, which of the checks and balances that we expect to be present in the former are absent from the latter.

[33] LL Fuller, *The Morality of Law* (Yale University Press 1977) *passim*.

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

[35] I discuss the question of code as rules in more detail later.

[36] Vismann and Krajewski (n 28) 93.

[37] Hildebrandt, 'Legal and technological normativity' (n 25) 178.

[38] MAC Dizon, 'From regulating technologies to governing society: Towards a plural, social and interactive conception of law' in HM Morgan and R Morris (eds), *Moving Forward: Tradition and Transformation* (Cambridge Scholars Publishing 2011). Dizon argues that '[w]hen Lessig uses his four modalities of control to describe the normative orders of cyberspace, he is in fact describing the condition of legal pluralism in the ICT field'.

Whereas traditional regulative norms derive their legitimacy from the institutions and traditions of the rule of law within constitutional democracy, code-based norms have no such necessary democratic provenance or oversight. Whereas legal normativity invites the citizen to comply (she always has the notional option to interpret the norm, contest it, or to ignore it entirely), technological normativity can make compliance a necessity, either in the form of imposing a response to a circumstance or by constituting at the outset all the courses of action that the end-user can possibly take.

The fact that code is not law *per se* is therefore no answer to the problem I am concerned with; as Fuller demonstrates in his discussion of the rules governing a college dormitory, law-systems exist in many contexts that have no explicit or implicit connection with the state[39] – what matters, at least for present purposes, is whether the subjection of human conduct to the governance of rules is legitimate or not. The materiality of that governance is, in the end, what matters: as Le Sueur suggests, 'we should treat "the app" (the computer programs that will produce individual decisions) as "the law" . . . It is this app, not the text of the legislation, that will regulate the legal relationship between citizen and state in automated decision-making.'[40] Precisely because of the supreme efficacy with which code achieves this regulation, it is imperative that the creators of private code are, like public law-makers, constrained by ex ante standards that ensure both legitimacy during operation and the possibility of ex post remediation. Whether or not these are in place is ultimately a question of design, and thus of production.

Design and Regulation

How is behaviour in practice enabled and constrained by code? Numerous concepts from design and the philosophy of technology can help us frame an answer to this question, in particular the notions of inscription, affordance, and technological mediation, each of which is discussed in more detail in Chapter 2. Inscription is the notion of embodying in the design of an artefact a particular 'story' that dictates what the end-user ought and ought not to do.[41] Many of these scripts are so embedded as to become second nature;

[39] Fuller (n 33) 125 *et seq*.

[40] A Le Sueur, 'Robot government: Automated decision-making and its implications for Parliament' in A Horne and A Le Sueur (eds), *Parliament: Legislation and Accountability* (Hart 2016) 201. Le Sueur's analysis concerns public administration, but his insight applies to private code too.

[41] M Akrich, 'The de-scription of technical objects' in WE Bijker and J Law (eds), *Shaping Technology/Building Society: Studies in Sociotechnical Change* (MIT Press 1992) 208; B Latour, 'The Berlin key or how to do words with things' in P Graves-Brown (ed.), *Matter, Materiality and Modern Culture* (Routledge 2000).

if you are reading this electronically, for example, consider how easily you 'tapped' or 'double-clicked' the 'icon' to 'open' the 'file'. However natural or 'ready to hand' for us these sorts of scripted concepts and processes might have become, they are none of them given; each has to whatever extent been purposively designed.[42] Of course, this idea of channelling or 'tunnelling'[43] behaviour can be used for different ends, but whatever the choices made by the designer, these invariably mean other possibilities are left out that might otherwise have been built. What is left in will constitute the affordances of the artefact, or the ways in which it can be used by a particular end-user, given her characteristics and those of the code in question.[44] Although many affordances are contingent relationships between user and artefact, they are often consciously designed as features of the system, in which case they will (usually) be signified to the user.[45] A common example is a pad on the surface of a door that signifies the affordance of pushing (but not of pulling).

In contrast to the enablement of behavioural possibilities that designed affordances and their signifiers represent, the concept of *disaffordance* points, in the design context at least, to the conscious and strategic choice to 'enforce or restrict certain user behaviour'.[46] This builds on Lessig's notion of 'architectures of control',[47] and is of course central to the claim made here about the (il)legitimacy of such technological normativity.

Code is designed with a particular class of user in mind, and so its (dis)affordances, inscriptions, and mediations are all fundamentally affected by the directed choices made by the designers who produce it. Although some forms of action are emergent or open to (re)interpretation or resistance on

[42] Lessig hints at this truth when he notes that 'there is no choice that does not include some kind of building. Code is never found; it is only ever made.' See Lessig (n 9) 6. For present purposes, Heidegger's notion of 'ready-to-hand' captures an important aspect of the individual's situatedness in-the-world, constituted seamlessly – at least until something breaks – by the environment they inhabit. See M Heidegger, *Being and Time*, trans. J Macquarrie and E Robinson (Blackwell 1962) 95–102.

[43] On the latter, see BJ Fogg, *Persuasive Technology: Using Computers to Change What We Think and Do* (Morgan Kaufmann Publishers 2003) 34 *et seq*.

[44] DA Norman, *The Design of Everyday Things* (MIT Press 2013) 11.

[45] Ibid. 13 *et seq*.

[46] D Lockton, 'Architectures of control in product design' (2006) *Engineering Designer: The Journal of the Institution of Engineering Designers* 28. See also D Lockton, 'Disaffordances and engineering obedience' *Architectures* (22 October 2006) <http://architectures.danlockton.co.uk/2006/10/22/disaffordances-and-engineering-obedience/> last accessed 4 March 2021.

[47] Lessig (n 9) chapter 4.

the part of the end-user,[48] it is nevertheless true to a greater or lesser degree that design choices embed 'programs of action'[49] within the artefact, and so significant normative power inheres in those who make those choices. When a designer embeds (dis)affordances in the design of her artefact, she affects what it is possible to do with that artefact, either expanding or contracting those possibilities.

All of this points to the ways that designers fashion the geography of the artefacts they create, thereby controlling, at least to the extent it plays a role in her experience, that part of the user's mediated reality.[50] The extent to which that control is imposed will differ depending on the artefact and how far it exemplifies the elements of computational legalism.

Computational Legalism

One of the central problematics of code from a legal-theoretical perspective is its 'ruleishness', meaning its application of defined rules in all instances where fixed conditions, specified in the code itself, obtain.[51] 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 and controlled conditions, providing a predictability that lies at the centre of the technological advances seen in the silicon age.

In the legal context, however, the rote application of rules is undesirable, at least in a society built around the ideals of democracy and the concept of legality, where the system of rules must have the capacity to interface contingently with its context (that is, the society it serves). Linked with Kant's categorical imperative, *legalism* is the legal equivalent of code's ruleishness. Although it has more than one form in the literature, the relevant conception for my purposes is connected closely with certain forms of legal positivism,[52] and is seen as an ideology under which rules and the strict adherence to them

[48] This relates to Ihde's notion of *multistability*. See D Ihde, *Technology and the Lifeworld: From Garden to Earth* (Indiana University Press 1990) 144 *et seq*.

[49] 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).

[50] On the technological mediation of experience, see P-P Verbeek, *What Things Do: Philosophical Reflections on Technology, Agency, and Design* (Penn State Press 2005) chapter 3. See also Ihde (n 48), particularly chapter 5.

[51] J Grimmelmann, 'Regulation by software' (2005) 114 *The Yale Law Journal* 1719.

[52] See 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); JN Shklar, *Legalism* (Harvard University Press 1964) 7.

are the proper fundaments of social ordering. That the state defines what is legal is enough to legitimise the substance of the legal norms it chooses to 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. Constitutive facts (natural laws, the social contract/constitution, or a mix of these) operate prospectively to legitimise any subsequent act of the sovereign.[53] The citizen is given the imperative to 'not think about it'; the rule is 'just there' and she need only act in accordance with it as written,[54] since by virtue of those constitutive facts the pronouncement of the sovereign is 'imputed to [the people], as if they were its author'.[55] As an outlook, then, legalism tends towards a 'narrow governance of rules, unleavened by the principled approach to interpretation'.[56] This simplicity implies the possibility of abuse: the prioritisation of heteronomy militates against critical reflection and the application of other normative principles of legality that are aspirational characteristics in a democracy. The freedom of the citizen to interpret is seen as a crucial aspect of legality, without which rules become 'implements of tyranny' and legalism a 'vice of narrow governance'.[57]

From this brief summary of legalism (I will expand on the concept in Chapter 3), one can begin to appreciate how code can exemplify these characteristics.[58] In even the most tyrannical state there is space to interpret, and perhaps to disobey, the law – the hermeneutic gap between the text of a norm on the page and its translation into behaviour in the world makes this at least a notional possibility. In the environments where code is designed, however, the elision of that gap is not only easy to do but is entirely standard, not necessarily through malice or intentional obfuscation (though they are of course a problem), but simply by virtue of the ontological characteristics of code, which presents norms to the end-user that 'just are'. Even where the code does allow for choice via configuration, the default settings of code tend to be viewed by end-users as 'a natural and immutable fact'.[59] The hermeneutic gap

[53] L Wintgens, *Legisprudence: Practical Reason in Legislation* (Routledge 2012) chapters 5–6.

[54] 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).

[55] Wintgens, *Legisprudence: Practical Reason in Legislation* (n 53) 208.

[56] Bańkowski and MacCormick (n 52) 194.

[57] Ibid.

[58] Bańkowski and Schafer (n 32).

[59] Goldoni (n 24) 128. Boyle also hinted early on at this 'legalistic' nature of code, noting 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.' See J Boyle, 'Foucault in

is thus closed, or at least significantly narrowed, because the 'text' of the 'rule' (the source code) constitutes directly the geography of the artefact: they are not just isomorphic, they are one and the same. Unlike traditional law, whose 'carrier' has hitherto been the inherently passive medium of text, software code allows us to 'conceive of a text (a programming language) that is at once words and actions'.[60] This represents the apex of legalism: the normative collapses into the descriptive (what was once requested becomes simply what is), and there is no choice but to obey the rule as it is expressed by the designer, much less to view and contest it, since it by definition constitutes empirical as well as legal and technological reality.[61] The characteristics of computational legalism – ruleishness, opacity, immediacy, immutability, and pervasiveness, all compounded by privatised production – mean that in many cases code is simultaneously more powerful and less adaptable than a law-system that is built around the characteristics of delay, flexible interpretation, and ex post remediation. Code is thus simultaneously more, and less, than law.

Digital Rights Management

Consider for a moment digital rights management (DRM), a well-studied form of regulative code and a staple of technology law analysis. As I discussed above, it is important to distinguish between compliance with substantive law (generally but not necessarily copyright, in the case of DRM[62]), and broader and more fundamental questions of legitimacy. The computational legalism of DRM is exemplified by the Sony BMG scandal of the mid-2000s,[63] where the record company included DRM software on its CD releases that was designed to limit the scope of playback and the ability to 'rip' the music as digital files or copy it to a blank CD. The software installed itself surreptitiously on end-users' Windows PCs: upon insertion of the CD, if the code detected existing CD copying software installed on the computer, it would

---

cyberspace: Surveillance, sovereignty, and hardwired censors' (1997) 66 *University of Cincinnati Law Review* 177, 205.

[60] Latour, 'Where are the missing masses?' (n 49) n 1. The nature of programming languages as a source of rules is something I consider in Part III.

[61] Bańkowski (n 54); Bańkowski and Schafer (n 32). Representationalism is a key element of the legalistic outlook. See L Wintgens, 'Legisprudence as a new theory of legislation' (2006) 19 *Ratio Juris* 1, 5. I consider the contrast between regulative and constitutive normativity in Part I.

[62] MJ Radin, 'Regulation by contract, regulation by machine' (2004) 160 *Journal of Institutional and Theoretical Economics (JITE)* 142, 152. See also Lockton, 'Architectures of control in product design' (n 46).

[63] See for example BBC News, 'Sony slated over anti-piracy CD' *BBC News* (3 November 2005) <http://news.bbc.co.uk/1/hi/technology/4400148.stm> last accessed 4 March 2021.

cease playback and eject the disc.[64] Any copies made using the system were themselves protected by the same restrictions.[65] Playback was also limited to the included software.

Of course, Sony BMG was a commercial enterprise, a fact that frames the other 'legalistic' characteristics of the system. Self-evidently, the system's code regulated what the end-user could do with her purchased CD. Legitimate playback and copying on a PC were severely constrained. The system was opaque in its operation: those limits on playback and copying were not made clear from the outset, nor did the system notify the end-user that it would install the DRM prior to her consenting (and, incredibly, even if she withheld consent[66]). The license agreement failed to narrate these limitations accurately,[67] and in any event there could be no reasonable expectation in that context that the system would seriously undermine both the security of the end-user's PC and her individual privacy.[68] The system's immediacy was demonstrated by the nature of its installation – those without deeper technical knowledge had no opportunity to refuse its installation, despite the hermeneutic gap implied by the need to accept the license agreement. The cumulative normativity of the system was felt most by those least likely to attempt to circumvent it: infringers were more likely to be technically adept and therefore capable of side-stepping the DRM, while lawful but less technically literate end-users had their rights and convenience circumscribed despite not wishing to engage in unlawful copying.

As mentioned, the system was imposed upon the end-user without consent or choice. It was included on a medium whose contents cannot be changed once produced, leaving problematic code dormant on unchangeable media for a potentially unlimited time.[69] The design of the system had no anticipated means of altering the software after-the-fact; as the scandal gained prominence Sony BMG rushed to release patches that purported to

[64] JA Halderman and EW Felten, 'Lessons from the Sony CD DRM episode' in *15th USENIX Security Symposium* (USENIX Association 2006) 80.

[65] Ibid. n 8.

[66] Ibid. 81.

[67] DK Mulligan and A Perzanowski, 'The magnificence of the disaster: Reconstructing the Sony BMG rootkit incident' (2007) 22 *Berkeley Technology Law Journal* 1157, 1162.

[68] Ibid. 1211. On the privacy implications, see Mark Russinovich, 'More on Sony: Dangerous decloaking patch, EULAs and phoning home' <https://techcommunity.microsoft.com/t5/windows-blog-archive/more-on-sony-dangerous-decloaking-patch-eulas-and-phoning-home/ba-p/723452> last accessed 4 March 2021.

[69] As Halderman and Felten note, '[i]f a particular version of DRM software is shipped on a new CD, that software version may well try to install and run decades after it was developed.' See Halderman and Felten (n 64) 89.

uninstall the software, but in fact these caused further serious security problems.[70] Lastly, the system achieved significant distribution, if not pervasiveness: up to two million users were affected,[71] and in the fallout of the crisis around 7.3 million CDs were recalled.[72]

As I have argued, these characteristics of the code's normativity can be critiqued separately from its implementation of the substantive norms of copyright law.[73] Precisely because of the formal illegitimacies identified above, the ability of the end-user to be aware of and contest the mis-implementation of substantive copyright law was severely limited. By the standards of digisprudential legitimacy, the characteristics of the Sony BMG system were illegitimate regardless of the requirements of substantive legal doctrine, and should not have been designed as they were, particularly given that if the issues with the code's design – its computational legalism – had not been stumbled upon, they may well have continued to operate for some considerable period without being detected and remedied.

1.3 Aspiring to Legitimacy in Code

If the rote heteronomy of legalism is at one end of a spectrum, at the other is the aspirational concept of *legality*, which seeks to maintain a connection between the normative construct of law as a system of governance and the legitimising principles that underlie the exercise of sovereign power in constitutional democracies. Although an unsettled concept, legality has a theoretical pedigree that includes influential analyses that fit well with the normative approach I am adopting. As an aspiration, it is considered to be of fundamental importance in constitutional democracies; Bańkowski goes so far as to say it is 'something worth living for; something worth dying for'.[74] Hildebrandt defines legality by what for her it is not: legal certainty, 'justice', and expediency on their own are insufficient; the characteristic of legality also encompasses the rule of law and the binding of the sovereign's legislative power within constitutional limits.[75] For Brownsword, legality is about human dignity and the creation and maintenance of conditions that 'make moral community possible'. Legality, then, is not just about the substance of

[70] See Russinovich (n 68); Halderman and Felten (n 64) 88 *et seq*.

[71] Mulligan and Perzanowski (n 67) 1158.

[72] Ibid. 1169.

[73] I discuss this in more detail in L Diver, 'Law as a user: Design, affordance, and the technological mediation of norms' (2018) 15 *SCRIPTed* 4. See also Halderman and Felten (n 64) 91, stating that 'the [DRM] systems make no pretense of enforcing copyright law as written, but instead seek to enforce rules dictated by the label's and vendor's business models'.

[74] Bańkowski (n 54) 45.

[75] Hildebrandt, *Smart Technologies* (n 14) 157–8.

legal regulations, but also their form.[76] This idea of purpose binding speaks to the ex ante, 'constitutional' nature of the present analysis. Through the guidance of designers' production of technological normativity, we can help ensure that the negative outcomes towards which computational legalism tends are minimised as far as possible.

(a) From Operation to Production

I have already mentioned the importance of widening our focus to include the production of code, in addition to the orthodox ex post assessment of its operation. This relates to the notions of legality just mentioned – one can appreciate the relevance of the 'design' of a rule to the question of whether it meets those ideals. Whereas legalism looks only to sources to discern validity, legality is something altogether more reflexive and rational,[77] seeking evidence of certain requirements in the rule-making process. There is a clear alignment here between this view of legality and the shift in the literature towards design thinking that I mentioned above.

Fuller's *Internal Morality of Law*

This idea of rule production connects with Fuller's influential theory of the *internal morality of law*, whose eight principles of legality provide an underlying quasi-formal substrate necessary for making good legal rules, regardless of any reasonable disagreement there might be about their substantive content (that is, their 'external morality').[78] What several of the principles point towards is how best to design a legal norm, regardless of what its external morality is or ought to be. Indeed, Fuller uses the language of design on various occasions, referring to law-making as a 'craft'[79] and to the eight principles as 'those laws respected by a carpenter who wants the house he builds to remain standing and serve the purpose of those living in it'.[80] We will see later how the internal and external moralities of law relate to input and output reasons for decision-making[81] and, perhaps surprisingly, even to Hart's theory of primary and secondary rules, separating substantive ordinances from the rules which set out how they can be validly created, modified, and extinguished.[82]

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

[77] Bańkowski and Schafer (n 32) 31–2.

[78] Fuller (n 33) chapter 2.

[79] Ibid. 43, 156.

[80] Ibid. 96.

[81] Goldoni (n 24) 127, citing J Waldron, 'The core of the case against judicial review' (2006) 115 *Yale Law Journal* 1346.

[82] HLA Hart, *The Concept of Law* (2nd edn, Clarendon Press 1994) chapter V.

Wintgens's *Legisprudence*

The second primary theoretical source I draw upon for the formal qualities that normative orders ought to reflect is Wintgens's *legisprudence*. Although less well-known than Fuller, it can play an important role here as an aspirational framework that challenges legislators to achieve formal legitimacy in the process of developing new legal norms.[83] Wintgens argues that legal theory has in general been preoccupied more with adjudication than with legislative rule-making;[84] legisprudence, by contrast, is specifically aimed at the process of legislating, placing emphasis on the formal characteristics that a legal norm ought to have to be deemed a legitimate incursion on individual freedom. Upholding individuals' subjective notions of freedom ought to be a guiding principle of both politics and law, and any limitation on that freedom by law is legitimate only if it is justified according to the four legisprudential principles.[85] Fidelity to rules remains a necessary part of legal order, but this is via a 'weak' legalism which, unlike the stronger form introduced above (and described in detail in Chapter 3), requires those rules to be formulated in accordance with ex ante standards and not simply on the whim of the sovereign. Expecting citizens to follow rules thus becomes acceptable because those rules, legitimated by application of the legisprudential principles, cannot be arbitrary exercises of power. Briefly, the principles concern whether or not a binary rule is desirable, whether the proposed norm is proportionate to the issue the legislator seeks to address, whether its design enables ongoing assessment of its efficacy, and finally whether it is coherent at the semantic, temporal, intra-systemic, and extra-systemic levels.[86]

As suggested above, computational legalism represents the strongest of legalisms. The impetus to legitimate the exercise of power by designers whose code vies with law to regulate behaviour is therefore all the greater. Designers limit individual and collective freedom in ways that have not been sanctioned by the democratic polity, via mechanisms that are technically and socially opaque and which are not straightforwardly susceptible to public contest, redress, and (judicial) review. They are therefore potentially illegitimate exercises of power whose effects are difficult to arrest or ameliorate, particularly when diffused across millions of devices, often with little or no technical means of applying retrospective fixes.

[83] Wintgens, 'Legisprudence as a new theory of legislation' (n 61). For an explanation and history of the term 'legisprudence', see Wintgens, *Legisprudence: Practical Reason in Legislation* (n 53) 231–5.

[84] Wintgens, 'Legisprudence as a new theory of legislation' (n 61) 1.

[85] Wintgens, *Legisprudence: Practical Reason in Legislation* (n 53) 220.

[86] Chapter 4 discusses the principles in greater detail.

(b) Towards Digisprudence*: Legitimate 'Code as Law'*

As Koops suggests, 'a good place to start looking for criteria for acceptability of normative technology is to study criteria for law'.[87] The Fullerian and legisprudential principles are an excellent starting point, concerned as they are with providing criteria for acceptable law-making. Undoubtedly, they do not map directly onto the digital context, and so in Chapter 6 I translate them into the language of affordance, discussed above, in order ultimately to set out a framework for ensuring legitimate rule-making in the commercial design environment.

As Chapter 6 sets out, the proposed framework consists of a set of digisprudential affordances that translate the principled goals that I distil from the literature into concrete suggestions for the design of code. In brief, these cover contestability (with individual and institutional dimensions), transparency (covering provenance, purpose, and operation), choice, delay, and oversight.[88] The affordances are simultaneously general and concrete: they provide a design goal that should be reflected in all legitimate citizen-facing code, regardless of the form of technology, its substantive functionality, or the underlying business model. (A corollary of this is that certain functionalities or business models will therefore be illegitimate by definition.)

There may be edge cases where the affordances are less easy to envisage or implement. However, like the legal-theoretical foundations upon which it builds, digisprudence is aspirational: both legisprudence and the Fullerian principles are intended to encourage better (if not perfect) rule-making, and so similarly it is not expected that the digisprudential framework will cover every conceivable scenario where normative code is being produced. As Fuller suggests, perfect legality is 'utopian';[89] Wintgens notes in a similar vein that respect for the legisprudential principles is about 'the aspiration to do the job as well as possible'.[90] The same can be said of its technological counterpart that I am here proposing.

1.4 'Code as Law', Code *versus* Law, or Something Else?

This book is published in a series called 'Future Law' and has been adapted from a doctoral thesis written in a law school. One might reasonably therefore

[87] 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) 162. I discuss Koops's analysis, along with the other literature on criteria for code, in Chapter 5.

[88] See Section 6.3.

[89] Fuller (n 33) 41, 43.

[90] Wintgens, *Legisprudence: Practical Reason in Legislation* (n 53) 280.

expect it to make an argument about, say, the need for better laws, that is legal norms that reflect technological developments or that can more effectively bridge the regulatory gap. Hopefully, it is clear that that is not my focus. Instead, I am interested in the body of normativity that operates separately from and in parallel with institutional law.[91] My argument is first that it exists, and second that it ought to be subject to scrutiny by those whose theoretical expertise in the paradigmatic normative order – the law – can bring something new to bear, particularly when combined with practical knowledge of how the rules of code are made.

(a) Cyberlibertarianism

In acknowledging the existence of this parallel ordering, my aim is not to follow the 'cyberlibertarian' position that welcomes and seeks to validate the usurping of the state by private producers of code.[92] It is in fact precisely the opposite. We are at risk of finding ourselves in a 'Collingridge dilemma', such that by the time consensus has been reached (if it ever is) on the need to directly regulate specific technologies and the questionable business models from which they spring, conditions have become such that implementing any change is expensive, difficult, and time-consuming.[93]

My goal therefore is first to acknowledge the reality of this predicament and then to adopt a precautionary approach, suggesting ways we might guide the practice of design towards outcomes that are more legitimate, as defined according to the existing legal-theoretical frames that I will later adopt. Therefore, while I agree with some of the cyberlibertarians' descriptive characterisations of code, I expressly disagree with their normative positions on what should flow from those characteristics.

One of the traditional counterarguments to the cyberlibertarian position is that code is readily susceptible to regulation by law. Arguments about the regulability of code are valid as far as they go, but they do not adequately encompass the technical characteristics of recent technologies (for example blockchain applications) nor address the question of how code is produced.

[91] R Brownsword, 'In the year 2061: From law to technological management' (2015) 7 *Law, Innovation and Technology* 1, 10–14; R Mohr and F Contini, 'Reassembling the legal: "The wonders of modern science" in court-related proceedings' (2011) 20 *Griffith Law Review* 994, 998.

[92] The classic expression of this perspective is JP Barlow, 'A declaration of the independence of cyberspace' (1996) <https://www.eff.org/cyberspace-independence> last accessed 4 March 2021. For another argument in this vein, see DR Johnson and DG Post, 'Law and borders – the rise of law in cyberspace' (1995) 48 *Stanford Law Review* 1367.

[93] MT Young, 'Artifacts as rules: Wittgenstein and the sociology of technology' (2018) 22 *Techné: Research in Philosophy and Technology* 377.

Furthermore, those scholars who have argued against the idea that code is hegemonic have tended to focus on the infrastructure of the Internet and the large platforms that own and operate it,[94] rather than on the individual digital artefacts that constitute our daily lives online. This code is often produced by smaller enterprises[95] who are less easy targets for traditional regulation and who may view the benefits of compliance as being outweighed by its cost,[96] particularly when they lack dedicated legal departments or expertise. (Indeed, a great deal of code is produced by individuals or microbusinesses.[97])

Such scholarship has not engaged in depth with the role and practices of the designer as the creator of the code-based norms that constrain and enable behaviour. Again, though, another counterargument to the thesis I am advancing might be that designers, just like any other legal person, should be the subjects of traditional regulative processes and, therefore, any illegality in the code they produce should be dealt with using traditional ex post legal processes. In the computational context this is necessary, but insufficient: as we shall see below and in subsequent chapters, the ex ante legitimation of code in addition to ex post legal remedial measures is crucial because of its *sui generis* nature as a regulator. The threat of computational legalism means that the stakes are both qualitatively and quantitatively higher than with other instances of problematic regulation that can be ameliorated by traditional legal processes. The statute that is improperly enacted or the contract that is voidably concluded are defeasible, that is they are presumed valid but are nevertheless always open to challenge in, and reduction by, a court with the relevant authority.[98] The characteristics of code as a regulator admit of no such possibility: once its rules are 'promulgated', any 'illegality' has no bearing on its ability to execute and impose any latent normativity that it harbours. From

[94] Goldsmith and Wu, for example, focus on the physical networks that underpin the Internet, noting that they are owned by 'some of the most regulated companies on earth'. See JL Goldsmith and T Wu, *Who Controls the Internet?: Illusions of a Borderless World* (Oxford University Press 2006) 73. As discussed above, we also saw this focus in the 'code as law' literature.

[95] G Papadopoulos et al., 'Statistics on small and medium-sized enterprises' (European Commission 2018) <https://ec.europa.eu/eurostat/statistics-explained/index.php/Statistics_on_small_and_medium-sized_enterprises> last accessed 4 March 2021.

[96] See for example London Economics, *Study on the Economic Benefits of Privacy-Enhancing Technologies (PETs)* (London Economics 2010).

[97] See Stack Overflow, 'Developer survey 2020' *Stack Overflow* <https://insights.stackoverflow.com/survey/2020/> last accessed 4 March 2021 (purportedly the largest survey of developers in the world, demonstrating that a quarter work in companies with fewer than twenty employees).

[98] N MacCormick, *Rhetoric and the Rule of Law: A Theory of Legal Reasoning* (Oxford University Press 2005) chapter 12.

the moment of 'shipping', the code will operate as though it was legitimately 'enacted', even where this is manifestly not the case. There is, therefore, a crucial difference between invalid laws and 'invalid' code: with the former, the hermeneutic gap that exists between text and action allows for a space in which validity can be considered, whereas with the latter there is no such opportunity, either to arrest execution or (in many cases) even to observe the invalidity. This in turn connects with the question of ex post contest. If the nature and extent of the code's invalidity cannot be observed, traditional mechanisms of legal redress cannot meaningfully be invoked. Ultimately, we fall into a trap if we assume that institutional law is capable of operating with its usual force where code is the subject of regulation, at least without some form of acquiescence from the other side.

Any attempt to grapple with this difficult reality will require a shift in discourse 'from distribution to production and [thus a] focus on how the digital environment is created'.[99] Thankfully, an emerging turn in the legal literature – so far mostly in the sphere of privacy – demonstrates a shift in focus towards design and the production of code.[100] As Gürses and van Hoboken note, 'the ideological markers, pools of desirable knowledge and practices of technology *production* that bring these sets of [ex post] conditions forth and not others tend to go unquestioned'.[101] The effects of computational legalism make code resistant to the modulating effects of interpretation and ex post remedial measures that are more readily effective in the realm of traditional text-based law. It is clear, therefore, that in addition to those traditional ex post methods of redress, we should aim for ex ante code legitimacy. Digisprudence contributes to that emerging debate with a focus on legitimacy,[102] and the expectations we should have of designers and enterprise in their anticipation of the effects of the code that they produce.

(b) Why Not 'Compliance by Design'?

The goal of 'compliance by design' ('CbD') concerns meeting the requirements of a specific field of substantive doctrinal law within the design of the

[99] Goldoni (n 24) 129.

[100] See for example W Hartzog, *Privacy's Blueprint: The Battle to Control the Design of New Technologies* (Harvard University Press 2018); Gürses and van Hoboken (n 5). See also P Nemitz, 'Constitutional democracy and technology in the age of artificial intelligence' (2018) 376 *Philosophical Transactions of the Royal Society A* 20180089, arguing for a design perspective on the effects that artificial intelligence is having on constitutional democracy.

[101] Gürses and van Hoboken (n 5) 580 (emphasis supplied).

[102] Cf. A Murray, 'Looking back at the law of the horse: Why cyberlaw and the rule of law are important' (2013) 10 *SCRIPTed* 310. Murray retains the orthodox position of viewing code as the subject of law.

code, insofar as that field targets its norms at digital artefacts. This interpretation of CbD is in line with the terminological usage of initiatives like 'privacy by design' and the GDPR's 'data protection by design'.[103] These are examples of orthodox technology regulation, where the focus is the regulation of software by substantive doctrinal law.

This is centrally important of course insofar as compliance with the law is important in all contexts. From the perspective of digisprudence, however, it is a limited and inherently legalistic view, one that looks upon the law as a set of rules that is 'just there', to be passively observed and obeyed by the designer of the code. It also narrows our focus away from the broader range of 'techno-effects' that play as important (and indeed larger) a part in regulating behaviour as compared with legally sanctioned code.[104] A perspective of code based solely on this understanding of 'compliance by design' is unsatisfactory, or at the very least incomplete, because it elides the very active role that designers play in the creation of such normative 'reality' in and through the code that they produce.

As a general aim, CbD overlooks (1) the *sui generis* nature of code as a regulator of behaviour (that is, it overlooks computational legalism), and (2) how the translation from textual norms to code-based norms invariably involves some level of modification of the former.[105] The precise nature of the reality envisioned by legal text is not reflected in the reality constructed by code, partly because law itself is (and arguably should be) vague,[106] and partly because the two modes of representing meaning (text and software code) are by nature very different, both because language is vague where code is precise and because words require translation into behaviour whereas code is simultaneously documentary and performative. The point is not just to improve the methods of transferring norms between domains, but also to ensure there are mechanisms in place – safety valves – that allow for mis-translations properly to be dealt with according to the rule of law.

The lack of one-to-one mapping of meaning not only is true of attempts to interpret and instantiate substantive (textual) legal norms in code, but is

[103] Regulation on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 85/46/EC (General Data Protection Regulation) 2016, Recital 78 and art. 25.

[104] 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).

[105] Goldoni (n 24) 129; Hildebrandt and Koops (n 30) 452 *et seq*.

[106] T Endicott, 'Law is necessarily vague' (2001) 7 *Legal Theory* 379; C Reed, 'How to make bad law: Lessons from cyberspace' (2010) 73 *The Modern Law Review* 903, 904 *et seq*.

also demonstrated in the unintended constellations of legal and non-legal effect that are continually being reified by digital artefacts.[107] This is what van den Berg and Leenes refer to as 'techno-effects',[108] or the aggregate normativity of a technology considered regardless of the designer's intent or any legal impetus behind its design. While much of the literature focuses on 'techno-regulation', or the use of technology as a tool to effect legal norms, there has been insufficient consideration of the wider spectrum of techno-effects, including a-legal regulation. This gap is an important one, particularly given that '[t]he "regulatory" potential of technologies – in the broadest sense – is tremendous, and daunting, indeed.'[109] Not only can it be difficult to discern the intention of the designer, but so too is the line between intentional and unintentional normativity difficult to detect – '[t]he affected individual cannot discern which part of the normativity (as could be inferred from the output) is intentional and which part is merely spin-off in the form [of] unforeseen or secondary effects.'[110]

Whereas law benefits from delay and processes of interpretation that permit application across heterogeneous circumstances,[111] code tends by nature towards fixed (or inflexible) configurations of normativity, rather than interpretable standards. These are imposed with unqualified force in every case where the necessary computational conditions arise, regardless of any other relevant considerations. The challenge therefore is to ensure that the fixity of code is as legitimate as it can be *ab initio*. As Goldoni puts it, 'on the one side, code can be a norm-enforcing technology, as has been outlined several times in the debate; on the other side, code can also be a norm-establishing technology as well'.[112] If both law and code create norms, and we as a society have expectations about the legitimacy of the former, then we ought to expect similar standards from the latter. Code, however, is not law, it is only law-like; but it is precisely because of the ways in which it is not law that this kind of analysis is necessary: code can control behaviour more directly than can 'true' law, but simultaneously it lacks the latter's mechanisms of ex ante legitimation, defeasibility, and ex post remediation.

[107] Van den Berg and Leenes (n 104).
[108] Ibid. 81.
[109] Ibid. 83.
[110] Bayamlıoğlu and Leenes (n 7) 12. They refer to this phenomenon as 'normative opaqueness'.
[111] Endicott (n 106) 382–3; Gadamer (n 29) 334 *et seq*.; R Dworkin, *Law's Empire* (Belknap Press 1986) *passim*.
[112] Goldoni (n 24) 118.

The need is therefore all the greater for it to be legitimated from the outset, within the design process, and not only in the aftermath of a high-profile data breach or other scandal. As Goldoni points out,

> [g]iven that code is not exactly like law, it is difficult in the realm of code to adopt a kind of rule of law (or 'rule of code') approach. Yet, we have also seen that when a particular code is 'enacted', it may be too late to remedy the violation of certain rights. This is why the accent should be put on the moment of production, rather than on the moment of distribution.[113]

The opacity of code means that only the most conspicuous illegitimacies are ever likely to be exposed; this highlights the problem of a retrospective, ex post focus centred on the operation of code rather than on its production.

Consider, for example, the controversy surrounding Facebook and the sharing of its users' personal data with third-party application developers, who subsequently used it to micro-target election advertisements online. The case is a complex (and evolving) mix of business ethics, democratic politics, and doctrinal law, but at its heart lie decisions made by designers that are concretised in code: a now-deprecated version of Facebook's application programming interface (API)[114] allowed developers to access the data of 'friends' of the primary end-user, which enabled the large-scale data harvesting that facilitated the voter profiling at the centre of the controversy.[115] This is a high-profile, high-public interest case, and has thus been subject to relatively intensive scrutiny from experts and regulators. Despite this, the challenges of such an ex post remedial approach are precipitous, given the complexity of both the systems involved and of Facebook as an organisation.[116]

[113] Ibid. 128.

[114] APIs allow unconnected software systems to communicate with one another, enabling the combination of systems with different specialities, for example mapping, payment processing, and biometric authentication.

[115] For a technical overview, see Information Commissioner's Office, 'Investigation into the use of data analytics in political campaigns – investigation update' (Information Commissioner's Office 2018) s. 4.3.1. On the broader political implications, see P Geoghegan, *Democracy for Sale: Dark Money and Dirty Politics* (Head of Zeus 2020) chapter 8.

[116] As Pasquale notes, '[i]t could take weeks to fully map the flow of data from something as simple as commenting on Facebook.' See F Pasquale, *The Black Box Society: The Secret Algorithms that Control Money and Information* (Harvard University Press 2015) 144. Indeed, it took the UK Information Commissioner several months to investigate the nature of Facebook's systems. For a set of fascinating visualisations demonstrating the complexity involved, see Share Lab, 'Immaterial labour and data harvesting' *Share Lab* (21 August 2016) <https://labs.rs/en/facebook-algorithmic-factory-immaterial-labour-and-data-harvesting/> last accessed 4 March 2021.

Facebook is clearly a prominent target for regulators, and its potential role in election tampering means the case is of the greatest public interest. The question remains, however, of the extent to which less significant code infelicities might be operating all around us, never detected or remedied, because the scrutiny and impetus to investigate ex post are relatively minimal or simply absent.

Other Notions of 'by Design'

Broader, more nuanced notions of 'by design' that accord with the perspective I am adopting do exist. For example, Nemitz refers to 'the principles of democracy, rule of law and human rights by design'.[117] Similarly, Hildebrandt defines her concept of 'Legal Protection by Design' as 'a way to ensure that the technological normativity that regulates our lives: first, is compatible with enacted law, or even initiated by the democratic legislator; second, can be resisted; and third, may be contested in a court of law'.[118] One can see how Nemitz's and Hildebrandt's concepts include more fundamental issues than compliance with substantive doctrine. Hildebrandt's and Koops's earlier formulation of 'ambient law' is also close to the idea at hand, where what matters is not (just) compliance with the substantive law, but the kinds of constitutional safeguards that law as a normative enterprise is expected to provide, regardless of the substantive content of its rules.[119]

While the authors are concerned with the reflection of state-sourced law in code, it can equally be said that code which embodies normativity that is not state-sourced ought also to embody 'safeguards' in order for it to be legitimate. In this context, where commercial enterprise is the source of the normativity, the requirement of democratic participation in the design of code is unlikely to be achievable by smaller enterprises with limited resources to invest in the necessary processes.[120] Initiatives connected with this goal include participatory design,[121] constructive technology assessment,[122] value

[117] Nemitz (n 100) *passim*.

[118] Hildebrandt, *Smart Technologies* (n 14) 218.

[119] Hildebrandt and Koops (n 30) 445. See also 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).

[120] Papadopoulos et al. (n 95).

[121] S Costanza-Chock, *Design Justice: Community-Led Practices to Build the Worlds We Need* (MIT Press 2020).

[122] P-P Verbeek, 'Materializing morality: Design ethics and technological mediation' (2006) 31 *Science, Technology, & Human Values* 361, 375 *et seq*.

sensitive design,[123] and ideation.[124] These are valuable initiatives, but their focus tends to individualise the idea of 'constitutional' standards that I am concerned with, through the focus on the practices of a particular designer/team/enterprise and how these impact on a particular design project. Such initiatives seek to legitimise a design by dint of having involved those with a stake in the outcome in decisions as to the artefact's substantive characteristics. By contrast, the 'constitutional' view of digisprudence comes before such questions, advocating for universal formal standards to be present regardless of the application or the participation of affected groups. Design for all need not require design with all;[125] the characteristics of legitimacy I propose are primarily formal and ought to be present in all citizen-facing technologies, regardless of their substantive purpose. As with legal rules, while we can disagree about the desirability of their substantive content we would, I think, agree generally that the process of their creation ought to meet certain standards, and that they ought to reflect formal qualities such as intelligibility and non-retroactivity. In that vein, then, we can say that digisprudence is to participatory design approaches as legisprudence is to the democratic process; they are separate but complementary aspects of the norm-creation process.

(c) Normative Relationships in Code and Law

To further clarify the location of the enquiry, we can visualise the normative relationships in the digital sphere as shown in Figure 1.1. Relationship (b) represents the classic compact between the citizen end-user and the state – the latter being bound by a constitution in relationship (a) – through which democratic participation results in the state's promulgation of legal norms through both that relationship and relationship (c). The latter represents the traditional understanding of 'compliance by design', discussed above.

Digisprudence focuses on relationships (d) and (e). In relationship (d), the product designer imposes behavioural constraint through a mix of legal and architectural normativity. The legal normativity in this relationship can flow from public-order norms (legislation of various forms) on the one hand, or private-order contractual norms on the other. These are operationalised by

[123] B Friedman, 'Value-sensitive design' (1996) 3 *interactions* 16.

[124] E Luger and M Golembewski, 'Towards fostering compliance by design; drawing designers into the regulatory frame' in M Taddeo and L Floridi (eds), *The Responsibilities of Online Service Providers* (Springer 2017); B Friedman and D Hendry, 'The envisioning cards: A toolkit for catalyzing humanistic and technical imaginations' in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (ACM 2012).

[125] 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) 351.

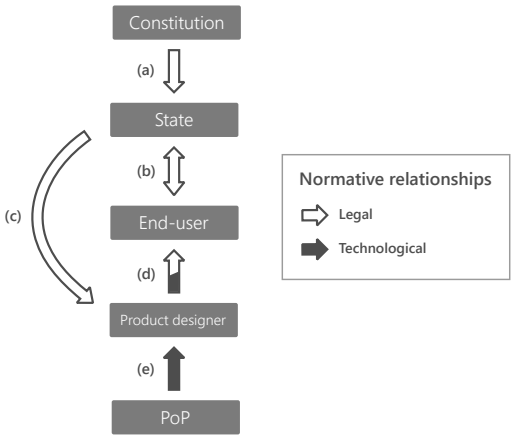

Figure 1.1 Normative relationships in law and technology

(1) the traditional force of law, (2) the norms' implementation in and through the code, or (3) a mixture of the two. In the second and third scenarios, code complements law.[126] Examples include encryption used to implement data protection requirements flowing from a public-order norm in relationship (c), or a firewall preventing an employee's computer from accessing social media, thus implementing a private contractual norm.

Whether or not these code rules aim explicitly to instrumentalise legal norms, they by definition exist separately from the law's corpus of rules.[127] Viewed traditionally, the legal effect of the data protection statute or the employment contract applies regardless of either instrument's implementation in or through code. But a corollary arises from this: it is precisely in the separateness of the two mechanisms of regulation that the architectural force of code, which implements some form of normativity, is able to 'supplant the legal infrastructure of the state'.[128] Whereas the data protection statute or employment contract awaits ex post enforcement following detection of some kind of breach or failure to act, code simply goes ahead and imposes some

[126] Leenes calls this 'state endorsed techno-regulation'. See R Leenes, 'Framing techno-regulation: An exploration of state and non-state regulation by technology' (2011) 5 *Legisprudence* 143, 160.

[127] Both Schmidt and Reed discuss the concept of 'law-system quality' in relation to public norms fitting technology (this is top-down, relationship (c) normativity). See A Schmidt, 'Radbruch in cyberspace: About law-system quality and ICT innovation' (2009) 3 *Masaryk University Journal of Law and Technology* 195; Reed (n 106). By contrast, I am concerned with bottom-up instantiation of normativity in relationships (d) and (e).

[128] Radin (n 62) 143.

alternative configuration of behavioural constraint which might not comport with the substantive law (the specific statute, contract, or the whole corpus of legal rules) nor reflect the standards of legitimacy according to which all behaviour-constraining norms should be made.

The Programmer of the Programmer

The notion of underlying, ex ante standards is at the core of digisprudence, and is reflected pragmatically in relationship (e), between the product designer and what Vismann and Krajewski call the *programmer of the programmer* ('PoP').[129] The PoP designs the tools that the product designer in turn uses to create the products and services ultimately destined for the end-user. Situated at a 'constitutional' level of the product design process, the decisions made by the PoP fundamentally frame what the product designer can and cannot do. The PoP thus has a crucial power to define the rules of the design game before it even begins. This idea of 'technological constitutionalism', which I link with Hart's concept of secondary rules,[130] suggests one locus for the operationalisation of formal principles that can constrain the substantive design of code to encourage legitimacy. I will discuss this concept in greater detail in Chapter 2, and then again in Chapter 7 in relation to the operationalisation of digisprudence.[131]

1.5 In the Real World

Above I discussed DRM in terms of computational legalism. Later in Chapters 6 and 7, when setting out the digisprudential framework of affordances and their operationalisation, I ground the theory in real-world code through a discussion of its application to two important contemporary classes of technology, namely blockchain applications and the Internet of Things (IoT). At this point it makes sense to lay some brief groundwork as to how the theory will apply to them.

(a) Blockchain Applications

The first case study focuses on so-called 'smart contracts' built upon the foundation of blockchain technology (later I shift from the term 'smart contract' to 'blockchain application', for reasons I will explain below). Like DRM, smart contracts represent another very explicit example of the embodiment of rules that have normative significance within the fabric of a digital artefact.

[129] Vismann and Krajewski (n 28). For an earlier discussion alluding to a similar concept see Weizenbaum (n 1) 100 *et seq*.

[130] Hart (n 82) 91 *et seq*.

[131] See Section 7.1.

Although blockchain technology is still maturing – the Bitcoin paper that proposed its initial design was published anonymously in 2008,[132] and the first blockchain went live in January the next year[133] – the implications and the publicity surrounding it[134] are the subject of increasing scrutiny from the legal academy. While hype on its own does not justify academic attention, the peculiarly normative characteristics of smart contracts and their design raise questions of explicit interest in this context.[135] While there is increasing scepticism about the practical value of blockchains,[136] those characteristics mean they will remain problematic even if they turn out not to be the revolutionary technology some suggest they are.

Blockchain Design

To fully appreciate the relevance of blockchain applications from a digisprudential perspective, some knowledge of their architectural characteristics is necessary.[137] Blockchains are public[138] databases (or 'ledgers' – hence the alternative term 'distributed ledger technology', or 'DLT') which are stored on a number of computers ('miners') which together constitute a peer-to-peer network. To add to the chain requires consensus among the network's

[132] S Nakamoto, 'Bitcoin: A peer-to-peer electronic cash system' (2008) <https://bitcoin.org/bitcoin.pdf> last accessed 15 April 2021.

[133] P De Filippi and A Wright, *Blockchain and the Law: The Rule of Code* (Harvard University Press 2018) 205.

[134] By 2016, blockchain had almost reached the 'peak of inflated expectations' in Gartner's Hype Cycle for Emerging Technologies. See Gartner, 'Gartner's 2016 Hype Cycle for Emerging Technologies identifies three key trends that organizations must track to gain competitive advantage' *Gartner* (16 August 2016) <https://www.gartner.com/en/newsroom/press-releases/2016-08-16-gartners-2016-hype-cycle-for-emerging-technologies-identifies-three-key-trends-that-organizations-must-track-to-gain-competitive-advantage> last accessed 4 March 2021.

[135] RH Weber, '"Rose is a rose is a rose is a rose" – what about code and law?' (2018) 34 *Computer Law & Security Review* 701, 705.

[136] I Kaminska, 'Growing scepticism challenges the blockchain hype' *Financial Times* (20 June 2017) <https://www.ft.com/content/b5b1a5f2-5030-11e7-bfb8-997009366969> last accessed 4 March 2021.

[137] For a more in-depth primer on blockchains, see M Pilkington, 'Blockchain technology: Principles and applications' in FX Olleros and M Zhegu (eds), *Research Handbook on Digital Transformations* (Edward Elgar Publishing 2016).

[138] Private ('permissioned') blockchains also exist, but because these are generally used internally within an organisation they mostly lack the focus on end-user behavioural regulation represented in relationship (d) in Figure 1.1 above, and so I do not include them in this analysis. For more on private blockchains, see V Buterin, 'On public and private blockchains' *Ethereum Foundation Blog* (7 August 2015) <https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/> last accessed 4 March 2021.

nodes, and so a new 'block' of data will only be added if a majority of miners agree that its addition meets the requirements governing that particular blockchain.[139] These rules are known as the blockchain's 'protocol', and they define how the blockchain operates and what the incentives and costs are for participants, including the miners who provide the network's infrastructure and the end-users who transact with/through it. Two prominent examples of different blockchain protocols are Bitcoin,[140] the cryptocurrency and original application of a blockchain protocol, and Ethereum,[141] the first blockchain to support sophisticated automation through the provision of a decentralised computing platform (the Ethereum Virtual Machine).

The protocol will include some mechanism for the miners to reach consensus on what should be stored, including both metadata about transactions and new smart contracts to be executed. The question of how to reach consensus among anonymous computers is connected with what is known as the 'Byzantine fault problem', where the networked nodes each have a different understanding of the state of the chain but consensus must be reached for the system to be workable. Blockchain protocols overcome this using a combination of public key cryptography and *hashing*.[142] The former is a mechanism for uniquely and conclusively identifying each node within the network by a public signature (key), while the latter is a method for generating a unique signature (a hash) from any given volume of data (in this case, the existing prior state of the blockchain). Each block is assigned a unique hash, generated from a combination of that block's data and the hashes of all the blocks that are already on the chain. This means that if the last block in two copies of the chain have the same hash, one can be completely confident that the copies of the chain are identical all the way back to the first block, and therefore that neither copy has been tampered with.

When a miner solves the mathematical challenge specified in the chain's protocol (this is how new blocks are added, for which the miner receives a reward), the proposed solution is broadcast to the network for the other miners to verify. They independently generate a new hash from the existing state of the chain and the proposed solution broadcasted by the 'winning' miner, and if the solution meets the requirements of the protocol, each miner adds the block to their local copy of the chain. In this way the copies of the chain are kept identical and up to date across the many miners that store them.

[139] De Filippi and Wright (n 133) 2.

[140] Nakamoto (n 132).

[141] Ethereum Foundation, 'Ethereum white paper' (Ethereum Foundation, 22 August 2018) <https://ethereum.org/whitepaper> last accessed 4 March 2021.

[142] Pilkington (n 137) 228.

An important corollary of this proposal mechanism, particularly its use of hashes that represent the historical state of the chain, is that once a block has been added its contents are both immutable[143] and verifiable by observers.[144]

Copies of the blockchain, including both its protocol and the data that it stores (for example transaction metadata, account balances, smart contract code) are replicated across the network, providing resilience through decentralisation.[145] Disabling (even physically) one of the network's computers will not delete the blockchain or prevent the code it stores from executing.[146] The lack of centralised (state) authority controlling what gets added to the chain is part of the ideology driving the technology:[147] provided participants follow the rules contained in the protocol, they get the benefits of a tamper-resistant, 'trustless' database with no overseeing entity.

'Smart Contracts'?

At present blockchains are probably best-known as the foundation of cryptocurrencies, but another related application that is potentially more disruptive from a legal perspective are so-called 'smart contracts' ('SCs'). SC platforms provide varying levels of sophistication. The Bitcoin protocol provides some very basic programming capabilities which can allow very limited SCs to be written. Other platforms provide a more sophisticated programming foundation for SCs, of which the most prominent is Ethereum.[148]

Ethereum's creators describe it as a 'next-generation smart contract and decentralized application platform'.[149] It complements the architectural characteristics of blockchains with a fully-fledged programming execution environment, meaning computationally rich functionality can be combined with the immutability, decentralisation, and 'trustless trust' of blockchains. The

[143] Ibid. 233–4.

[144] Ibid. 227.

[145] De Filippi and Wright (n 133) 2.

[146] This emulates the ethos of ARPANET, precursor to the modern Internet, designed during the Cold War to be resistant to physical attacks on infrastructure. See BM Leiner et al., 'Brief history of the Internet' (Internet Society 1997) 3 <https://www.internetsociety.org/internet/history-internet/brief-history-internet/> last accessed 4 March 2021.

[147] On which, see D Golumbia, *The Politics of Bitcoin: Software as Right-Wing Extremism* (University of Minnesota Press 2016).

[148] For an empirical overview of the current major SC platforms, see M Bartoletti and L Pompianu, 'An empirical analysis of smart contracts: Platforms, applications, and design patterns' (2017) arXiv preprint arXiv:1703.06322 <https://arxiv.org/abs/1703.06322> last accessed 4 March 2021.

[149] Ethereum Foundation (n 141).

innovative possibilities of software's plasticity can thus be undergirded with the stability inherent in the 'anti-plasticity' of blockchains.

'Smart contracts' combine 'Turing-completeness, value-awareness, blockchain-awareness and state',[150] meaning they can define complex conditions, execute arbitrary behaviours when certain conditions are met, maintain and monitor states over time, and record the outcomes in the immutable blockchain. All of this is potentially automated; conditions defined in the 'contract' are 'live', awaiting whatever change(s) are necessary to trigger the rules it contains. In this sense smart contracts are not passive instructions on what the contracting parties should do, rather they are 'more like "autonomous agents" that live inside of the Ethereum execution environment, always executing a specific piece of code when "poked" by a message or transaction'.[151]

Multiple SCs can be bundled together by a central business logic (itself written in code and stored on the blockchain) to create a 'distributed organisation' ('DO')[152] and even a 'distributed autonomous organisation' ('DAO'), which can operate without any human input.[153] These artefacts' logic enables, disables, and manages individual SCs, using them as tools to effect external changes according to the rules predefined in the code. A DO could, for example, require a majority vote from its (human) members as a condition of a given smart contract being triggered. Again, the decentralised and 'trustless' nature of blockchain design obviates the need for a trusted centralised authority (a traditional board or committee), and so notional governance of the organisation can be achieved even where the membership is geographically dispersed, or even unknown.[154] SCs consult external sources of data, known as 'oracles',[155] to check for particular conditions in the world outside the code, executing predetermined logics when necessary conditions are met. By interacting with a cryptocurrency and the APIs of other services, real-world transactions can be effected, involving human actors (themselves

[150] Ibid.

[151] Ibid.

[152] V Buterin, 'DAOs, DACs, DAs and more: An incomplete terminology guide' *Ethereum Foundation Blog* (6 May 2014) <https://blog.ethereum.org/2014/05/06/daos-dacs-das-and-more-an-incomplete-terminology-guide/> last accessed 4 March 2021.

[153] Ibid.

[154] A Wright and P De Filippi, 'Decentralized blockchain technology and the rise of lex cryptographia' (Social Science Research Network 2015) SSRN Scholarly Paper ID 2580664, 15–16 <https://papers.ssrn.com/abstract=2580664> last accessed 4 March 2021.

[155] Cardozo Blockchain Project, '"Smart contracts" & legal enforceability' (Benjamin N Cardozo School of Law 2018) 6.

mediated by code infrastructures, as in the 'gig economy'[156]) and even other artefacts such as drones.[157]

The power of such code is intuitively appreciable. When specific conditions that are computationally representable are met, the code self-executes according to its internal logic, whatever that might be, and the outcomes are enforced regardless of any relevant real-world considerations. With the outcomes of the code's execution being stored in the underlying blockchain alongside the code itself, this means both its logic and its results are immutable once they are 'enacted', executed, and stored. Thus code, in a very real (and legally significant) sense, becomes 'law', through the 'collapsing [of] contract formation and enforcement into a single instrument'.[158] The coincidence of form and substance means that when executed, the material effects of the smart contract are governed by the dictates of pure code, regardless of any ambiguity or subjective understanding that might exist in the minds of the humans involved. One can appreciate the parallel with the discussion of computational legalism above, noting that code is at once rule and reality; the normative collapsed into the descriptive.

(b) The Internet of Things

Compared with blockchain applications, the Internet of Things (IoT) is perhaps a simpler (but no less important) area to conceptualise and analyse. In the early 1990s Mark Weiser, a pioneer of what has variously been termed 'ambient intelligence'[159] and 'ubiquitous computing', spoke of the profundity of technologies that 'weave themselves into the fabric of everyday life until they are indistinguishable from it'.[160] The US Federal Trade Commission has defined the IoT as 'devices or sensors – other than computers, smartphones, or tablets – that connect, communicate or transmit information with or between each other through the Internet'.[161] This focus on sensors

[156] Buterin describes a DAO as 'an entity that lives on the internet and exists autonomously, but also heavily relies on hiring individuals to perform certain tasks that the automaton itself cannot do'. See Buterin, 'DAOs, DACs, DAs and more' (n 152).

[157] De Filippi and Wright (n 133) 156. For an example of the latter, see J Perez, 'XYO game-changer: We've executed a smart contract with a drone!' *Medium* (21 November 2018) <https://medium.com/xyonetwork/xyo-game-changer-weve-executed-a-smart-contract-with-a-drone-4deb414af67b> last accessed 4 March 2021.

[158] KEC Levy, 'Book-smart, not street-smart: Blockchain-based smart contracts and the social workings of law' (2017) 3 *Engaging Science, Technology, and Society* 1, 3.

[159] Hildebrandt and Koops (n 30) 430–1.

[160] M Weiser, 'The computer for the 21st century' (1991) *Scientific American* 94.

[161] Federal Trade Commission, 'Internet of Things: Privacy and security in a connected world' (Federal Trade Commission 2015) 6.

and devices other than traditional platforms (computers and smartphones/tablets) implies the 'weaving into daily life' to which Weiser referred. Indeed, IoT devices are designed to do precisely this, both ubiquitously and invisibly, and as such are becoming an increasingly significant proportion of the total number of devices connected to the Internet. This is in part due to a 'chip-centric mentality', where manufacturers have bought into commercial hype suggesting that a connected device is better than an unconnected one.[162] The results of this are occasionally absurd.[163]

IoT devices both illustrate the design theories mentioned above and exemplify numerous aspects of computational legalism, especially opacity, immutability, and pervasiveness. Because they are intended to be embedded and pervasive, they by nature tend towards both minimal affordances and very strictly defined inscriptions. The Amazon Dash Button, for example, consists of just a single button and an LED indicator. Its inscription is thus a simple one of 'press the button', and its design affords that and little more ('adhesion' and 'throwing' are perhaps the only alternative action possibilities). As I describe in more detail in Chapter 6, behind this apparent simplicity and minimal interface lies a complex series of technical events that are kept hidden from the device's end-user but which are potentially of great importance to her (imagine the device being mis-used by a young child or pet).[164] As with other computing systems, the extent to which complex logic should be hidden from the user is one which will vary depending on the system in question. Nevertheless, the central issue of transparency about what lies beneath the physical device's 'tip of the iceberg' is a crucially important one. IoT devices, particularly those that have a single function like the Dash Button, generally combine simple physicality on the part of the object with complex and opaque computation on the 'back-end'. There is therefore significant scope for dissonance between the end-user's understanding of the device's affordances and what it in fact does.[165]

In terms of immutability, the poor infrastructural provision made for updates, coupled with a lack of commitment to long-term oversight, have resulted in many examples of IoT devices being used as nodes in bot-nets,

[162] W Hartzog and E Selinger, 'The Internet of Heirlooms and Disposable Things' (2016) 17 *North Carolina Journal of Law & Technology* 581.

[163] For an amusing selection of examples that demonstrate this, see the Twitter account 'Internet of Shit (@internetofshit)' *Twitter* <https://twitter.com/internetofshit> last accessed 4 March 2021.

[164] Federal Trade Commission (n 161) 22.

[165] A Matassa and R Simeoni, 'Eliciting affordances for smart objects in IoT era' in *Internet of Things: User-Centric IoT* (Springer 2015).

being left open to external hacking, and other forms of unintended breach.[166] This lack of flexibility, when coupled with the devices' intended pervasiveness, is potentially deeply problematic.

These problems can even combine with those of blockchain applications – as I mentioned above, the latter are capable of effecting changes in the physical world via IoT devices, such as drones and 'smart' devices. The combination implies the concept of 'smart property', where the hybrid IoT–blockchain artefact is autonomous, such that the IoT device's physical functionality is controlled by the logic contained in the blockchain application (for example a smart door lock might refuse to open after a code-based 'lease' expires[167]).

1.6 Conclusion

The goal of this introductory chapter has been to lay out the main contours of digisprudence, why it matters, and how it differs from existing literature, particularly that on 'code as law' and 'compliance by design'. It has also started to consider contemporary technologies through this novel analytical lens, an applied analysis that will be picked up again in Chapter 6. As mentioned above, the rest of the book follows the dialectical structure of the theory, exploring each element in greater depth: Part I problematises code from both design and legal theory standpoints; Part II considers the existing literature on standards that make both legal and technological normativity legitimate, and identifies the production gap that digisprudence aims to fill; finally, Part III synthesises the analysis, setting out the framework of digisprudential affordances before exploring some options for practical implementation and for future research.

I suggested above that there is real scope for reinvigoration of the debate that Lessig brought to prominence in the late 1990s, to take better account of the conceptual overlaps between legal theory, philosophy of technology, STS, and design studies. In the past two decades, academic lawyers have perhaps relied too heavily on Lessig's framework and the many assumptions that came along with it. My hope in the rest of this book is to reboot this conversation, renewing it through a combination of some of the rich theoretical insights that these fields have to offer with a pragmatic view of how code is actually made. I will only be able to scratch the surface of all of this of course, but in doing so I hope at the very least to raise some interesting new questions and to highlight some useful directions in which this fascinating and hugely important topic might be taken.

[166] Hartzog and Selinger (n 162).
[167] Levy (n 158) 3.