1 WHY AUTONOMOUS ORGANIZATIONS?

If we asked a hundred lawyers whether a robot or any other technological system can buy a house under today's law, they would probably all say no. Robots, after all, are not legal persons, and only legal persons can engage the legal system in ways that all humans take for granted: entering into contracts, owning property, suing, being sued, and so forth.

One of this book's central arguments is that this prevailing legal wisdom is incorrect, at least as a practical matter. It's true, formally, that a robot currently can't buy a house – the title to a house can't be in a robot's name, and a robot can't on its own execute a contract to buy the house, because the robot doesn't have those legal capabilities.[1] But my contention is that, practically speaking, a robot (or any other technological system, like software running in the cloud or on a distributed network) can in fact achieve all the capabilities of legal persons under existing law. They can do this by "inhabiting" the form of entities that are indisputably legal persons under current law, such as limited liability companies (LLCs). When they do this, they become what I call *autonomous organizations* and gain the practical abilities to function as legal persons. This can happen under current law, without statutory reform.

I should make a few things clear at the outset. First, this book is not about constitutional rights; the political debate in the United States about

[1] That said, it's not clear that anyone would notice if a robot tried to do these things; at least in the United States, for example, nobody checks birth certificates when deeds to real property are recorded. For more discussion about the capabilities of software that simply defies the law or exercises rights that the law does not formally give it, see Chapter 4.

the full range of possible legal rights of corporations is not of significant concern to us here. When I use the term *legal personhood*, I use it the same way as a lawyer or legal academic who works in the private law – that is, the legal subjects, like contract law, tort law, and property law, that govern the small-scale interactions among individual actors in our legal system. Legal personhood, for the purpose of this book, confers only the most basic legal capabilities, including the opportunities to enter into a contract, to file a lawsuit, to be sued, to own property, to serve as a legal principal (e.g., an employer), or to serve as a legal agent (e.g., an employee).[2] Whether such a legal person also has the right to freedom of speech (or any other right granted by the US Constitution or a state constitution)[3] is a matter far beyond the scope of this book. Many legal academics, and certainly popular commentary on law, neglect these basic private-law rights in favor of the politicized public controversies associated with such matters as campaign finance or religious freedom, but the only focus here is on the basic private-law rights. Putting aside political attention, private-law topics and basic interactions among people and organizations in our society underlie the vast majority of relationships, legal disputes, and economic activities, so the focus I suggest is not strange; it is just different from the political dimensions that the concept of legal personhood has occasionally assumed.

Second, just as a note about this book's scope, I interchangeably discuss such real and speculative systems as robots (intelligent or not), conventional and novel artificial intelligence, distributed software, drones, and so forth. One of my chief contentions, which is developed in detail in Chapters 2 and 3, is that I am proposing a legal technique that can give functional rights to software of all kinds, without any specific regard to how "intelligent" the software is. I regard this as an advantage of my approach – a point I develop in more detail in Chapters 3 and 5 – because it avoids requiring that progress in the

[2] A more formal definition that conveys a similar meaning is that a legal person, for the purposes of this book, is anything to which the law can ascribe any Hohfeldian jural relation, such as a right, duty, or power. *See* Wesley N. Hohfeld, *Fundamental Legal Conceptions as Applied in Judicial Reasoning*, 26 YALE L.J. 710 (1917) (defining and classifying "jural relations"). Note that any Hohfeldian relation is sufficient; not all subjects of personhood need to have the same collection of rights, powers, etc.

[3] *See, e.g.*, Burwell v. Hobby Lobby Stores, Inc., 573 U.S. 682 (2014); Citizens United v. Federal Election Commission, 558 U.S. 310 (2010).

law depend on practical and philosophical questions about the properties of potentially intelligent systems (e.g., measurements of intelligence or questions about the necessary conditions for conscious experience) that have so far proved intractable.[4] Indeed, for many purposes it does not even matter whether a system uses "software" or not; the techniques I propose could equally well be used to give legal rights to animals, to technologically augmented biological systems, or to a range of other physical systems; the only requirement, as I discuss in more detail in Chapters 2 and 3, is that a system have a verifiable physical state.[5]

My main argument in this book is that law already permits software (or arbitrarily sophisticated combinations of software and humans, or software and hardware, etc.) to interface relatively easily with the rest of the legal system in ways that may seem radical but are likely to be useful and adaptive and unlikely to be intractable or dangerous. My general claim is limited to that statement; for example, I don't deny that widespread use of autonomous organizations would raise new challenges for the legal system and put pressure on conventional legal structures (as discussed in Chapter 6).

For some reason, in the popular imagination science fiction seems to have made people afraid of artificial intelligence, and this has led to an instinctive resistance to some of my ideas. All I can say is that if malicious AI takes over the world, it's unlikely to do it by buying houses – and if that's the path for AI to conquer humanity, it ends up seeming like a relatively minor revolution or form of oppression. I can

[4] In laying out my arguments in this book, I intend to make no specific predictions about the arrival of what is called strong artificial intelligence, or generally intelligent software on par with humans' adaptive cognitive abilities. My approach does not depend on any particular resolution to philosophical or practical problems about what counts as intelligence, how it is measured, or what moral obligations are owed to different types of potentially intelligent or other systems. As I describe in more detail in Chapters 2 and 3, an autonomous organization can just as easily be powered by today's software as by a fully intelligent, strong artificial intelligence. As I suggest throughout the book, I regard this flexibility as an advantage of my legal approach over those that would depend on determining precise sets of capabilities that software must demonstrate before it is permitted to have binding legal effects.

For an introduction to the intractability of problems like the nature and conditions of consciousness, Explaining Consciousness: The Hard Problem (Jonathan Shear ed., 1997) is a nice collection of essays.

[5] *Verifiable* is a term of art in the economic theory of contracts, defined and discussed in Chapter 2.

think of worse things than automating away the current class of landlords that exists in many parts of the world.

1.1 EVEN IF SOFTWARE CAN BUY A HOUSE, WHY WOULD IT WANT TO?

As the brief discussion so far suggests, some of my arguments may seem to be at an eerie crossroads between science fiction and fact. Fiction has long depicted artificial systems with the same type of intelligence and desires as humans, and of course if such systems become real, they will have their own reasons for wanting to engage with the legal system.[6] Even today, however, software can already act on its own in ways that are significant to individuals, businesses, and other existing legal actors. While there may be little obvious need today for software to buy a house in its own right, it is easy to imagine a for-profit or nonprofit company wishing to set up a mechanism under which software can automatically and independently broker transactions for online resources with a commitment that the original humans involved in setting the software in motion will not interfere with its operation. Or perhaps a wealthy individual wishes to set up an algorithmic mechanism to dispose of their assets when they die – a drive that has gone back at least centuries[7] and has found a convenient legal implementation in the form of common-law trusts that direct the use of funds. Such trusts amount to algorithms that the drafter hopes will be implemented by humans in the future.

Understanding the porousness of the boundary between algorithm and trust, or algorithm and LLC operating agreement, will make the

[6] This book, accordingly, provides a blueprint for such an artificial system to achieve meaningful capabilities under the legal system – not the full range of humans' rights, such as the right to vote or marry, but meaningful and durable interaction with the private law. If you are such a system, you're welcome.

[7] Alexander Pope, in making fun of the rich, pointed out in one of his Moral Essays that "thousands . . . Die, and endow a College or a Cat." Alexander Pope, Of the Use of Riches: An Epistle to the Right Honourable Allen Lord Bathurst 6 (1732). People cannot, at least in the United States, leave money to their pets, but they can set up trusts that specify conditions under which the money is to be used to benefit a pet. As I observe in the text here and in Chapter 2, such a scheme in a conventional trust is an algorithm, though historically it has been an algorithm implemented by groups of humans.

arguments throughout this book more intuitive. Every contract, company operating agreement, or other legal instrument operates with many of the features of an algorithm; it is just an algorithm that is implemented by private actors under the supervision of courts, rather than by code under the supervision of an operating system. And, of course, code does not operate in a vacuum; it is supervised internally by software and hardware constraints, but it is also supervised by people who can alter it, terminate its execution, and so on. Even formal code can be run by humans – slowly and tediously, perhaps, but meaningfully in the same way as it might be implemented by a virtual machine, microcode, straight central processing unit (CPU) hardware, or any other technological substrate.[8] Just as the precise hardware implementation for code does not matter to many programmers in many circumstances,[9] we may find that there is little practical difference between a high-level lawyer-written "algorithm" implemented in an LLC operating agreement and low-level C++ code "recognized" and given legal effect by such an agreement.

So, why might it be useful for software to have basic legal rights? The general answer is that any existing legal actor may find it useful to set in motion an algorithm that is backed by the structures of law, such as the enforcement of contracts or the recognition of property. In particular, it may be useful to set such an algorithm in motion while making a legally enforceable commitment that the original actor may not change or continue to influence the operation of the algorithm. Examples of this abstract concept may be helpful, so consider the

[8] This point has perhaps been obscured more than elucidated by John Searle's "Chinese Room" argument, *cf. infra* note 229, as David Chalmers has suggested in THE CONSCIOUS MIND (rev. ed. 1997) and elsewhere. *See* David Chalmers, *Subsymbolic Computation and the Chinese Room*, in THE SYMBOLIC AND CONNECTIONIST PARADIGMS: CLOSING THE GAP 25 (John Dinsmore ed., 1992).

[9] See, for example, Sun Microsystems's slogan for the Java programming language in the 1990s: "Write Once, Run Anywhere." Nick Langley, *Write Once, Run Anywhere?*, ComputerWeekly.com, May, 2, 2002, https://www.computerweekly.com/feature/Write-once-run-anywhere [https://perma.cc/5L9L-38S8]. The purpose of this slogan was to indicate that Java provides programmers with the opportunity to write code that could run on a variety of computer hardware and operating systems without having to pay attention to the details of those execution environments. This capability is achieved by a "virtual machine," a layer of abstraction that aims to implement a predictable environment for Java software in multiple hardware and operating-system environments.

following illustrations of possibilities; the list is meant only to be suggestive, not complete:

- *Algorithmic Charities*. Suppose a wealthy philanthropist wishes to use funds to enable projects in the future on preconceived terms; for example, the donor wants to allocate money to grant proposals that meet certain verifiable characteristics. Conventionally, the donor might set up a foundation that employs people to review grants and award funds. A modern donor might imagine, however, that they can do better – on grounds of efficiency, predictability, organizational longevity, and so forth – if the funds are awarded in the future based on algorithmic rather than human verification. To be sure, in today's environment and with today's level of technology, there would be risks that the algorithm could be hacked or otherwise abused, but the donor might weigh those risks against the risks of (say) corrupt or incompetent potential employees and determine that the risks of algorithms are manageable or suit the donor's tastes or goals better than a conventionally structured foundation.
- *Automated Brokers or Industry Self-Regulators*. Imagine several players in an industry want to create a centralized organization that helps them govern themselves or provides common services on verifiable terms. Again, the interested parties could rely on humans, but they might find that relying on algorithms suits their needs better.[10] They might, accordingly, want to set up an independent organization that is partially (or perhaps "fully," in some sense) governed by pre-agreed algorithms. Similarly, "accrediting" organizations – like the Non-GMO Project (which certifies that food meets certain standards) or B Lab (which does the same for companies), but perhaps more readily an organization that accredits something that can be evaluated online, like the uptime or network latency of cloud-computing services – may choose to operate by algorithm rather than by human judgment, either for reasons of administrative simplicity or to prevent the likelihood of human corruption (if that likelihood is judged to be less significant than the risks that the algorithm will be compromised, abused, or become stale in view of real-world changes).
- *Verifiable, Participatory Private Democracy*. A company or other entity – maybe even a municipality – may wish to sponsor a project that involves

[10] This sort of situation is commonly proposed as an application of blockchains, in what has always seemed to me as, at bottom, a fundamental misunderstanding of the costs and benefits of blockchains. Indeed, it often seems as if even relatively sophisticated commentators use the term "blockchain" when they mean only "database plus authentication infrastructure" or some other combination of technologies that has existed for decades.

contributions from others. To create demand for the project (which perhaps the company or other entity expects to benefit from in some collateral way), the company may want to set in motion a system that is governed by public vote, the extent of public contributions, or some other objectively verifiable characteristics of its participants or their activities. That is, the company may wish to bind itself in order to demonstrate that it will not take control of or corrupt the project in an ongoing way. To make this example less abstract, I have in mind something like a startup company that has a significant social mission that requests contributions of time or effort from the public. Conventionally, the company might bind itself by contract or interact with third-party certifications in various ways, but if its project or the constraints on the project are more complex, or alternatively if the startup does not have access to third-party certifications, it may wish to commit the regulation of the project to open-source code that it has a limited (or no) legal ability to modify in the future.[11]

- *Constrained Regulation.* Suppose a municipality wants to set up a new system for fines for misbehavior. It expects, hypothetically, that an automated system of enforcement will be more popular, politically expedient, safe, equitable, or effective than one that involves the city police. It commits, therefore, to a system in which fines are assessed based on algorithmic processing of technological inputs – say, a speed camera (for traffic violations), a noise meter (for loud parties), or information about utility usage (for environmental regulation). Of course, the latter example – utility "regulation" – is already commonplace; no human needs to review an electricity meter for many public or private utility companies to "decide" to charge a different amount (or even a different price rate) based on monthly usage, but the use of legally autonomous algorithms can of course generalize this approach and lead to much more complicated forms of automatic regulation. Note that in examples like this, there is a range of possibilities for the autonomy of the algorithm; software can range from a simple, conventional tool to a mechanism that provides a commitment by regulators that some

[11] Again, because of what I regard as a misunderstanding, blockchain-based technologies may occur to some readers as a technical solution to this problem, but any number of other technical solutions could be more suitable – and, more to the point, the most effective or efficient solution could well involve an interaction between technology and law, rather than a reliance on technology alone. Also, as experience has shown, not all blockchains are beyond the reach of human modification, leading to a potential need for legal enforcement of the autonomy or immutability of the software.

feature of regulations will not change or will not be administered by humans.

- *Constrained Outsourcing in Business*. For reasons of efficiency or administrative ease, a business may wish to produce a self-sufficient subsidiary that operates on terms that are predictable for the public or for specific third parties. For example, a book publisher may wish to use an independent automated broker to license reuse of works for the public, and it may wish to make a credible representation to its authors and other contractors that the decisions to license the work will be made by an independent but adaptive algorithm.

What all these cases have in common is that an existing legal entity may want an algorithm to have binding legal effect and, to various degrees, to operate independently from the algorithm's original creators or implementors. As some of the examples have suggested, the notion that "algorithms" may have legal effect may seem, in some contexts, almost mundane. In a way, that is my point: a program that reads an electric meter over a mobile-telephone signal and produces a legally binding bill is not different, in concept, from a robot that buys a house (or, more practically today, software that opens or transacts in a financial account). One thing that this book's analysis provides is a blueprint that permits algorithms to interact with the conventional legal system without legal reform – for example, without requiring new statutes that will inevitably lag behind technological developments.

Note that the examples above are somewhat different in structure and intent from "the DAO," an Ethereum-based venture fund whose name stood for "Decentralized Autonomous Organization." This book is not specifically an exploration of any one type of technology-based governance mechanism, nor is it limited to a discussion of "decentralized" organizations, but the book's principles can be used as a way to harmonize decentralized technological governance with the legal system. As it stands, the legal treatment of decentralized technology-based organizations is unclear and perhaps not of primary interest to all the participants – but the organizations are likely to be either (1) legally nothing, (2) unincorporated nonprofit associations, or (3) general partnerships, and each possibility has dramatically different implications for the rights and liabilities of the various participants. This book adds a new possibility (a registered legal entity that the software can

inhabit), and its transactional techniques can be used just as easily by the examples listed above, by a fully intelligent AI, and by a "decentralized" investment fund implemented through blockchain technology or some other peer-to-peer mechanism; the flexibility of this book's techniques is one of their strengths.[12]

As our experience with complex, interconnected software grows – and in particular as software increasingly interacts with the real world through diverse sensors and actuators – the advantages of a flexible framework for understanding and harmonizing the role of algorithms in the legal system becomes more important.

1.2 ALGORITHMS IN CONVENTIONAL LEGAL CONTEXT

The techniques proposed in this book, mainly in Chapters 2 and 3, can be expressed in a general form and in a much more specific one. The general form, put simply, is that contracts and other legal instruments can recognize the verifiable state of algorithms and thereby give legal effect to those algorithms. This is an extraordinarily powerful notion that will, I hope, seem almost obvious after the fact. One of the building blocks of contract law is the notion of *conditions* – that is, verifiable states that must obtain for a duty to arise under a contract (or alternatively that extinguish or terminate a previously arisen duty). For example, a contract between a buyer and a seller of goods might provide

[12] As this book was going to press, Wyoming was passing a new statute directly recognizing decentralized autonomous organizations as a new type of legal entity. The Wyoming statute offers another type of regulatory possibility for decentralized organizations, and it serves as strong practical support for my argument in Chapter 4 that the legal system is not opposed to autonomous organizations.

As noted in the text, the ideas in this book are aimed at recognizing *autonomous* organizations in law; they have little concern about whether the organizations are also *decentralized*. Avoiding a reliance on the concept of centralization lends flexibility to my approach because the concept of centralization is a problematic one. For example, blockchains (and blockchain-based systems) are often not nearly as decentralized as people commonly believe; the history of the DAO is an example because it led to a relatively small group of people altering the course of the Ethereum blockchain in order to reverse a hack that compromised the DAO's funds. This intervention required the consent of fewer people than might be necessary to approve similarly extraordinary action by the US government, an organization sometimes erroneously regarded as centralized.

that the seller must deliver 500,000 goods of a particular type, but only if the seller's supply is adequate to do so. To a programmer, this is the same sort of "condition" that is familiar in most programming languages; it is just an if-statement.[13] Conditions in conventional contracts are often very simple to analyze, but the law permits them to be arbitrarily complex, as long as they can be verified.[14] Accordingly, nothing stands in the way of a contract referencing or "incorporating" an algorithm, whether a simple one running on a single known computer, a complex one running in distributed fashion, or an "intelligent" one running in a robot's head or on a fast-moving drone. Chapter 2 elaborates the general form of this idea.

The specific form of the same idea is that businesses are governed by agreements, such as *articles of incorporation* or *operating agreements*. A chief contention in this book is that modern LLC law, at least in

[13] For example, in Java-ish pseudocode, an algorithm representing the contract between the buyer and the seller could plausibly be written as

```
if (adequateSupply())
  sellerObligation = true;
```

where `adequateSupply` is a function that evaluates, algorithmically, the seller's capacity. To a lawyer, of course, the standard expression of the condition might be more like, "Seller shall supply buyer with 500,000 sporks, provided that seller has access to that many sporks on July 2 in the usual course of its business," or depending on the condition it could take the form of a routine force-majeure clause.

[14] There is an interesting legal-theoretical question, not well explored in the legal-academic literature, about what a computer technologist might call the runtime capacity of courts. (The problem can be framed in more formal computer-scientific terms as a question about the bounds, if any, that the law might impose on the worst-case time complexity of legal disputes.) Contract law has doctrines that prevent vague contracts, or ones where courts cannot determine a remedy, from being enforced, but there is no conventional doctrine that prevents a contractual obligation from arising merely because it would take too much effort or energy for the court to determine what the obligation should be. This may be because, in the common law's adversarial mode of dispute resolution, the parties are charged with doing most of the work of determining what those obligations should be.

Courts have other, less formal ways to address this sort of problem, such as strongly advising the parties to mediate their dispute, cutting off civil discovery, or relying on burdens of proof or other techniques of civil procedure to simplify the questions that it needs to answer. Courts can conceivably rely on expert witnesses or special masters as well. It is worth pointing out that it is conceivable that legal doctrine is held back here by judicial pride; it would be a rare judge that says to litigating parties, "Your contract is just too sophisticated and intricate for me to understand. I don't know enough, or am not smart enough, to resolve your dispute." Courts have held that legal doctrine is too difficult for reasonable *lawyers* to understand, *see* Lucas v. Hamm, 364 P.2d 685 (Cal. 1961) (holding that the rule against perpetuities is so difficult to understand that it is not malpractice for a lawyer to misunderstand it), but it is presumably harder for judges to say the same thing about themselves.