

# 7

## The Enclosure of Science and Technology: Two Case Studies

Over the last forty years, much has changed in the way that scientific research and technological development are organized, funded, and institutionally arranged. Much has also changed in the type of scientific and technical material that is covered by intellectual property rights, the ways that material is covered, the parties who hold the rights, and the state of research and development at which rights claims are made. Many academics who study both science's organizational structure and the intellectual property claims that surround it are concerned about the results. To say this is not to conjure up a tragically lost world of pure research science, untainted by property claims or profit motives. That world never existed and it is probably a good thing too. Intellectual property rights, and the profit motive more generally, have a vital and beneficial role in moving innovations from lab bench to bedside, from computer simulation to actual flight. The question is not *whether* intellectual property rights are useful as part of scientific and technological development. The question is what type of rights they should be, where in the research process those rights are best deployed, how they should coexist with

state funded basic scientific and technological research, how broad they should be, how they should deal with new technologies, how long they should last, how they should treat follow-on innovations.

I cannot hope here to answer all those questions, though some fascinating research has begun the process. Instead, as with the music chapter, I will offer a case study—actually two case studies—that try to illuminate the process I am describing, to illustrate its pitfalls and its strange and unintended consequences.

The two defining technologies of the last thirty years are biotechnology and the networked computer. Each is both product and platform. Innovations themselves, they are also constitutive technologies that enable still more innovations. But at several historical moments in the development of each we came perilously close to breaking technology with law.<sup>1</sup> Some would say that it was not just a close shave: we actually have hampered or limited the full potential of technology, slowing down its dynamism with a host of overbroad software patents, gene patents, and materials transfer agreements. Others are more optimistic. They think that a series of rapid improvisations by courts, scientists, programmers, and businesspeople has largely mitigated any problems caused by the process of legal expansion.<sup>2</sup> But if mistakes were made, it is important to know what they were lest we continue or repeat them. If there were “fixes,” it is important to know if they can be replicated.

So were there mistakes? If so, have they been fixed, and how? Drawing on an article I co-wrote with my brilliant colleague Arti Rai,<sup>3</sup> this chapter suggests some answers to those questions by sketching out some details of the legal history of those technologies, concluding with a discussion of a single promising new technology that shares aspects of both—synthetic biology. The answers are important. Behind the abstract words “innovation” or “technological development” there are lives saved or lost, communicative freedoms expanded or contracted, communities enabled or stunted, wealth generated or not. The subject would benefit from informed, sophisticated, democratic attention. It is not something you want to leave a host of lawyers and lobbyists to decide among themselves.

#### A MACHINE THAT CONTAINS ALL OTHER MACHINES

Imagine a person staring at an infinite roll of paper tape. On the paper are symbols in some alphabet or number system. The reader carries out simple,

operable instructions on the basis of that data. “Add together the next two digits you are presented with and write down the answer. If the answer is odd, go to step 2. If the answer is even, go to step 3.” Now replace the person with a mechanical head that can “read” the instructions, carry out the desired operations, and write the answer down. The British mathematician Alan Turing imagined something like this—a little more complicated, perhaps, but fairly similar. What is it? We have the reading head, the set of instructions, the data on which the instructions are to be performed, the record of the result, and some kind of “state table” that tells the machine where it is in the process. These are the component parts of Turing machines—or as we know them better, computers. More accurately, Turing machines are a method of simulating the operation of computers, a metaphor that enables us to imitate their logical processes. In the words of Wikipedia, “despite their simplicity—[they] can be adapted to simulate the logic of any computer that could possibly be constructed.” And to give lawyers fits. But that is getting ahead of ourselves.

In Greek mythology, Procrustes had a bed to which he fitted its prospective occupants, whether they liked it or not. The tall were trimmed down. The short stretched on the rack. Intellectual property lawyers have many similarities to Procrustes. The technologies that are brought before them are made to fit the conceptual boxes the law provides, boxes with names such as “copyright” and “patent.” Occasionally, new conceptual boxes are made, but—for very good reasons—most of the time we stick with the boxes we have. As with Procrustes, things do not always fit and the process can be distressing for its subjects.

It is important to realize that the process of trimming and stretching can be done well or badly. If it is done really badly, the technology is stunted, deformed, even destroyed. If it is done well, the law aids the development of the technology in exactly the happy way described in Chapter 1. What did our Procrustean legal system do with computers and computer science?

I will focus on software—the set of instructions the machine is to perform. How should we think of it? Software is written down by programmers. It is recorded first in a form readable to humans, or at least geeks. Then, through a series of transformations, it is turned into the machine code, the ones and zeros that will operate the computer. But at its root it can be understood through the metaphor of the simple list of instructions to be carried out in order, just as with the Turing machine and its infinite tape.

How should we fit software into the categories of intellectual property? We have “writing,” fixation in some medium of symbols that can be read by

others—both machine and human. Writing is normally the domain of copyright. Are computer programs copyrightable? All kinds of problems present themselves. At least in the United States, copyright covers expression. As I pointed out in a previous book, at its base is the conception of the romantic author impressing her uniqueness of spirit on the work at the moment of writing. It is that expressive choice, not the facts or ideas on which the work is based, that copyright covers. And it is *only* original expression that copyright covers. It does not cover purely functional objects, systems, processes, or methods of operation. One cannot copyright the coat hanger, the mousetrap, or long division. One cannot even copyright a “sculpture” if the main function of its design is to serve as a bicycle rack. Admittedly, one can copyright some expressive works that serve a practical purpose. A book about how to do double-entry bookkeeping is copyrightable. Yet copyright covers only the expressive choices used in selecting the words to explain the method, and the images to represent it, not the methods it describes or the facts or ideas it contains. Can copyright cover computer programs? Should we see them as copyrightable how-to books or as uncopyrightable machines made of words?

Machines and other functional innovations are normally the domain of patent rights. One can patent the mousetrap, and then one gets an exclusive right to the actual mechanically enabled method of catching mice, not just the artistic flourishes on the blueprint. Patents have more demanding criteria than copyrights. The invention needs to be novel and have utility, or usefulness; I cannot get a patent over something that would have been an obvious idea to an insider in the relevant field of technology, a “person having ordinary skill in the art,” or PHOSITA, in the jargon of patent lawyers. But once I get my patent, it gives me a very strong power to exclude others from the invention—even if they came up with it independently. The right lasts for twenty years. Follow-on innovators who improve on my idea can get a patent on that improvement. They can block me from using the improvement. I can block them from using the original invention. Thus we have an incentive to negotiate if either of us wants to bring the improved innovation to market.

So where did software fit? Was it copyrightable writing or patentable invention? There are two issues here. The first is whether there should be any intellectual property rights over software at all. The basic case for that proposition is simple, a classic example of the public goods problem described in the first chapter. Software costs money to create, but is cheap to copy. When a youthful

Bill Gates wrote his 1976 letter to the wonderfully named *Dr. Dobb's Journal of Computer Calisthenics & Orthodontia*, he put the point clearly.

Who can afford to do professional work for nothing? What hobbyist can put 3-man years into programming, finding all the bugs, documenting his product and distribute it for free? The fact is, no one besides us has invested a lot of money into hobby software. We have written 6800 BASIC, and are writing 8080 APL and 6800 APL, but there is very little incentive to make this software available to hobbyists. Most directly, the thing you do is theft.<sup>4</sup>

He signed the letter “Bill Gates, General Partner, Micro-Soft.” The hyphen would disappear in time. The philosophy stuck around.

Though there are quibbles about the facts in Gates’s letter—critics claim he himself did a lot of free riding on public domain code and government-funded computer time—his basic point is that software needs to be protected by (enforceable) property rights if we expect it to be effectively and sustainably produced. Some software developers disagree. But assuming one concedes the point for the sake of argument, there is a second question: should software be covered by copyright or patent, or some unidentified third option?

In practice, software ended up being covered by both schemes, partly because of actions by Congress, which included several references to software in the Copyright Act, and partly as a result of decisions by the Copyright Office, the Patent and Trademark Office, and judges. One could copyright one’s code and also gain a patent over the “nonobvious,” novel, and useful innovations inside the software.

At first, it was the use of copyright that stirred the most concern. As I explained in the last chapter, copyright seems to be built around an assumption of diverging innovation—the fountain or explosion of expressive activity. Different people in different situations who sit down to write a sonnet or a love story, it is presumed, will produce very different creations rather than being drawn to a single result. Thus strong rights over the resulting work are not supposed to inhibit future progress. I can find my own muse, my own path to immortality. Creative expression is presumed to be largely independent of the work of prior authors. Raw material is not needed. “Copyright is about sustaining the conditions of creativity that enable an individual to craft out of thin air an *Appalachian Spring*, a *Sun Also Rises*, a *Citizen Kane*.”<sup>5</sup>

There are lots of reasons to doubt that this vision of “creation out of nothing” works very well even in the arts, the traditional domain of copyright law. The story of Ray Charles’s “I Got a Woman” bears ample witness to those

doubts. But whatever its merits or defects in the realm of the arts, the vision seems completely wrongheaded when it comes to software. Software solutions to practical problems do converge, and programmers definitely draw upon prior lines of code. Worse still, as I pointed out earlier, software tends to exhibit “network effects.” Unlike my choice of novel, my choice of word processing program is very strongly influenced, perhaps dominated, by the question of what program other people have chosen to buy. That means that even if a programmer could find a completely different way to write a word processing program, he has to be able to make it read the dominant program’s files, and mimic its features, if he is to attract any customers at all. That hardly sounds like completely divergent creation.

Seeing that software failed to fit the Procrustean bed of copyright, many scholars presumed the process of forcing it into place would be catastrophic. They believed that, lacking patent’s high standards, copyright’s monopolies would proliferate widely. Copyright’s treatment of follow-on or “derivative” works would impede innovation, it was thought. The force of network effects would allow the copyright holder of whatever software became “the standard” to extract huge monopoly rents and prevent competing innovation for many years longer than the patent term. Users of programs would be locked in, unable to shift their documents, data, or acquired skills to a competing program. Doom and gloom abounded among copyright scholars, including many who shared Mr. Gates’s basic premise—that software should be covered by property rights. They simply believed that these were the wrong property rights to use.

Copyright did indeed cause problems for software developers, though it is hard to judge whether those problems outweighed the economic benefits of encouraging software innovation, production, and distribution. But the negative effects of copyright were minimized by a remarkably prescient set of actions by courts and, to a much lesser extent, Congress, so that the worst scenarios did not come to pass. Courts interpreted the copyright over software very narrowly, so that it covered little beyond literal infringement. (Remember Jefferson’s point about the importance of being careful about the scope of a right.) They developed a complicated test to work out whether one program infringed the details of another. The details give law students headaches every year, but the effects were simple. If your software was similar to mine merely because it was performing the same function, or because I had picked the most efficient way to perform some task, or even because there was market demand for doing it that way, then none of those similarities counted for the

purposes of infringement. Nor did material that was taken from the public domain. The result was that while someone who made literal copies of Windows Vista was clearly infringing copyright, the person who made a competing program generally would not be.

In addition, courts interpreted the fair use doctrine to cover “decompilation”—which is basically taking apart someone else’s program so that you can understand it and compete with it. As part of the process, the decompiler had to make a copy of the program. If the law were read literally, decompilation would hardly seem to be a fair use. The decompiler makes a whole copy, for a commercial purpose, of a copyrighted work, precisely in order to cause harm to its market by offering a substitute good. But the courts took a broader view. The copy was a necessary part of the process of producing a competing product, rather than a piratical attempt to sell a copy of the same product. This limitation on copyright provided by fair use was needed in order to foster the innovation that copyright is supposed to encourage. This is a nice variation of the Sony Axiom from Chapter 4.

These rulings and others like them meant that software was protected by copyright, as Mr. Gates wanted, but that the copyright did not give its owner the right to prevent functional imitation and competition. Is that enough? Clearly the network effects are real. Most of us use Windows and most of us use Microsoft Word, and one very big reason is because everyone else does. Optimists believe the lure of capturing this huge market will keep potential competitors hungry and monopolists scared. The lumbering dominant players will not become complacent about innovation or try to grab every morsel of monopoly rent, goes the argument. They still have to fear their raptor-like competitors lurking in the shadows. Perhaps. Or perhaps it also takes the consistent threat of antitrust enforcement. In any event, whether or not we hit the optimal point in protecting software with intellectual property rights, those rights certainly did not destroy the industry. It appeared that, even with convergent creativity and network effects, software could be crammed into the Procrustean bed of copyright without killing it off in the process. Indeed, to some, it seemed to fare very well. They would claim that the easy legal protection provided by copyright gave a nascent industry just enough protection to encourage the investment of time, talent, and dollars, while not prohibiting the next generation of companies from building on the innovations of the past.

In addition, the interaction between copyright and software has produced some surprising results. There is a strong argument that it is the fact that software is copyrightable that has enabled the “commons-based creativity” of free

and open source software. What does commons-based creativity mean? Basically, it is creativity that builds on an open resource available to all. An additional component of some definitions is that the results of the creativity must be fed back into the commons for all to use. Think of English. You can use English without license or fee, and you can innovate by producing new words, slang, or phrases without clearance from some Academie Anglaise. After you coin your term, it is in turn available to me to build upon or to use in my own sentences, novels, or jokes. And so the cycle continues. As the last chapter showed, for the entire history of musical creativity until the last forty years or so, the same had been true of at least a low level of musical borrowing. At the basic level of musical phrases, themes, snatches of melody, even chord structures, music was commons-based creativity. Property rights did not reach down into the atomic structure of music. They stayed at a higher level—prohibiting reproduction of complete works or copying of substantial and important chunks. So in some areas of both music and language, we had commons-based creativity because there were no property rights over the relevant level. The software commons is different.

The creators of free and open source software were able to use the fact that software is copyrighted, and that the right attaches automatically upon creation and fixation, to set up new, distributed methods of innovation. For example, free and open source software under the General Public License—such as Linux—is a “commons” to which all are granted access. Anyone may use the software without any restrictions. They are guaranteed access to the human-readable “source code,” rather than just the inscrutable “machine code,” so that they can understand, tinker, and modify. Modifications can be distributed so long as the new creation is licensed under the open terms of the original. This creates a virtuous cycle: each addition builds on the commons and is returned to it. The copyright over the software was the “hook” that allowed software engineers to create a license that gave free access and the right to modify and required future programmers to keep offering those freedoms. Without the copyright, those features of the license would not have been enforceable. For example, someone could have modified the open program and released it without the source code—denying future users the right to understand and modify easily. To use an analogy beloved of free software enthusiasts, the hood of the car would be welded shut. Home repair, tinkering, customization, and redesign become practically impossible.

Of course, if there were no copyright over software at all, software engineers would have other freedoms—even if not legally guaranteed open access

to source code. Still, it was hard to deny that the extension of the property regime had—bizarrely, at first sight—actually enabled the creation of a continuing open commons. The tempting real estate analogy would be environmentalists using strong property rights over land to guarantee conservation and open access to a green space, where, without property rights, the space could be despoiled by all. But as I have pointed out earlier, while such analogies may help us, the differences between land and intellectual property demand that they be scrutinized very carefully. It is hard to overgraze an idea.

So much for copyright. What about patents? U.S. patent law had drawn a firm line between patentable invention and unpatentable idea, formula, or algorithm. The mousetrap could be patented, but not the formula used to calculate the speed at which it would snap shut. Ideas, algorithms, and formulae were in the public domain—as were “business methods.” Or so we thought.

The line between idea or algorithm on the one hand and patentable machine on the other looks nice and easy. But put that algorithm—that series of steps capable of being specified in the way described by the Turing machine—onto a computer, and things begin to look more complex. Say, for example, that algorithm was the process for converting miles into kilometers and vice versa. “Take the first number. If it is followed by the word miles, then multiply by  $8/5$ . If it is followed by the word kilometers, multiply by  $5/8$  . . .” and so on. In the abstract, this is classic public domain stuff—no more patentable than  $E=mc^2$  or  $F=ma$ . What about when those steps are put onto the tape of the Turing machine, onto a program running on the hard drive of a computer?

The Court of Appeals for the Federal Circuit (the United States’s leading patent court) seems to believe that computers can turn unpatentable ideas into patentable machines. In fact, in this conception, the computer sitting on your desk becomes multiple patentable machines—a word processing machine, an e-mail machine, a machine running the program to calculate the tensile strength of steel. I want to stress that the other bars to patentability remain. My example of mile-to-kilometer conversion would be patentable subject matter but, we hope, no patent would be granted because the algorithm is not novel and is obvious. (Sadly, the Patent and Trademark Office seems determined to undermine this hope by granting patents on the most mundane and obvious applications.) But the concern here is not limited to the idea that without a subject matter bar, too many obvious patents will be granted by an overworked and badly incentivized patent office. It is that the patent was supposed to be granted at the very end of a process of investigation and scien-

tific and engineering innovation. The formulae, algorithms, and scientific discoveries on which the patented invention was based remained in the public domain for all to use. It was only when we got to the very end of the process, with a concrete innovation ready to go to market, that the patent was to be given. Yet the ability to couple the abstract algorithm with the concept of a Turing machine undermines this conception. Suddenly the patents are available at the very beginning of the process, even to people who are merely specifying—in the abstract—the *idea* of a computer running a particular series of algorithmic activities.

The words “by means of a computer” are—in the eyes of the Federal Circuit—an incantation of magical power, able to transubstantiate the ideas and formulae of the public domain into private property. And, like the breaking of a minor taboo that presages a Victorian literary character’s slide into debauchery, once that first wall protecting the public domain was breached, the court found it easier and easier to breach still others. If one could turn an algorithm into a patentable machine simply by adding “by means of a computer,” then one could turn a business method into something patentable by specifying the organizational or information technology structure through which the business method is to be implemented.

If you still remember the first chapters of this book, you might wonder why we would want to patent business methods. Intellectual property rights are supposed to be handed out only when necessary to produce incentives to supply some public good, incentives that otherwise would be lacking. Yet there are already plenty of incentives to come up with new business methods. (Greed and fear are the most obvious.) There is no evidence to suggest that we need a state-backed monopoly to encourage the development of new business methods. In fact, we *want* people to copy the businesses of others, lowering prices as a result. The process of copying business methods is called “competition” and it is the basis of a free-market economy. Yet patent law would prohibit it for twenty years. So why introduce patents? Brushing aside such minor objections with ease, the Court of Appeals for the Federal Circuit declared business methods to be patentable. Was this what Jefferson had in mind when he said “I know well the difficulty of drawing a line between the things which are worth to the public the embarrassment of an exclusive patent, and those which are not”? I doubt it.

It is commonplace for courts to look at the purpose of the law they are enforcing when seeking to understand what it means. In areas of regulation which are obviously instrumental—aimed at producing some particular result

in the world—that approach is ubiquitous. In applying the antitrust laws, for example, courts have given meaning to the relatively vague words of the law by turning to economic analysis of the likely effects of different rules on different market structures.

Patent law is as instrumental a structure as one could imagine. In the United States, for example, the constitutional authorization to Congress to pass patent and copyright legislation is very explicit that these rights are to be made with a purpose in view. Congress has the power “to promote the progress of science and useful arts, by securing for limited times to authors and inventors the exclusive right to their respective writings and discoveries.” One might imagine that courts would try to interpret the patent and copyright laws with that purpose, and the Jefferson Warning about its constraints, firmly in mind. Yet utilitarian caution about extending monopolies is seldom to be found in the reasoning of our chief patent court.

The difference is striking. Jefferson said that the job of those who administered the patent system was to see if a patent was “worth the embarrassment to the public” before granting it. The Constitution tells Congress to make only those patent laws that “promote the progress of science and useful arts.” One might imagine that this constitutional goal would guide courts in construing those same laws. Yet neither Jeffersonian ideals nor the constitutional text seem relevant to our chief patent court when interpreting statutory subject matter. Anything under the sun made by man is patentable subject matter, and there’s an end to it. The case that announced the rule on business methods involved a patent on the process of keeping accounts in a “hub-and-spoke” mutual fund—which included multiplying all of the stock holdings of each fund in a family of funds by the respective current share price to get total fund value and then dividing by the number of mutual fund shares that each customer actually holds to find the balance in their accounts. As my son observed, “I couldn’t do that until nearly the end of third grade!”<sup>6</sup>

In theory of course, if the patent is not novel or is obvious, it will still be refused. The Supreme Court recently held that the Court of Appeals for the Federal Circuit has made “nonobvious” too easy a standard to meet.<sup>7</sup> It is unclear, however, whether that judgment will produce concrete effects on actual practices of patent grants and litigation. The Patent and Trademark Office puts pressure on examiners to issue patents, and it is very expensive to challenge those that are granted. Better, where possible, to rule out certain subject matter in the first place. Tempted in part by its flirtation with the “idea made machine” in the context of a computer, the Court of Appeals for

the Federal Circuit could not bring itself to do so. Where copyright law evolved to wall off and minimize the dangers of extending protection over software, patent law actually extended the idea behind software patents to make patentable any thought process that might produce a useful result. Once breached, the walls protecting the public domain in patent law show a disturbing tendency to erode at an increasing rate.

To sum up, the conceptual possibilities presented to copyright and patent law by the idea of a Turing machine were fascinating. Should we extend copyright or patent to cover the new technology? The answer was “we will extend both!” Yet the results of the extension were complex and unexpected in ways that we will have to understand if we want to go beyond the simple but important injunctions of Jefferson and Macaulay. Who would have predicted that software copyrights could be used to create a self-perpetuating commons as well as a monopoly over operating systems, or that judges would talk knowingly of network effects in curtailing the scope of coverage? Who would have predicted that patents would be extended not only to basic algorithms implemented by a computer, but to methods of business themselves (truly a strange return to legalized business monopolies for a country whose founders viewed them as one of the greatest evils that could be borne)?

#### SYNTHETIC BIOLOGY

If you are a reader of *Science*, *PLoS Biology*, or *Nature*, you will have noticed some attractive and bizarre photographs recently. A field of bacteria that form themselves into bull’s-eyes and polka dots. A dim photograph of a woman’s face “taken” by bacteria that have been programmed to be sensitive to light. You may also have read about more inspiring, if less photogenic, accomplishments—for example, the group of scientists who managed to program bacteria to produce artemisinin, a scarce natural remedy for malaria derived from wormwood. Poking deeper into these stories, you would have found the phrase “synthetic biology” repeated again and again, though a precise definition would have eluded you.

What is “synthetic biology”? For some it is simply that the product or process involves biological materials not found in nature. Good old-fashioned biotechnology would qualify. One of the first biotechnology patent cases, *Diamond v. Chakrabarty*, involved some bacteria which Dr. Chakrabarty had engineered to eat oil slicks—not their natural foodstuff.<sup>8</sup> The Supreme Court noted that the bacteria were not found in nature and found them to be

patentable, though alive. According to the simplest definition, Dr. Chakrabarty's process would count as synthetic biology, though this example antedates the common use of the term by two decades. For other scientists, it is the *completely* synthetic quality of the biology involved that marks the edge of the discipline. The DNA we are familiar with, for example, has four "base pairs"—A, C, G, and T. Scientists have developed genetic alphabets that involve twelve base pairs. Not only is the result not found in nature, but the very language in which it is expressed is entirely new and artificial.

I want to focus on a third conception of synthetic biology: the idea of turning biotechnology from an artisanal process of one-off creations, developed with customized techniques, to a true engineering discipline, using processes and parts that are as standardized and as well understood as valves, screws, capacitors, or resistors. The electrical engineer told to build a circuit does not go out and invent her own switches or capacitors. She can build a circuit using off-the-shelf components whose performance is expressed using standard measurements. This is the dream of one group of synthetic biologists: that biological engineering truly become *engineering*, with biological black boxes that perform all of the standard functions of electrical or mechanical engineering—measuring flow, reacting to a high signal by giving out a low signal, or vice versa, starting or terminating a sequence, connecting the energy of one process to another, and so on.

Of course an engineer understands the principle behind a ratchet, or a valve, but he does not have to go through the process of thinking "as part of this design, I will have to create a thing that lets stuff flow through one way and not the other." The valve is the mechanical unit that stands for that thought, a concept reified in standardized material form which does not need to be taken apart and parsed each time it is used. By contrast, the synthetic biologists claim, much of current biotechnological experimentation operates the way a seventeenth-century artisan did. Think of the gunsmith making beautiful one-off classics for his aristocratic patrons, without standardized calibers, parts, or even standard-gauge springs or screws. The process produces the gun, but it does not use, or produce, standard parts that can also be used by the next gunsmith.

Is this portrayal of biology correct? Does it involve some hyping of the new hot field, some denigration of the older techniques? I would be shocked, shocked, to find there was hype involved in the scientific or academic enterprise. But whatever the degree to which the novelty of this process is being subtly inflated, it is hard to avoid being impressed by the projects that this

group of synthetic biologists has undertaken. The MIT Registry of Standard Biological Parts, for example, has exactly the goal I have just described.

The development of well-specified, standard, and interchangeable biological parts is a critical step towards the design and construction of integrated biological systems. The MIT Registry of Standard Biological Parts supports this goal by recording and indexing biological parts that are currently being built and offering synthesis and assembly services to construct new parts, devices, and systems. . . . In the summer of 2004, the Registry contained about 100 basic parts such as operators, protein coding regions, and transcriptional terminators, and devices such as logic gates built from these basic parts. Today the number of parts has increased to about 700 available parts and 2000 defined parts. The Registry believes in the idea that a standard biological part should be well specified and able to be paired with other parts into subassemblies and whole systems. Once the parameters of these parts are determined and standardized, simulation and design of genetic systems will become easier and more reliable. The parts in the Registry are not simply segments of DNA, they are functional units.<sup>9</sup>

Using the Registry, a group of MIT scientists organizes an annual contest called iGEM, the International Genetically Engineered Machine competition. Students can draw from the standard parts that the Registry contains, and perhaps contribute their own creations back to it. What kinds of “genetically engineered machines” do they build?

A team of eight undergraduates from the University of Ljubljana in Slovenia—cheering and leaping onto MIT’s Kresge Auditorium stage in green team T-shirts—won the grand prize earlier this month at the International Genetically Engineered Machine (iGEM) competition at MIT. The group—which received an engraved award in the shape of a large aluminum Lego piece—explored a way to use engineered cells to intercept the body’s excessive response to infection, which can lead to a fatal condition called sepsis. The goal of the 380 students on 35 university teams from around the world was to build biological systems the way a contractor would build a house—with a toolkit of standard parts. iGEM participants spent the summer immersed in the growing field of synthetic biology, creating simple systems from interchangeable parts that operate in living cells. Biology, once thought too complicated to be engineered like a clock, computer or microwave oven, has proven to be open to manipulation at the genetic level. The new creations are engineered from snippets of DNA, the molecules that run living cells.<sup>10</sup>

Other iGEM entries have included *E. coli* bacteria that had been engineered to smell like wintergreen while they were growing and dividing and like bananas when they were finished, a biologically engineered detector that

would change color when exposed to unhealthy levels of arsenic in drinking water, a method of programming mouse stem cells to “differentiate” into more specialized cells on command, and the mat of picture-taking bacteria I mentioned earlier.

No matter how laudable the arsenic detector or the experimental technique dealing with sepsis, or how cool the idea of banana-scented, picture-taking bacteria, this kind of enterprise will cause some of you to shudder. Professor Drew Endy, one of the pioneers in this field, believes that part of that reaction stems from simple novelty. “A lot of people who were scaring folks in 1975 now have Nobel prizes.”<sup>11</sup> But even if inchoate, the concerns that synthetic biology arouses stem from more than novelty. There is a deep-seated fear that if we see the natural world of biology as merely another system that we can routinely engineer, we will have extended our technocratic methods into a realm that was only intermittently subject to them in a way that threatens both our structure of self-understanding and our ecosystem.

To this, the synthetic biologists respond that we are *already* engineering nature. In their view, planned, structured, and rationalized genetic engineering poses fewer dangers than poorly understood interventions to produce some specific result in comparative ignorance of the processes we are employing to do so. If the “code” is transparent, subject to review by a peer community, and based on known parts and structures, each identified by a standard genetic “barcode,” then the chance of detecting problems and solving them is higher. And while the dangers are real and not to be minimized, the potential benefits—the lives saved because the scarce antimalarial drug can now be manufactured by energetic *E. coli* or because a cheap test can demonstrate arsenic contamination in a village well—are not to be minimized either.

I first became aware of synthetic biology when a number of the scientists working on the Registry of Standard Biological Parts contacted me and my colleague Arti Rai. They did not use these exact words, but their question boiled down to “how does synthetic biology fare in intellectual property’s categories, and how can we keep the basics of the science open for all to use?” As you can tell from this book, I find intellectual property fascinating—lamentably so perhaps. Nevertheless, I was depressed by the idea that scientists would have to spend their valuable time trying to work out how to save their discipline from being messed up by the law. Surely it would be better to have them doing, well, science?

They have cause for concern. As I mentioned at the beginning of this chapter, synthetic biology shares characteristics of both software and biotechnology.

Remember the focus on reducing functions to black boxes. Synthetic biologists are looking for the biological equivalents of switches, valves, and inverters. The more abstractly these are described, the more they come to resemble simple algebraic expressions, replete with “if, then” statements and instructions that resolve to “if  $x$ , then  $y$ , if not  $x$ , then  $z$ .”

If this sounds reminiscent of the discussion of the Turing machine, it should. When the broad rules for software and business methods were enunciated by the federal courts, software was already a developed industry. Even though the rules would have allowed the equivalent of patenting the alphabet, the very maturity of the field minimized the disruption such patents could cause. Of course “prior art” was not always written down. Even when it was recorded, it was sometimes badly handled by the examiners and the courts, partly because they set a very undemanding standard for “ordinary expertise” in the art. Nevertheless, there was still a lot of prior experience and it rendered some of the more basic claims incredible. That is not true in the synthetic biology field.

Consider a recent article in *Nature*, “A universal RNAi-based logic evaluator that operates in mammalian cells.”<sup>12</sup> The scientists describe their task in terms that should be familiar. “A molecular automaton is an engineered molecular system coupled to a (bio)molecular environment by ‘flow of incoming messages and the actions of outgoing messages,’ where the incoming messages are processed by an ‘intermediate set of elements,’ that is, a computer.” The article goes on to describe some of the key elements of so-called “Boolean algebra”—“or,” “and,” “not,” and so on—implemented in living mammalian cells.

These inscriptions of Boolean algebra in cells and DNA sequences can be patented. The U.S. Department of Health and Human Services, for example, owns patent number 6,774,222:

This invention relates to novel molecular constructs that act as various logic elements, i.e., gates and flip-flops. . . . The basic functional unit of the construct comprises a nucleic acid having at least two protein binding sites that cannot be simultaneously occupied by their cognate binding protein. This basic unit can be assembled in any number of formats providing molecular constructs that act like traditional digital logic elements (flips-flops, gates, inverters, etc.).

My colleagues Arti Rai and Sapna Kumar have performed a patent search and found many more patents of similar breadth.<sup>13</sup>

What is the concern? After all, this is cutting-edge science. These seem like novel, nonobvious inventions with considerable utility. The concern is that

the change in the rules over patentable subject matter, coupled with the Patent and Trademark Office's handling of both software and biotechnology, will come together so that the patent is not over some particular biological circuit, but, rather, over Boolean algebra itself as implemented by any biotechnological means. It would be as if, right at the beginning of the computer age, we had issued patents over formal logic in software—not over a particular computer design, but over the idea of a computer or a binary circuit itself.

“By means of a computer” was the magic phrase that caused the walls around the public domain of algorithms and ideas to crumble. Will “by means of a biological circuit” do the same? And—to repeat the key point—unlike computer science, biotechnology is developing after the hypertrophy of our intellectual property system. We do not have the immune system provided by the established practices and norms, the “prior art,” even the community expectations that protected software from the worst effects of patents over the building blocks of science.

Following the example of software, the founders of the MIT Registry of Standard Biological Parts had the idea of protecting their discipline from overly expansive intellectual property claims by turning those rights against themselves. Free and open source software developers have created a “commons” using the copyright over the code to impose a license on their software, one that requires subsequent developers to keep the source open and to give improvements back to the software commons—a virtuous cycle. Could the Registry of Standard Biological Parts do the same thing? The software commons rests on a license. But, as I pointed out in the last section, the license depends on an underlying property right. It is because I have automatic copyright over my code that I can tell you “use it according to these terms or you will be violating my copyright.” Is there a copyright over the products of synthetic biology? To create one we would have to take the extension of copyright that was required to reach software and stretch it even further. Bill Gates might argue for intellectual property rights over software using the logic of his article in *Dr. Dobbs' Journal*. Will the argument for copyrights over synthetic biological coding be “I need the property right so I can create a commons”?

In practice, I think the answer is, and should be, no. Of course, one could think of this as just another type of coding, making expressive choices in a code of A's, C's, G's, and T's, just as a programmer does in Java or C++. Yet, software was already a stretch for copyright law. Synthetic biology strikes me as a subject matter that the courts, Congress, and the Copyright Office are unlikely to want to cram into copyright's already distorted outlines—

particularly given the obvious availability of patent rights. As a matter of conceptual intuition, I think they will see biological subject matter as harder to fit into the categories of original expressive writing. On one level, yes, it is all information, but, on another level, the idea of programming with gene sequences will probably raise hackles that the idea of coding inside a programming language never would. As a normative matter, I think it would be a poor choice to apply copyright to the products of synthetic biology. Attempting to produce a particular open commons, one might enable the kind of hundred-year monopolies over functional objects that the critics of software copyright initially feared.

If one wishes to keep the basic ideas and techniques of synthetic biology open for subsequent innovators, there are alternatives to the idea of a synthetic biology open source license. The Registry of Standard Biological Parts or the BioBricks Foundation can simply put all their work into the public domain immediately. (This, indeed, is what they are currently doing.) Such a scheme lacks one key feature of open source software: the right to force subsequent innovators to release their code back into the commons. Yet it would make subsequent patents on the material impossible, because it had already been published.

Regardless of the decisions made about the future of synthetic biology, I think its story—coupled to that of software and biotechnology more generally—presents us with an important lesson. I started the chapter with the metaphor of Procrustes's bed. But in the case of software and biotechnology, both the bed—the categories of copyright and patent—and its inhabitants—the new technologies—were stretched. Cracks formed in the boundaries that were supposed to prevent copyright from being applied to functional articles, to prevent patents extending to cover ideas, algorithms, and business methods.

Until this point, though the science would have been strange to Jefferson or his contemporaries, the underlying issue would have been familiar. The free-trade, Scottish Enlightenment thinkers of the eighteenth and nineteenth centuries would have scoffed at the idea that business methods or algorithms could be patented, let alone that one could patent the “or,” “if-then,” and “not” functions of Boolean algebra as implemented by a biological mechanism. The response, presumably, is to fine tune our patent standards—to patent the mousetrap and the corkscrew, not the notion of catching mice or opening bottles by mechanical means. Still less should we allow the patenting of algebra. These are fine points. Later scholarship has added formulae, data, and historical analysis to back up Jefferson's concerns, while never surpassing his

prose. As I said at the beginning of the book, if we were to print out the Jefferson Warning and slip it into the shirt pocket of every legislator and regulator, our policy would be remarkably improved.

But it is here that the story takes a new turn, something that neither Jefferson nor the philosophers of the Scottish Enlightenment had thought of, something that goes beyond their cautions not to confuse intellectual property with physical property, to keep its boundaries, scope, and term as small as possible while still encouraging the desired innovation.

Think of the reaction of the synthetic biologists at MIT. They feared that the basic building blocks of their new discipline could be locked up, slowing the progress of science and research by inserting intellectual property rights at the wrong point in the research cycle. To solve the problem they were led seriously to consider claiming copyright over the products of synthetic biology—to fight overly broad patent rights with a privately constructed copyright commons, to ride the process of legal expansion and turn it to their own ends. As I pointed out earlier, I think the tactic would not fare well in this particular case. But it is an example of a new move in the debate over intellectual property, a new tactic: the attempt to create a privately constructed commons where the public domain created by the state does not give you the freedom that you believe creativity needs in order to thrive. It is to that tactic, and the distributed creativity that it enables, that I will turn to now.