INTERNATIONAL COMPUTATIONAL LAW JOURNAL

CODEX SOURCE ARCHIVE

The Legacy of Hammurabi

Michael Genesereth

Computational Law Report · Release 3.0 · 2021-12-07

Reader edition prepared by The Research University under CC BY 4.0. This cover was added; the original PDF pages follow unchanged. Original authorship, publication identity and source links are preserved. This independent republication is not sponsored or endorsed by Stanford University, Stanford Law School, CodeX or MIT.

Volume 1 · Issue 5 · Evidence, Interpretation and Machine Judgment

Original source and license evidence are recorded on the article landing page.

MIT Computational Law Report •

The Legacy of Hammurabi

Michael Genesereth

Published on: Dec 07, 2021

License: Creative Commons Attribution 4.0 International License (CC-BY 4.0)

ABSTRACT

Computational Law is the branch of legal informatics concerned with the automation of legal reasoning. From a pragmatic point of view, it has special value in automated compliance management, i.e. the development and deployment of computer systems capable of assessing, facilitating, or enforcing compliance with rules and regulations. This article discusse technologies, challenges, and social implications related to this application. The article is a lightly edited transcript of the keynote presentation on Computational Law delivered at ReMeP 2021, Vienna, Austria.

1. Introduction

Around 1750 BC, The Babylonian king Hammurabi mandated that the laws of the land be encoded in written form (literally cast in stone) so that citizens could know what was expected of them and what would happen if they violated those expectations. In his own words, he wanted "to bring about the rule of righteousness in the land ... so that the strong should not harm the weak".

Hammurabi was not the first to do this. There are historical references to earlier legal codes in Ebla around 2400 BC, in Sumeria around 2300 BC; and there is the code of Ur-Nammu around 2100 BC (in which laws are arranged in the form of explicit *if-then* rules). However, the Code of Hammurabi is the most comprehensive and best known of these early efforts, and so he often gets the credit.



The upshot of these efforts in Ebla and Sumeria and Babylon is a tradition of *legal codification* that has lasted through millennia. One that has served us well. However, it

is not without its problems.

Today, we live in a complex regulatory environment. We are subject to governmental regulations from multiple jurisdictions. In the United States, there are federal regulations and state regulations and local regulations.

The sheer number and size of regulations can be daunting. We may all agree on a few general principles; but, at the same time, we can disagree on how those principles apply in specific situations. The Declaration of Independence is an important document in American history. It outlines the principles on which the country is based in just 1322 words. By contrast, the regulations on the sale of cabbages alone reputedly run to 26,911 words. That's not bad writing. The number and size of the regulations is essential to deal with specific issues and special cases.

Complicating the situation is the complexity of these regulations. Even small regulations can be very complex. Moreover, once regulations are created, complexity often increases as they are changed and then changed again. Here is an example in the context of insurance. A few years ago my house suffered some damage from flood water. Looking at page 32 of my insurance contract, I was pleased to see that water damage is covered. Unfortunately, when i called my insurance company, the adjustor pointed out the qualification on page 112 stating that the coverage on page 32 does not apply when various conditions exist "due to flood water". Policies likes this are typical in the insurance industry. And they are difficult for most people to understand without specialized legal knowledge or at least a substantial amount of study.

To make matters worse, regulations are not always well coordinated, arising, as they do, in different settings for different purposes. Sometimes, there are gaps, leaving important cases uncovered. More often, regulations overlap other regulations and in some instances are inconsistent with each other.

These problems make it difficult for affected individuals to find and comply with applicable regulations. The result is occasional lack of compliance, widespread inefficiency, and frequent disenchantment with the regulatory system.

This is a failure of our legal system. One of the functions of the law is to help individuals predict the consequences of their actions. If we do not know what the law is, the law does not serve this function; and, as many people have observed, the law today is far too complex for people to understand fully.

Lee Loevinger captured the irony of this situation in an article written in 1949: "It is one of the greatest anomalies of modern times that the law, which exists as a public guide to conduct, has become such a recondite mystery that is incomprehensible to the public and scarcely intelligible to its own votaries."

2. Computational Law

All is not lost. In my opinion, these problems are not insurmountable. As a computer scientist, I see these as information processing problems. As such, I believe that they can be mitigated by information technology. What is needed is appropriate "legal technology" - information technology applied to laws.

One step in this direction has already been taken. Today, the text of many legal documents is available online. In some cases, the information is adorned with "semantic" keywords to help in search. The good news is that these documents can be found using general search services, such as Google, or using services that specialize in legal information, e.g. those provided by companies like Westlaw and LexisNexis. Unfortunately, the quality of such search is limited. They often return too many documents, and sometimes they fail to find relevant documents. More importantly, there is no technology for interpreting these documents; human specialists must still be there to read the documents and apply them to individual cases.

An alternative, the subject of this article, is an extreme form of legal technology known as Computational Law. Computational Law is that branch of legal informatics concerned with the automation of legal reasoning. Not just legal search but legal analysis. Answers, not documents!

There are multiple opportunities for computational law in our current legal system - in adjudication and in regulatory analysis and synthesis. However, today, i want to talk primarily about a more immediate application, vz. compliance management, by which i mean the development and deployment of computer systems capable of assessing, facilitating, or enforcing compliance with rules and regulations.

Intuit's Turbotax is an oft-cited example of a compliance management system. Millions in the United States use it each year to prepare their tax returns. Based on values supplied by its user, it automatically computes the user's tax obligations and fills in the appropriate tax forms. Turbotax illustrates some of the key advantages of Computational Law for the individual. (1) It embodies tax regulations in computable form. (2) And it does its work in a *situated* fashion, not in the abstract but rather in the

context of the user's circumstances, thus teaching the user about relevant regulations in real world setting.

Of course, taxes are not the only application of Computational Law in compliance management. There are many other areas of the law that are amenable to similar treatment. Portico is a prototype of a system developed at Symbium for assisting architects and homeowners in formulating architectural designs that comply with planning codes and building codes. Analogous systems can be built other areas areas - e.g. management of child support, immigration aids, and building inspections.

Note that rules processed by Complaw systems are not restricted to governmental rules. They can equally well be the policies of universities and corporations (e.g. travel expense reimbursement, product configuration worksheets, and pricing rules). The program sheet shown here is an example. Students at Stanford use worksheets like this one to choose courses that fulfill all program requirements. (a) The worksheet shows all requirements explicitly - appearing in red until those requirements are satisfied, after which they turn blue. (b) It calculates units in accordance with departmental policies, ensuring no double counting. (c) It enforces that all prerequisite classes are taken. (d) It prevents students from taking mutually exclusive courses. And so forth.

There are also applications that do not involve governmental laws. The rules and regulations can just as well be the terms of contracts (e.g. insurance covenants, delivery schedules, real estate transactions, financial agreements).

Over the last 50 years, we have seen a significant amount of research on Computational Law, starting with the pioneering efforts of Thorne McCarty, Bob Kowalski, Marek Sergot, Edwina Rissland, and others.

A field that has flourished over the years with the help of conferences, lke ICAIL and JURIX and FutureLaw and so forth and work that has found a home at MIT and Stanford's CodeX and in the European CompuLaw project.

More practically, there are numerous companies capitalizing on the opportunities here. See the techindex for other companies in legal tech. There are 1350 at last count, including dozens doing work in complaw.

Complaw is even making its way onto the radar screens of some outside the community. Gartner recently released its report on the Hype Cycle for Emerging Technologies, which outlines technologies and trends that show promise in delivering

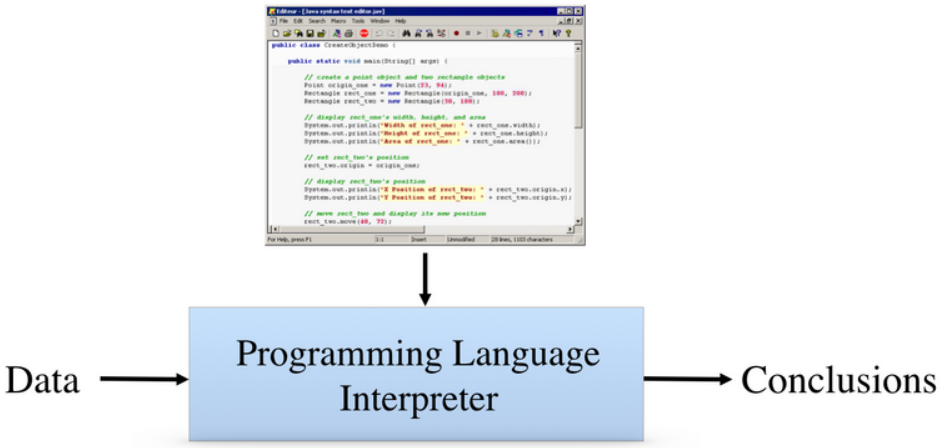
competitive advantage over the next 5-10 years. It mentions specific innovations, such as NFTs, Generative AI, and quantum machine-learning. And, for better or worse, machine-readable translation is on the list.

3. Technologies

So much for overview. Time now for some details. Let's start with a look at technologies. How do we build Complaw-enabled compliance management systems?

First of all, compliance management systems need data. The good news is that there is vast and daily increasing amount of data online in the form of tables in relational databases and in the form or so-called knowledge graphs. In addition there is the data supplied by users of interactive forms and design tools like the ones we saw earlier. The *question* is how to build systems that can analyze this data from a legal point of view. There are a variety of approaches we can take.

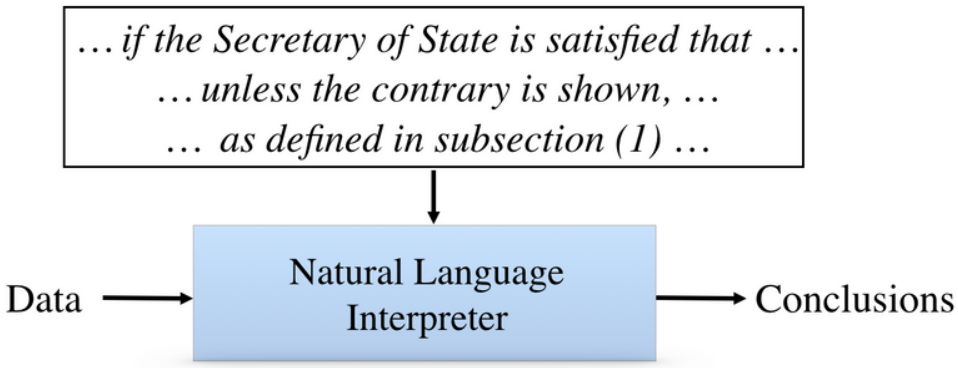
The obvious choice is **traditional programming**. In traditional programming, systems take the form of programs written in languages like C and Java and so forth. Such programs tell the system what to do but they do not represent the assumptions or goals. They include numerous programming details, such as variable declarations, that are no t essential to the laws being automated. And programs are typically task-specific - each is designed to accomplish a specific task.



There are substantial benefits to this approach. The resulting systems are usually quite efficient. There are lots of programmers capable of doing the work. And there are well-established practices for managing software engineering projects.

Unfortunately, there are also some disadvantages. Creating and maintaining programs is expensive and time-consuming. It is difficult to verify and/or analyze programs. Different programs are required for different tasks - one program for compliance checking, another for planning, and so forth. And there is no ways to generate explanations of results or to explain programs without writing separate programs.

Building smarter interpreters is better. There are optimists who think that we can avoid these problems by through the use of **natural language processing** to interpret texts of rules and regulations. The basic idea is good. Unfortunately, in my opinion, it is not practical for a couple of reasons.

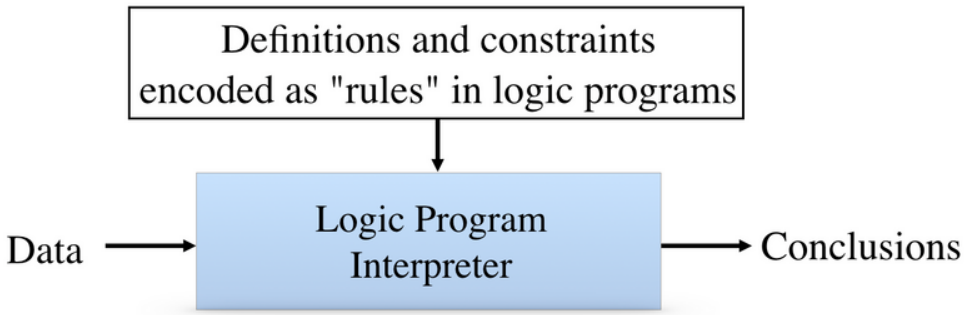


First of all, today's natural language processing technology is not very reliable. It is great for use by computer assistants in understanding simple command, like setting timers. Unfortunately, it is not so good at understanding more complex information. Here is an example shared by Henry Kautz, the director of the Computer Science office at the National Science Foundation. He was listening to Barber of Seville and reading Google's translation of the libretto. At one point, he noticed this curious translation shown in the subtitle: "Dunkin Donuts indeed". Now, I do not speak Italian, but I am pretty sure that that is not what Gioachino Rossini intended.

Unfortunately, the drawbacks of this approach are not entirely due to limits of our current technology. There is a deeper problem. The fact is that natural language is inherently ambiguous, as illustrated by this sentence: *There is a girl in the room with a telescope*. Is the girl in the room that has a telescope or is there a girl in the room and she is holding a telescope? There is no way to know for sure without significant context.

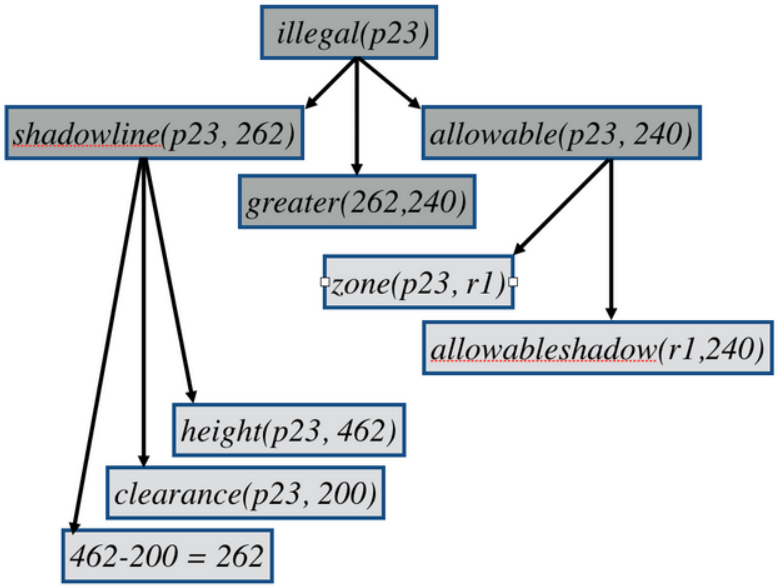
A third approach is **logic programming**. In this approach, we encode information in the language of logic. using this language, we can define new concepts; we can state

physical constraints; and we can encode rules and regulations. A grandparent is a parent of a parent. Parents are older than their children. And people may not be married to two different people at the same time. Like natural language, the language of Logic is expressive; yet, unlike natural language, it is grammatically simple and unambiguous. Moreover, we know how to build interpreters that can reliably derive conclusions from data and rules.



The good news is that we can use these interpreters for multiple purposes. For example, we can check compliance with rules and regulations. Here we have an example of some corporate definitions and rules. Officemates are people who share an office. Managers and subordinates may not be officemates. If John manages Ken and John is in 22 and Ken is also in 22, it is easy to detect the violation. And a logic program interpreter can also be used to plan for compliance. In this same situation, if John manages Ken, then teh system can suggest room assignments that do not violate the regulations.

But wait, there's more. Things do not necessarily end with the derivation of answers. We can also give users explanations. In deriving conclusions, LP systems implicitly or explicitly generate derivation trees for those conclusions; and it is possible to generate sensible explanations by incrementally exhibiting those derivation trees to the use. The example here comes from the Portico system we saw earlier.



Using this derivation tree, the system can explain why the building plan was rejected by citing the *actual shadow line* of the building and comparing it to the *allowable shadow line*.

p23 is illegal because
the shadow line is 262 cm and
the allowable shadow 240 cm and
262 cm is greater than 240 cm

If asked, the system can go one step farther and explain the actual shadow line and the allowable shadow line. There are subtleties in explanation, but this approach renders comprehensible and accurate explanations relatively easily.

A couple of points about encoding rules as logic programs. The computer programming community has long recognized the value of Interactive Development Environments (IDEs) to help in developing and maintaining programs. The use of LP makes possible the development of especially powerful IDEs for law.

These systems can save authors work providing pre-existing ontologies and knowledge bases and sample laws. They can also make available tools for the verification, analysis, and debugging of legal codes.

And they can provide technology for automatically translating to and from other formal languages. They can support languages that are more expressive that Logic Programming, e.g, FOL and beyond.

They can support languages that are more human-friendly. An especially intriguing idea here is to allow various forms of controlled natural language, such as Kowalski's Logical English, thus making possible pseudo-natural language authoring without the drawbacks.

4. Challenges

Okay, so much for the good news. Now for a look at some of the difficulties, the challenges in using logic programming or, for that matter, any other approach to building compliance management systems.

One technical problem, familiar to many individual with legal training, is open texture. Consider a municipal regulation stating "No vehicles in the park". On first blush this is fine, but it is really quite problematic. Just what constitutes a vehicle? Is a bicycle a vehicle? What about a skateboard? How about roller skates? What about a baby stroller? A horse? A repair vehicle? For that matter, what is the park? At what altitude does it end? If a helicopter hovers at 10 feet, is that a violation? What if it flies over at 100 feet? The problem is that the words used to define this rule do not have precisely defined meanings. Why not make the rule more precise by defining the notion of vehicle precisely?

There are good reasons that such problems arise. In some cases, rules are stretched by changes in technology, e.g. motorized skateboards in our example. More generally, we might expect to see problems with other laws due to technological changes like reusable rockets, autonomous cars, gene editing - developments that strain the rationale for the rules on the books.

In some cases, concepts are left ambiguous intentionally, in order to make it possible for politicians and regulators to compromise, effectively "kicking the can" down the road for other regulators and courts to deal with.

Even when the concepts are well defined, there are often exceptions to our rules. Consider a woman slightly exceeding the speed limit while driving her brother to catch a train. Or what about a husband driving his pregnant wife to the hospital? We might be willing to let minor infractions of this sort go. So why not make that part of the law rather than treating such cases as exceptions.

There are several reasons for this. There may simply be too many possibilities to consider. And enumerating the exceptions makes for a very complicated legal code. And, of course, there are lots of unanticipated situations.

Legislation as social programming. Of course, if law-making bodies and regulators did their jobs, there might be less need for legal debugging. A good program needs less debugging. Better laws might lead to less litigation. However, as i just suggested, there are good reasons for not trying to deal with all cases in advance.

Adjudication as legal debugging. The good news is that, in common law countries, like the US, we have mechanisms for dealing with such situations. We have judges and courts to hear specific cases, to deal with ambiguities and exceptions and to further refine laws and regulations accordingly. And we have appeals courts and in the US, a supreme court to ensure that they do it right. From a computer science point of view, the job of these judiciary bodies is to deal with ambiguities and exceptions. They are, in effect, debugging and, through the power of precedent, repairing deficiencies in the law.

By the way, providing technological help in these areas is another possible application for computational law. In the not too distant future, we might see automated or at least semi-automated adjudication - not just applying logical rules but also modifying or elaborating rules to handle specific cases.

And, someday. we may even be able to help rule makers by introducing automation into the regulatory process itself, analyzing propose rules and regulations, and maybe even recommending options that cover more cases from the outset and thereby eliminate the need for adjudication after the fact.

Of course, if we want to automated legislation and adjudication, there are factors that must be taken into account when formulating or revising revising rules and regulations that do not apply when simply following rules. For example, we need some idea of what it is that people want, what matters to them. And we need some sort of policy on equity. Are we simply optimizing overall societal utility or do we also care about balancing the utilities of the different members of society in equitable ways?

Also, adjudication and legislation need something more than logical reasoning / deduction to deal with cases where there are no unique answers. In the words of Edwina Rissland: "Law is not a matter of simply applying rules to facts via modus ponens", and when regarding the broad application of AI techniques to law, this is

certainly true. The rules that apply to a real-world situation, as well as even the facts themselves, may be open to interpretation, and many legal decisions are made through case-based reasoning, bypassing explicit reasoning about laws and statutes. As a computer scientist, I am intrigued by how we might formalize analogical reasoning in a way that produces acceptable answers in legislation and adjudication.

We do not at this point have adequate technologies for dealing with these matters. However, that does not mean that we cannot deploy compliance checking systems and reap the benefits. Just because a set of laws is not perfect does not mean that there is no value in informing citizens and organizations about the implications of the laws we have on the books.

There is an analogy here between laws and our theories of the physical world. Classical (Newtonian) Mechanics tells us about the results of applying forces to objects. We teach it to our kids. We use it ourselves.

Of course, classical mechanics is wrong. Nevertheless, we use it in 99.99% of our daily lives and for the other .01% we have specialized techniques. Despite the fact that it is not 100% correct, it is a useful approximation. In the case of the law, when we observe failures, we luckily have court systems to deal with those cases.

5. Implications

At this point , we do not have adequate technologies for dealing with these challenges. However, that does not mean that we cannot deploy deductive compliance management systems and reap the benefits. Just because a set of laws is not perfect does not mean that there is no value in informing citizens and organizations about the implications of the laws we have on the books.

(1) **Embedded Law**. The potential for deployment of such applications is substantial due to technological developments like the Internet, mobile systems (such as smart phones and smart watches), and the emergence of autonomous systems (such as self-driving cars and robots). All of which allow us to make automated compliance management tools available to *citizens* in their daily lives. Which brings me to a key point in my presentation today - the *cop in the backseat*.

Suppose that we had the benefit of a friendly policeman in the backseat of our car whenever we drove around (or perhaps an equivalent computer built into the dash panel of our car). The car can and should offer regulatory advice as we drive around - telling us speed limits, which roads are one-way, where U-turns are legal and illegal,

where and when we can park, and so forth. The Cop in the Backseat. But a friendly cop rather than a punitive one. Maybe we should instead consider the possibility of a “Lawyer in the Backseat" or a “Driving Instructor in the Backseat".

Capabilities like this already exist to limited extent in aviation, where displays like this one provide feedback on restricted areas and areas with special requirements (these concentric circles).



On the Internet when we are deciding whether to buy that drug from Canada or ship that alcohol to Virginia.

This makes lots of things possible. Suppose you are walking through the woods of Massachusetts and you see an attractive flower. You take a photo with your iPhone. Your *plant identification app* identifies it as a type of orchid and lets you know. At the same time, your *legal compliance app* tells you that, no, you may not pick it. Unless you cross the state line to a neighboring state.

(2) **Automated Enforcement**. A second implication is in area of automated enforcement. Technology makes it possible for us to enforce laws in ways that were not previously feasible. Automated reporting and billing is one potential of technology. Red Light cameras are examples. But there are also more interesting possibilities. Regimented systems, g.g. apple's photo policy.

Enforcement technology might someday invade our own devices. Suppose that the cop in the backseat were not a friendly cop but instead a cop with the power to ding us for violations of the law. In the case of a computerized policeman with an internet connection, we could imagine the policemen immediately reporting the violation to the DMV (Department of Motor to Vehicles). Insurance companies already make devices that track and report driving performance, allowing conservative drivers to benefit from lower rates and increasing the premiums for more aggressive drivers. Taking this one step further, we can imagine cars showing the results of such reporting to their drivers as well other performance factors. The cars could display not just actual speeds, but also speed limits, DMV fine balances, insurance premiums, and so forth. It would be interesting to see the effects of such reporting on drivers. Would they drive more conservatively when they see their bills mounting every time they exceed the speed limit?



It seems clear that there are positive features to possibilities like these. They promote safety while enhancing efficiency. At the same time, there are concerns, e.g. whether it is equitable to discriminate on the basis of personal characteristic, whether automatic DMV reporting compromises our right to privacy, and so forth. When I mention these possibilities, I often hear “No way, no how. It will never happen." Maybe so. The question is whether it should. Is it a good idea or a bad idea? If it is a good idea, how can we help to make it happen? And, if it is a bad idea, how can we prevent it from happening?

(3) **Technology-Enabled Law**. A third implication is what might be called *technology-enabled law*. We already have speed limits based on location, typically based on the lane of travel. Speed limits based on time of day are quite common.





What about speed limits based on the type of vehicle. Trucks are required to drive more slowly than passenger vehicles for reasons of safety. Perhaps Ferraris should be permitted to drive faster than Cadillacs because they are safer at higher speeds.





And what about speed limits based on personal characteristics, say age, or eyesight or blood pressure or other factors.



The Federal Aviation Administration already does this, disallowing individuals with poor eyesight from doing commercial flights. Vaccinations?

6. Conclusion

I began this article by citing the information problems posed by complex legal codes and by suggesting that Computational Law may offer a solution and, in particular, the application of computational law to compliance management. Although there are still challenges to automated compliance management, there is value in deploying this technology.

First of all, there is embedded law - building law into the devices we use to interact with each other and with the world. Complaw technology has the potential for *democratizing* the law. It takes law out of the courtroom and the law office and makes it available to the people it is designed to serve. It makes the law available to ordinary decision makers *at the point of decision*, when they are about to act or planning how to act. It can alert people to *obligations* and *restrictions*; it can help people *avoid mistakes of omission and commission*; and it can help them *get their due* from the government, from insurance companies; and so forth.

Second, this technology can decrease random enforcement, ensuring that laws are obeyed and eliminating the common frustration that result from being penalized for something when others go free.

And there is a broader implication as well. I cited complexity as one of the problems of our legal system. I believe that such complexity is inherent in translating general principals into practical form, going from standards to rules. The fact is that there is *virtue* in complexity. We need it to cover all of the cases without resorting to one-size-fits-all rules. But there is also harm in complexity. Complexity is the enemy of understanding. The use of computational tools allows us to reap the benefits of complexity without the harm. It allows us to make *better laws*.

Hammurabi and his predecessors began a lasting tradition. Over the centuries since those early legal codes were written down, it has been the norm to encode rules in written form and disseminate those rules - first via stone tablets, then via *books*, and more recently via the *internet*. However, just writing things down and making them available is not enough when the laws are voluminous and difficult to understand. By automating legal reasoning, Computational Law fixes this. It **facilitates compliance**; it **improves enforcement**; and it allows us to have **better laws**. In my opinion, it is the natural next step in the development of our legal system.

Michael Genesereth is a faculty member in the Computer Science Department at Stanford University. He received his Sc.B. in Physics from M.I.T. and his Ph.D. in Applied Mathematics from Harvard University. Prof. Genesereth is most known for his work on Computational Logic and applications of that work in Enterprise Computing, Computational Law, and General Game Playing. He is the current director of the Logic Group at Stanford and founder and research director of CodeX (The Stanford Center for Legal Informatics).

Header image generated with [Wombo]