TECHSHOW2021

Rules as Code: How Technology May change the Language in which Legislation is Written, and What it Might Mean for Lawyers of Tomorrow

WRITTEN BY:

Jason Morris

PRESENTERS:

Jason Morris: @RoundTableLaw

February 5, 2021

RULES AS CODE: HOW TECHNOLOGY MAY CHANGE THE LANGUAGE LEGISLATION IS WRITTEN, AND WHAT IT MIGHT MEAN FOR LAWYERS OF TOMORROW

What is "Rules as Code"?

The phrase "Rules as Code" is used to refer to a number of related ideas. What all of these ideas have in common is they involve the translation of legal rules, such as legislation, regulations, or contracts, into a language that computers can reason about.

To contrast, typing laws into a Microsoft Word document or storing them in a PDF is also translating them into a language that computers can use. The underlying digital representations are very useful for editing, printing, displaying, searching, and other capabilities that are familiar to us all. Rules as Code refers specifically to *the use of computer languages for representing rules that allow the computer to reason about the meaning of the rules*.

The "Small" Version of Rules as Code

Some people use the phrase "Rules as Code" in *only* this sense: the idea that it is useful to encode rules in computer languages in order to automate legal reasoning. And this is not a new concept, by any means. Organizations, public and private, of all sizes, have been using automated systems in order to comply with legal, regulatory, and internal policy rules for decades. Any time a website asks you to consent to the use of cookies in order to comply with the requirements of legislation like the EU's GDPR, that is an example of Rules as Code, in this minimal sense.

But the version of "Rules as Code" that I address today is a larger idea, that has larger implications for the legal profession of the future. For the rest of this paper, I will use the phrase "Rules as Code" to refer to this larger idea.

The "Big" Version of Rules as Code

The larger conception of Rules as Code proceeds from the acknowledgement that reasoning about rules is increasingly going to be automated. Whether your concern is automating legal services as a means of improving access to justice, or delivering public services online in a cost-effective and legally compliant way, or updating your business' systems when governments and regulators are constantly adding and changing rules, or taking advantage of smart contracts that are executed on the blockchain, everyone is facing a world in which rules eventually become code. But the process is not pretty.

Story: What is a "Week"?

Imagine a computer programmer who is attempting to get a website to follow a particular regulation. In the course of writing the code, they come across a section of the regulation that uses an undefined term. Let's say the term is "week." The programmer knows what a "week" is, intuitively. But the computer does not. And when the programmer needs to explain the concept of a "week" to the computer, the programmer discovers that there are a number of possible definitions, each of which will result in different benefits being due at different times.

Does a week begin on January 1, and a new week start every 7 days? Does a week begin on Sunday? Does it begin on Monday? If a week starts on January 28, what fiscal year is it a part of?

If the programmer is being careful, they may ask for a legal interpretation of what "week" means in that context. Perhaps there is an interpretation act that covers it. But if not, and the issue has not been litigated, there may be no real answer to be had. You might, at that point, seek guidance from the body that issued the rules. But not all regulators provide interpretive guidance, and none of them do it fast.

So most likely, the programmer just guesses, documents their guess in the code, and hopes that it never costs anyone any money.[1] Then the application is sent for quality assurance, and the organization's legal department is asked to determine whether or not the application complies with the relevant regulation. The legal department is given access to the code, but the code is not useful to them. They cannot read the programming language the software is written in, and even if they could, the vast majority of the code is not written to comply with legal obligations. The parts that are cannot be distinguished from the parts that are trying to do everything else, so even a lawyer who reads the programming language would be at a loss trying to figure out what the code says about the law.

Instead, the testers just use the application, repeatedly, providing it with different sorts of inputs and checking to see if the outputs match their expectations. But that doesn't tell you that the software is bug-free, legally speaking. It just tells you that you haven't found any legal bugs, yet.

And if you do get an answer from the software that you are not expecting, perhaps because the programmer guessed wrong with regard to the meaning of "week", the software cannot explain how it came to that conclusion. So, the testers cannot say "you interpreted 'week' wrong, it works this way." The testers can only send it back to the programmer with what they expected the result to be, and have the programmer debug it.[2]

This process is hard, halting, expensive, and then it is duplicated inside each regulated entity.[3] For many high-risk purposes, it is entirely inadequate. You might build a parking ticket app this way, or something that would be worth the investment by virtue of how often it was used, like a tax preparation tool. But there is a wide range of automated legal services for which this process is just too risky and expensive.

---

[1] This example has precedent. An inadequate definition of "week of work" in New Zealand's Holidays Act, incorrectly implemented in various payroll systems, resulted in tens of millions of dollars of retroactive holiday pay being owed by New Zealand employers. See https://www.nzherald.co.nz/business/holidays-act-nightmare-34000-workers-paid-out-189m-so-far/D3CGK7TKAJQKMQX5WGU5ZX5LII/.

[2] For a more detailed investigation of the problems with ensuring legal accuracy in automated legal systems, see Marc Lauritsen and Quinten Steenhuis, "Substantive Legal Software Quality: A Gathering Storm?" In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law* (ICAIL '19), 2019. DOI:https://doi.org/10.1145/3322640.3326706.

[3] Banking associations in the United Kingdom and Australia have been asking for financial regulators to adopt a Rules as Code approach in order to simplify the process of updating their automated compliance systems. See https://www.afr.com/companies/financial-services/laws-should-be-published-in-code-so-computers-can-read-them-csiro-20200115-p53rlu, and Micheler, E., Whaley, A. Regulatory Technology: Replacing Law with Computer Code. Eur Bus Org Law Rev 21, 349–377 (2020). https://doi.org/10.1007/s40804-019-00151-1.

The programmer is left thinking to themselves, "I would never have allowed them to write this rule this way. I would have told them that 'week' was going to be a problem to automate."

Computers are a Primary Audience of Rules

The Rules as Code movement ultimately looks at this story above and agrees with the programmer. Had they been in the room when the drafting of that regulation was taking place, they could have encoded the intended interpretation as the drafting proceeded. The problem with the definition of "week" would have been caught early, and easily fixed. And the programmer's encoding of the regulation could then have been provided to all the regulated entities, who could have used it to vastly simplify the process of automating their own systems.

Rules are an effort to communicate, and communication works better if you are conscious of the audience. Computers are increasingly the audience, and programmers are people who understand how to communicate with them. As such, **Rules as Code proponents argue that rules should be drafted in code, and in natural languages, at the same time**.

The Benefits of Rules as Code

Rules as Code, therefore, is an interdisciplinary approach to drafting legal rules.[4] It calls for the people who automate the rules to be involved in their drafting early. The potential benefits for people who are responsible for automating reasoning about rules are relatively clear. But surprisingly, experiments have shown that the benefits of this interdisciplinary approach are not limited to the programmers. They extend also to the drafters, and to their client[5]. Let's take a look at both categories, starting with drafting.

How Rules as Code Improves the Quality of Legal Drafting

The unvarying conclusion of the legislative drafters involved in Rules as Code experiments held in Canada and elsewhere is that had the Rules as Code approach been taken when the legislation was originally drafted, the result would have been better-drafted legislation.[6] The discipline involved in encoding legislation so as to explain it to computers is similar to, but stricter than, the discipline involved in modern legislative drafting best practices.

This benefit, unintuitively, does *not* come from "running the code." It comes primarily from merely being forced to think about how the code *might* be written. So this benefit of Rules as Code can be achieved with a person who knows a programming language, but does not actually possess a computer.

---

[4] The conversation in this paper presumes an ongoing divide between the legal expert and the programmer. Requiring two different experts is a source of expense and friction, and unlikely to be sustainable in the long term. The solution to that problem is for the legal and programming skillsets to be in the same person. But the rules as code process will nevertheless remain "interdisciplinary," those disciplines will both be practiced by the same person.

[5] "Client" here can have a lot of meanings. If the rules in question are a contract, the client may be the parties. If the rules are a regulation, the client is the regulator. If the rules are a law, the client may be a government. The client is the entity for whose purposes the rules are being drafted.

[6] A summary of the Rules as Code experiment performed by the Canada School of Public Service is available at Scott McNaughton's blog: https://medium.com/@mcnaughton.sa/week-50-reflections-on-rules-as-code-5878ff42d43c.

Automated Testing

In addition to the benefits that arise merely from the effort to encode the rules, the encoded rules themselves provide a tool that can be used to push the benefits even further.

Once rules are encoded, they can be tested, and that testing can be done automatically. Computer systems can generate random fact scenarios, enter those fact scenarios into the rules, and calculate the consequences. Those consequences can then be compared to expected outcomes.

For example, a proposed amendment to a piece of tax legislation might be intended not to increase anyone's taxes. You can specify a fact scenario, and the taxes owing that you expect before and after the change, and determine whether or not it has the expected effect. But you can also state generally that no outcome should result in increased taxes, and have the computer randomly generate any number of fact scenarios and test them to see if that condition is ever violated.

This task of coming up with situations in which the rule does not behave as expected is a major part of the expertise of legal drafting. Legal professionals use their imagination and their experience to think of scenarios, and then mentally check whether those scenarios will behave as expected. Rules as Code gives us the ability to get the help of computers with that process, supercharging that human ability.

Not only does this improve the ability of the legislative drafters to take their intended meaning and put it into words, it also, importantly, improves the ability of the clients (in the legislative case, usually a government department) to see whether or not their requested law actually does the thing they wanted it to do. In the public sector, you know better what law you want, and you know better whether or not that is the law you got. In the contract space, you can test various contracts to see which best protects you, and then also be more certain that you wrote the contract you wanted.

In this way, the benefit of testing accrues not only to the drafters, but to their clients. You can "try on" a contract or law to see if it fits, and if not, write a different one.

With shared data structures, it will also be possible to determine whether the new law or contract has any unanticipated effects on other related laws or contracts. It is also trivial to create visualizations that will illustrate the parts of this and other laws that will be affected by the proposed change, to alert drafters to possible unanticipated consequences.

Formal Verification

This automated testing approach can also be augmented with formal verification techniques. Formal verification tests the rules not by providing inputs and reviewing outputs, but by logical implication of the rules themselves. Tools such as model checkers and satisfiability module theories (SMT) tools can be used to make categorical statements about what is possible and impossible under the rules as written.

The sorts of questions that might be asked of a contract using formal verification include "is it true that no matter what the parties do, the contract will terminate with either the purchaser in possession of the good and the seller having received payment, or with the seller still in possession of the good, and payment having been reversed."

Providing a *mathematical proof* a given contract will terminate in one of two desired ways is a level of legal quality assurance that is, to my knowledge, unprecedented in the legal services profession, and impossible in the absence of encodings of legal rules. That is just one example among many of how Rules as Code could potentially bring into existence legal services that did not previously exist.

Natural Language Generation 

If the laws are drafted in technologies that are capable of generating controlled natural language versions of the encoded rules, then the encoding can be presented in something approaching English, or whatever other language the drafters are using. This allows non-programmers to review exactly what the code says, without having to review the code itself. In the drafting room, the programmer can encode the legislation, and then have the encoding generate an English-language version of itself. That can be read by the non-programmers in the room, who can then verify for themselves that the encoding means the same thing that the natural language version of the law does.

Again, all of these benefits accrue to the drafting process, even if the law, regulation, or contract is never automated. **Rules as Code is therefore not merely a way to make it easier to automate legal services. It is independently a methodology for high-quality legal drafting.**

How Rules as Code Benefits Implementation

The first and most obvious way in which the Rules as Code approach benefits implementation is that all of the benefits of drafting accrue to the person implementing the software. The rule is less likely to be under-specified, or needlessly vague. There is also more likely to be a central and authoritative version of the encoding, which eliminates the need to write your own.

There are also a wide array of other possible benefits, all of which depend on the specific technology used for encoding the rules.

Explanations

If the right technology is used to do the encodings, it is possible to generate explanations for the answers that are obtained from the encoded rules. An application developer gets access to answers and explanations with natural language versions and links to source rules, for "free."[7]

Expert Systems

If the language used to write the code supports backward chaining over logical rules, **it will be possible to generate a legal services application merely by posing a question**. The encoded rules will be given to a reasoner, which will determine what information it needs in order to be able to answer that question, and pose those questions to the user using controlled natural language expressions. It can then use the information provided to generate an answer to the question, display the answer to the user, and explain the answer to the user with links to source rules. With the right technology used to encode the rules, that

[7] DataLex and Neota Logic described below have this capability.

entire process can be automated. The resulting interface will not be very polished, but the task of rapidly prototyping an application that uses the encoded rules in a new way will be almost zero.[8]

Planning Applications

Again, depending on the technology used, it may also be possible to develop planning applications on the basis of encoded rules. While an expert system knows the rules, asks the user for the facts, and generates the outcome that the user was asking about, like "am I in compliance?" A planning system takes the specific outcome the user wants "I want to comply", collects information about the facts, and uses the rules to determine what the user would need to do to make that outcome occur.[9]

Ease of Maintenance

One of the biggest problems in automating legal services right now is maintainability. It is possible to write code that automates a legal service, but that code is frequently very difficult to maintain when the rules change. Rules as Code provides two ways to improve that situation. One is that if the rules are encoded centrally, the work involved in updating rules is minimized. The other is that because the rules are encoded separately from the rest of the application, only changes to the "interface" of the rules, and not changes to the rules themselves, will necessitate any changes in the applications that use them.

For example, if the percentage of tax owing changes, merely changing the encoded tax rules will completely update the application. Under existing methods, all changes to the rules may cause multiple changes is disparate parts of the application code.

In addition to those benefits, if defeasible logic programming systems are used, you can get a nearly one-to-one relationship between parts of rules and parts of code. When rules change, only the corresponding piece of the encoding needs to change, as opposed to needing to search through all your code for any parts that are affected by the rule change. This can be a major benefit for maintainability with rules that change frequently.

The Current Rules as Code Landscape

The Rules as Code conversation was recently renewed by experiments in the public sector in New Zealand and Australia.[10] As of now, there are no jurisdictions that have used the Rules as Code approach with regard to enacted legislation. Experiments continue in various jurisdictions.[11]

Who is Doing Rules as Code?

France is perhaps the jurisdiction that has traveled furthest down the Rules as Code path. Statutes in France are obliged to undergo impact analysis. The development of OpenFisca and its use in the LexImpact

[8] Datalex, Neota Logic, and Docassemble described below have this or similar capabilities.
[9] Logic Production System described below has this capability.
[10] See https://www.digital.govt.nz/dmsdocument/95-better-rules-for-government-discovery-report
[11] An excellent summary of the current state of Rules as Code adoption and technology is available from the OECD's Observatory for Public Sector Innovation, which published "Cracking the Code: Rulemaking for Humans and Machines" in 2020, available at https://oecd-opsi.org/projects/rulesascode/.

tool for analyzing the impact of amendments to French tax law are the most direct real-world example of the use of Rule as Code today.

Canada's federal government is currently experimenting with Rules as Code[12], and work on Rules as Code continues in Australia, where it has been touted by the Prime Minister as how legislation may be drafted in the future.[13]

In the United States, similar technologies are used to generate expert systems with regard to government programs. For example, it is reported that the ObamaCare websites were generated using a tool called Oracle Intelligent Advisor, which uses a controlled natural language policy language and automatically generates expert systems from that code. I am aware of no conversations in the United States with regard to the implementation of Rules as Code at drafting time.[14] Stanford University's CodeX Center for Computatonal Law is also actively researching the area, and has spawned at least one startup company, Symbium, that is using encoded municipal land use laws in order to generate services to advise citizens what they are permitted to do with their property. But while these projects use the same underlying technology, they are not Rules as Code in the larger sense, because the encoding is being done after the relevant law has been enacted.

What Tools Exist for Rules as Code?

The tools available today for Rules as Code can be usefully divided between applications and programming languages or libraries. Almost any programming language can be usefully used for Rules as Code, but there are a few that have been created specifically for legal purposes, and in others there are libraries that have been created to solve all or part of the problems faced in encoding legislation.

Some of these tools will be demonstrated during the conference presentation.

These lists are not intended to be exhaustive.

Applications for Rules as Code

Oracle Intelligent Advisor - https://www.oracle.com/cx/service/intelligent-advisor/

> Oracle Intelligent Advisor is not currently marketed for contract or statutory purposes, but for use for encoding internal policy for the automation of customer service in large enterprises. But the technology can easily be used in legislation and contract. OIA has the ability to generate web-based expert systems and document automation on the basis of encoded rules. The encoding language is a controlled natural language integrated with Microsoft Word and Excel. It

[12] Videos describing Canada's Rules as Code experiments are available at https://www.csps-efpc.gc.ca/video/rules-as-code-1-eng.aspx and https://www.csps-efpc.gc.ca/video/rules-as-code-2-eng.aspx.
[13] See https://www.zdnet.com/article/morrison-floats-legislation-written-in-code/
[14] There is some suggestion that Westminster-style parliamentary democracies are betted suited to Rules as Code at the level of legislation. That, however, does not account for the lack of apparent interest in Rules as Code from the United States with regard to delegated legislation and regulation.

is commercial software, and Oracle's licensing terms make it prohibitively expensive to deploy except in enterprise environments.

Neota Logic - https://www.neotalogic.com/

Neota Logic is a commercial tool for building expert systems and document automation on the basis of a logical encoding of rules created using a flowchart-like interface. It is marketed for legal purposes, and used in several law schools around the world to teach legal service automation. Neota Logic is paid software on a subscription model.

Blawx[15] – https://www.blawx.com

Blawx is an open source tool for Rules as Code online. It uses the Google Blockly library to create a drag-and-drop puzzle-piece visual programming environment. This visual code is then transalted to the Flora-2 programming language, which is an open source declarative logic programming language featuring defeasibility, constraint programming, and higher order logic. Blawx is free to use for commercial and non-commercial applications.

DataLex – https://www.datalex.org

DataLex is a tool offered by AustLII which allows you to express legislative provisions in a controlled natural language, and automatically generates an online chatbot interface that can answer questions about those pieces of legislation, explain those answers, and provide links to source materials. It is closed source, but free to use for non-commercial purposes. There are no commercial licenses available.

Clause.io - https://clause.io/

Clausie.io is a commercial application for generating smart contracts (which it calls "smart agreements"). It uses an interface that allows you to insert "clauses" into your smart contract which have both a natural language version and a code version. These standard clauses can be modified as required.

Docassemble – https://docassemble.org/

Docassemble is a very popular application for generating user interviews and document assembly applications, particularly in the pro bono space. Docassemble uses the logical structure of the target document template as a means of determining what questions to ask, when to ask them, and in what order, meaning it behaves much like an expert system. Rules encoded in Python can also benefit from this backward chaining method, allowing for an expert-system-like ability to generate an interview on the basis of encoded rules. It is maintained by Jonathan Pyle, a lawyer working in the pro-bono sector, and has features designed to simplify

[15] For full disclosure of personal interests, Blawx is developed and maintained by the author.

the creation of interviews and document automations involving court filings. It is open source and free to use or modify for any purpose.

Programming Languages and Libraries for Rules as Code

Catala – https://www.catala-lang.org

Catala is a domain-specific programming language under development by the French research institute Inria for encoding legislative texts. It strives to provide a one-to-one relationship between pieces of legislative text and pieces of Catala code, and to allow that Catala code to then be translated into other programming langauges for implementation in various places. It features "default logic", which allows for the encoding of exceptions to stated rules.

L4[16] – https://github.com/smucclaw/dsl

L4 is a domain-specific programming language for law under development at the Singapore Management University Centre for Computational Law, which is a joint venture before the Singapore Management University School of Law and the private company Legalese. It has a similar design intent as Catala, but aspires to create a more user-friendly syntax usable in legislation and contract, and to generate code that is capable of advanced formal verification, multi-lingual natural language generation, and expert systems.

OpenFisca – https://openfisca.org

OpenFisca is an open source library for the Python programming language, designed for use in encoding and analyzing statutes and amendments dealing with financial benefits. It is used by the French National Assembly's LexImpact tool, for analyzing proposed changes to French tax law.

Ergo – https://accordproject.org/projects/ergo/

Ergo is an open source programming language for encoding legal contracts under development by the Accord Project. It features type-safety, and is developed with the intent of being able to do formal verification. The Accord Project also features a templating language Cicero, and a data structure specification language, Concerto. Clause.io is a partner in the Accord Project, and uses these technologies in its offerings.

Logic Production System (LPS) - http://lps.doc.ic.ac.uk/

LPS is a programming tool developed by Imperial College of London and commercialized by Logical Contracts. Almost uniquely, LPS is designed to be able to both predict an outcome based on facts, *and* to be able to generate a plan of action in order to achieve a desired outcome.

[16] For full disclosure of personal interests, the author is employed by the Singapore Management University Centre for Computational Law.

Research into generating LPS code from an easy-to-read controlled natural language called "Logical English" is ongoing.

What Does Rules as Code Mean for Lawyers of the Future?

If the Rules as Code experiments that are underway now are successful, which is to say that they demonstrate benefits to the quality of drafting and the automation of public rules that are more than proportional costs in speed and expense of the interdisciplinary drafting approach, there are a number of things that we can expect. Specific guesses are unlikely to be accurate, but here are a few possibilities:

- The number and quality and ease of use of Rules as Code technologies will increase. With sufficient advancement in the technology, the basic use of these technologies will be a part of first-year law school curriculum. These tools will include the ability to automatically generate an interview-style user interface in order to collect information that might be relevant in determining a legal question posed in code, effectively reducing the task of building a new automated legal services tool to the task of posing a question.

- Legal research providers will begin encoding their statutory resource materials, and offering access to those encodings. That service will become less and less valuable as governments begin to offer their own encodings publicly.

- The use of memos as the primary means of recording legal information in enterprise knowledge management systems will be supplemented with the use of encodings. Those encodings will be used in conjunction with data about past services provided to determine whether any client was provided with advice that would now differ, based on a new interpretation of the rules.

- There will be a growth in the use of ontologies that provide basic concepts that can be re-used across different rulesets. Ideas such as "legal person" and "agent" will not be coded from scratch but imported from a library. Interpretation Acts will begin to be used to hold these consistently used definitions.

- Legal service providers will begin providing services that do not now exist, such as the formal verification and automated testing of contract and legislative terms, providing high-risk scenarios with greater confidence that known types of loopholes have been avoided. These services will be provided first in the smart contract space, but slowly expand to other areas of legal drafting.

- Firms which now provide updates in areas of law as a subscription service will instead maintain an updated encoding of that area of law and provide access to clients who wish to answer questions using that encoded knowledge, which will also be used as a source of client development.

- Governments will slowly transition to publishing encodings of statues as they can be generated. Large amendments will not be allowed without an encoding of the existing legislation. Because impact assessments will be vastly simplified, they will become mandatory in many areas.

- A far wider range of legal services will be automated, and the access to justice gap will become smaller.

These are clearly optimistic projections, and speculative. If they come to pass, they will undoubtedly come to pass more slowly than I would prefer. Susskind is still waiting for the rise of "Legal Engineers."

What can be predicted with more evidence is that the legal profession will undergo changes that are similar to the changes that occurred in the profession of accountancy after the advent of electronic spreadsheets. In broad strokes, demand for accounting professionals increased because they were capable of doing more, more quickly, and more accurately.

But the triggering event in accounting – the invention of electronic spreadsheets – does not yet have a parallel in law. There is no tool for automating legal logic that provides as low a barrier to entry and as high a return on investment in learning. Research continues, and that tool will certainly arrive. When it does, basic competence with it will be a requirement of legal regulators within 10 years.[17]

How do you Encode Subjective Legal Issues?

When introduced to the concept of Rules as Code, the reaction of many legal professionals is to ask about the things that they suspect cannot be encoded. How, for example, might you encode the meaning of "the couple are in a committed relationship?"

The answer that question is actually quite simple. Either the couple are in a committed relationship, or they are not in a committed relationship. So that would be encoded as a Boolean "yes or no" variable with regard to that couple.

Which only reveals that we are asking the wrong question. The right question is "how do we encode the process by which a judge would determine whether or not a couple are in a committed relationship?"

The answer to that question is more complicated. Is the process for coming to that conclusion set out in the law? Then encode whatever the law sets out. If the process is not set out in the law, you would no longer be doing Rules as Code, you would be doing Jurisprudence as Code[18].

Which only tells us that we are still asking the wrong question. The right question is "what do we do about the parts of the law that we can model using a Rules as Code approach, but can't calculate?"

The first thing to notice in answering this question is that there are two different sets of benefits from Rules as Code: implementation benefits, and drafting benefits. Not being able to calculate how certain conclusions are reached does not reduce the drafting benefits. Even if there are legal conclusions that cannot be calculated, they can be used to calculate other conclusions, and all of the benefits of encoding the legislation are still obtained with regard to those other conclusions. By encoding "committed

[17] Approximately ten years is how long it took electronic spreadsheets to become a basic competence of the accounting profession. In all likelihood, the timeline in law would actually be much shorter, because electronic spreadsheets were invented before most people owned computers.

[18] There are other rule-based artificial intelligence approaches specifically for solving this sort of problem, see e.g. https://github.com/Gauntlet173/docassemble-openlcbr.

relationship" as a Boolean, we can test whether it is having the desired effect on another conclusion, like "obliged to pay spousal support."

Subjective legal issues pose no problem at all to obtaining the drafting benefits of a Rules as Code approach.

If you are trying to build an automated legal service, subjective legal issues that cannot be calculated on the basis of data known to the user can be dealt with in a number of ways. As just three examples, you can:

- Provide conclusions that are contingent on the subjective legal issue, with reasonable informed consent.
- Refer the user to a human professional to get an opinion with regard to the subjective legal issue, and let the system pick up again where the human professional leaves off.
- Use the fact that the issue is relevant as a way to triage people away from the automated legal service.

There are other approaches, too. But the important point from a Rules as Code perspective is that the existence of subjective issues in a rule does not preclude the rest of the rule from being usefully encoded.

As a result, **there are no categories of law, regulation, or contract that cannot be beneficially encoded, at least in part**.

Where is Rules as Code Best Used?

Just because any rule can be encoded to some degree does not mean that all rules will benefit equally from encoding. Some factors that make the application Rules as Code more valuable include:

- Rules about which impact assessments are mandated
- Rules with high risk to liberty, or wealth
- Rules that will otherwise be encoded inconsistently by a large number of regulated entities
- Rules that will be used in decision-making frequently
- Rules that change frequently
- Rules that are used without the benefit of legal assistance
- Rules that are highly deterministic
- Rules that are long, or complicated

The government of France, for example, has mandated impact assessments with regard to a wide range of social benefit laws, which has motivated the Rules as Code movement in that country. Tax law is likewise also an area for which most of these factors indicate that Rules as Code would be appropriate. But Rules as Code may be an appropriate approach to take with regard to family law statutes that are long, complicated, used frequently by people without legal help, and have high impact on wealth and liberty interests.

Rules as Code is Not Artificial Intelligence

There is a saying that the definition of "artificial intelligence" is whatever you are still surprised a computer can do. Artificial Intelligence can be broadly divided into "waves". First-wave artificial intelligence, which came into existence in the 1970s and grew in popularity through the 1980s, was "symbolic" artificial intelligence. It was also known as "rule-based" artificial intelligence. Essentially, the idea was to give computers the ability to reason using deductive logic, and then model your real-world problem in a logical language. "Second-wave" artificial intelligence is more recent, and what people currently think of as "AI". It involves neural networks, machine learning, and similar technologies. It is also known as "data-based" AI.

Symbolic, "first-wave", or "old-school" artificial intelligence is seldom referred to as AI, today, except in computer science academic circles (and my current job title). But those technologies remain some of the best available for the task of encoding legal rules. Indeed, one of the first declarative logic programming languages ever created, Prolog, was almost immediately put to work encoding British legislation.[19] That approach to the digital representation of legal knowledge is the work on which legal futurist Richard Susskind wrote his PhD dissertation in the 1980s.[20] These "rule-based" approaches are also the approach taken by the SMU Centre for Computational Law, and the ones implemented in Blawx.com.

It is also possible to imagine, at some point in the future, that "data-based", "second-wave", "new-school" artificial intelligence techniques could be used to simplify the Rules as Code encoding process, perhaps even by doing simple translations from natural language rules to encoded rules. But these techniques learn from examples, and as of now there are inadequate examples of one-to-one translations between rules and code to be able to train artificial intelligence to do the translation. Put another way, we do not yet have the "rosetta stone" that would allow machine learning to learn to write rules as code. Also, the translation from natural language rules to encoded rules inevitably involves making explicit a number of things that are only implicit in the natural language versions. It is not clear whether current machine-learning approaches would be any good at that task.

So you can certainly use old-school artificial intelligence to do Rules as Code. And at some point in the future we might be able to use new-school artificial intelligence to improve the process.

Fundamentally, though, the Rules as Code proposal is one in which the human tells the computer what the law says, and not the other way around.

---

[19] M. J. Sergot, F. Sadri, R. A. Kowalski, F. Kriwaczek, P. Hammond, and H. T. Cory. 1986. The British Nationality Act as a logic program. Commun. ACM 29, 5 (May 1986), 370–386. DOI:https://doi.org/10.1145/5689.5920

[20] Richard Susskind, *Expert Systems in Law*, Oxford University Press (1987).

Legal, Ethical, and Regulatory Implications of Rules as Code

Rules as Code creates a challenge for a number of common law concepts. The requirement that the law be accessible to people has generally been considered to be satisfied if the text of the actual law was publicly accessible. Arguably, if at some point in the future encoded versions of legislation become authoritative, merely providing the public with access to the source code will be inadequate to serve the interests protected by the principle of access to law.

The doctrines surrounding the interpretation of contracts, in particular the admissibility of information not contained in four corners of the contract itself, will almost certainly need to be revised in circumstances where the parties encoded their intention, with the opportunity to test that encoding before entering into the agreement, and where they subsequently disagree about whether the encoding is consistent with their intent. The admissibility of the source code of a smart contract for the purpose of determining the intent of the parties also raises problems of whether there are judges who can read that code, and experts enough to advise them if they cannot. The first generation of smart contract litigation will occur in courtrooms where the encoded contracts cannot be read and interpreted by the judges

The ethical implications of Rules as Code are not yet known, as it is not yet widely utilized. It certainly involves a risk that greater attention will be paid to those parts of legislation that can be automated in implementation, as opposed to merely in drafting, but it remains to be seen how much that risk materializes.

With regard to implementation, the ethical risks of Rules as Code are likely not significantly different from those of any other attempt to automate legal services. Fundamentally, the question is whether the risk being adopted by the user of the automated system has been minimized, the remaining risk is reasonable, the user is informed, and the user consents to adopt that risk.

Rules as Code, because it simplifies the process of doing quality assurance on the legal aspect of automated systems, should allow us to reduce the risk associated with the automation of legal services. That ought to make it possible for more services to be ethically automated, and should therefore be beneficial to the use of automated legal services in access to justice applications.

Because Rules as Code is not data-based artificial intelligence, the widely known risks of bias in those systems do not apply to Rules as Code. Systemic bias can still be crystallized in encoded rules, but not invisibly or without human input.

Who should be responsible for determining that a legal encoding is accurate, how they should do so, and whether there is some regulatory body that can certify them as capable of doing so, are questions that legal regulators will need to grapple with in the future.

No one lawyer is ethically obliged to serve everyone. Collectively, however, the profession is ethically obliged to ensure that the way it operates is in service of the public's best interests. That ethical duty is enhanced with a moral obligation arising from the operation of a self-governing regulatory monopoly.

The regulation of the automation of legal services, whether using Rules as Code or other approaches, should therefore weigh the risks of imperfect automation against the risks of no help whatsoever, which

is too often the choice people actually face. **Both competence and the provision of efficient services are ethical duties, and the regulation of automated legal services should reflect a balance between them.**