Research Note

Rules as Code

By **Matthew Waddington**, Senior Legislative Drafter at the Legislative Drafting Office of Jersey, **Orcid:** <https://orcid.org/0000-0002-2396-5417>

Legislative Drafting Office, Jersey

ABSTRACT

“Rules as Code” is a label adopted by people in governments working on the idea of encoding (or just marking up) legislation while it is being drafted, so that the logic of the resulting legislation can be “read” (and checked) by a computer, to improve the manner in which legislation is produced and the way in which it is available digitally. The idea is currently attracting attention in the legal community, particularly on the question of whether it will lead to automation replacing human interpretation of the law. This note briefly describes Rules as Code, at least as conceived by a legislative drafter. It argues that this conception of Rules as Code does not have the over-reaching ambitions that it may sometimes appear to harbour, and in particular does not trespass into removing key interpretative functions.

Keywords – *Rules as Code, Legislation, coding, statutory interpretation*

Acknowledgements: *I thank Jeffrey Barnes and Pompeu Casanovas for their attentive reading and useful comments.*

Disclosure statement – *No potential conflict of interest was reported by the authors.*

License – *This work is under Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) <https://creativecommons.org/licenses/by-nc-sa/4.0/>*

Suggested citation: Waddington, M. 2020. “Research Note. Rules as Code.”, *Law in Context*, 37 (1): 179-186, DOI: <https://doi.org/10.26826/law-in-context.v37i1.134>

Summary

1. Introduction
2. What is distinctive about the Rules of Code approach?
 - 2.1 “Coding” before enacting
 - 2.2 Not executable programs ready to “implement” law, let alone automatically
3. Is Rules as Code a misconceived attempt to hand interpretation of law over from humans to computers?
 - 3.1 A source of misunderstanding
 - 3.2 Interpretation and imported material
 - 3.3 A spectrum of interpretation
4. References

1. INTRODUCTION

“Rules as Code” is currently a question, rather than a theory or a product. It is a label of convenience for a set of attempts to investigate the question of whether a new approach to coding (or at least marking up) legislation might produce a workable scheme which benefits both the way legislation is produced and the way it is made available through the medium of computer programs. The idea started in New Zealand (GNZ 2018, 2019), before spreading to New South Wales (GNSW 2019), and then to Canada¹, Jersey², the United Kingdom³ and elsewhere⁴, and then being reported on by OPSI⁵.

It is a practical movement led by developers and public sector digital innovators, whose preferences are for “show, don’t tell”, “sprints”, “wireframe models” and “minimum viable products”. That approach is welcome to the legislative drafters and policy officials who would adopt any resulting scheme, and who tend to prefer to see a demonstration of something workable, rather than a theoretical explanation. They also mostly take a neutral stance on the question of what the best technical approach might be (or even whether it is possible to find one that is workable).

This means that there is no unified⁶ line on what “Rules as Code” means (or even whether it is the best label to use). But it is most usefully seen as separate from (though related to) the idea of “computational law” in the sense of how law (of all kinds, not just legislation) might best be represented in code, and whether law can be adequately represented in that way. Of course, computer programs have been made, sold and used for decades to implement some legislation – nobody imagines that humans are still sitting with quill pens in social security and tax offices working out what benefits should be paid, or tax demanded. Equally, Rules as Code is separate from, but related to, the idea of “automated decision making” – it

is not mainly driven by ambitions for automation, and may represent one of the best defences against the urge in some quarters to automate legal decisions that should be made by humans.

But it is common for those coming upon the idea of Rules as Code for the first time to assume that it must have the same ambitions and faults as some of the wilder fringes of the computational law approach. Given Rules as Code is such an open and practical field, it is not surprising that those of us working on it (and others using the label for other purposes) sometimes use generalisations or shorthand that can be misconstrued as implying that Rules as Code is about removing interpretation, discretion and the human role from all of the law (or as if it is about AI, or as if no lawyers are working in it, or as if it is just hype and ignores the history of work on computing and law, and so on). This note is an attempt to cast some light on those assumptions and misleading impressions, and to invite those interested in the issues to engage positively without assuming that Rules as Code must be up to no good.

2. WHAT IS DISTINCTIVE ABOUT THE RULES AS CODE APPROACH?

2.1 “CODING” BEFORE ENACTING

A distinctive aspect of Rules as Code is that it builds on the “Better Rules” work undertaken in New Zealand. That means it is engaged with the fact that legislation is a conscious attempt to alter the law to give effect to some policy of the promoter of the legislation, and is interested in producing more coherent policy and legislation which is more readily comprehensible and effective. So one key distinguishing feature of Rules as Code is that it is about using technology to help at the stage of forming policy, writing drafting instructions, consulting on legislative proposals and drafting legislation. That is a very different

¹ See the work of Scott McNaughton (2020) from the Government of Canada.

² See our report to the UK and related jurisdictions’ knowledge-sharing event at the Bank of England on 28 February 2020. Cf. Waddington (2020).

³ See for example the work of Adam Wyner (2019), with the UK, Welsh & Scottish Parliamentary Counsel Offices, and on LegalRuleML, Wyner et al. (2017).

⁴ See for example the global 2019 Rules as Code Show and Tell Event at https://www.youtube.com/channel/UCjGU3qQWPMMy_DSRABNTuw and the work of Singapore Management University School of Law in partnership with Legalese on computational law - <https://news.smu.edu.sg/news/2020/03/11/smu-awarded-15-million-grant-computational-law-research> and <https://cclaw.smu.edu.sg/>

⁵ The Observatory of Public Sector Innovation (an initiative of the OECD) produced a report in its “Innovation Primer” series, titled “Cracking the Code: Rulemaking for humans and machines”, Cf. Mohun and Roberts (2020).

⁶ This note therefore should not be seen as definitive. It is one participant’s contribution, from the perspective of a legislative drafter (rather than a coder or other digital innovator), reflecting perhaps a less ambitious view of Rules as Code than many. Twitter gives a good overview of diversity of work and views on Rules as Code, using #RulesAsCode. A key figure is Pia Andrews, who has worked in New Zealand, New South Wales and Canada.

question from using AI or NLP to crawl back over existing legislation and try to interpret it to encode the results.

When lawmakers pass legislation they are acting on their understanding of the draft, which is normally aided by policy explanations and legal notes from civil servants and legislative drafters, which are generally published alongside the legislation (but do not have legal force, unlike the legislation itself). A coded version of the draft could help to ensure that the drafter and policy officers are understanding each other, that consultees can more easily grasp the ideas and suggest changes, and that lawmakers can see an illustration of what the legislation is intended to achieve. It could also help the drafter, and the policy maker to spot inconsistencies or unintended consequences in their drafts, improving the rigour that legislative drafters already apply to their work.

2.2 NOT EXECUTABLE PROGRAMS READY TO “IMPLEMENT” LAW, LET ALONE AUTOMATICALLY

The other main distinctive aspect is that Rules as Code is not about deciding what is the best way to turn legislation into code as such. The idea is to publish something when the legislation is enacted (and before if there is consultation) which is then kept up to date on an “application programming interface” (an API), so that others can use it to make programs. The “coding” that the government publishes would not be an executable program that implements the law automatically in some way. Instead others would be producing executable programs for their own particular purposes, and there is no reason to assume those would be primarily to implement the law, let alone to do so automatically (and of course a statute cannot be implemented automatically when it gives a human a

duty to make a non-delegable decision). Those purposes might be to explain the legislation to particular audiences (like an advice agency for young people), or to form part of the internal procedures of a particular business (like a bank’s procedures for reporting to a regulator), or as part of a government department’s implementation of their duties under the legislation.

That means the version published by government would need to be program-neutral as far as possible. Many of those working on the technical side are looking at “declarative” logic programming languages—taking the lead from the work in 1986 using Prolog on the *British Nationality Act* (Sergot et al. 1986a, 1986b)—instead of the usual “imperative” or “procedural” programming languages (such as Python, Java and others). Others are looking at extending mark-up languages such as Legal-RuleML⁷ and Akoma Ntoso⁸—including using tools like the “Rules Advanced Web Editor” or RAWE (Palmirani et al. 2013a, 2013b). Others favour using more traditional programming, but in a form such as OpenFisca⁹. Similarly, there is a question of whether policy officials and legislative drafters should learn to encode the rules themselves, or whether computing experts should join the team at the drafting stage. Jason Morris¹⁰, creator of Blawx¹¹, argues that new rule-making software should be as easy for lawyers to use as spreadsheets now are for accountants. In a related field, possibly applicable to Rules as Code, Professor Robert Kowalski is arguing that computers should be taught to understand natural languages, or at least controlled natural languages that are easy for non-programmers to use, and his “Logical English”¹² is inspired¹³ partly by the way that legislation uses a natural language (in our case English) in a fashion that has become increasingly disciplined and is

⁷ https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=legalruleml

⁸ <http://www.akomantoso.org/>

⁹ <https://openfisca.org/en/>

¹⁰ Jason also recently started working with the Centre for Computational Law (<https://cclaw.smu.edu.sg/>) at Singapore Management University, in partnership with Legalese (<https://legalese.com/>). Their work is much wider than Rules as Code (looking at commercial contracts and other legal texts, not just legislation), and does have much more ambitious goals (more akin to those that are often wrongly assumed to be behind all Rules as Code work).

¹¹ <https://www.blawx.com/>

¹² See <https://www.doc.ic.ac.uk/~rak/papers/LPOP.pdf>

¹³ See <https://www.doc.ic.ac.uk/~rak/>

¹⁴ It is interesting that Rules as Code started in Commonwealth countries and has spread in those countries. They have lawyers working full time for governments or parliaments as legislative drafters or parliamentary counsel, sharing information through the Commonwealth Association of Legislative Counsel. The modern style of legislative drafting in Commonwealth jurisdictions is much more disciplined than it was in the past, and much closer to a controlled natural language. In particular “and” and “or” are separated in lists where only one of those connectives can apply at any given level (paragraph, sub-paragraph and so on). Also the move from “shall” to “must”/“is” has shone more light on when the drafter is writing a constitutive rule (such as a definition, or a “counts as” rule) or a normative rule (which can be contravened, meaning the drafter should be satisfied that the legal consequences of contravention are clear even if they are not expressly set out) or a more troublesome rule on the borderline between the two (such as those related to validity or to whether a notice, application or appeal succeeds in having any effect in changing someone else’s legal status). For discussion of the ways these distinctions can be drawn, see for example Sartor (2006), particularly sections 11 to 13.

becoming much closer to a controlled natural language in the formal sense¹⁴.

The point is that Rules as Code is not wedded to any of these potential technology solutions so much as to the idea that the “coding” (or mark-up) of the legislation should be widely usable, traceable to the legislation, rather than adding material to reflect assumptions about procedures or implementation.

3. IS RULES AS CODE A MISCONCEIVED ATTEMPT TO HAND INTERPRETATION OF LAW OVER FROM HUMANS TO COMPUTERS?

3.1. A SOURCE OF MISUNDERSTANDING

Rules as Code is a practical proposal which will go nowhere unless it sparks interest among governments, so there can be a temptation to exaggerate what it is meant to achieve. It also needs to bring together many people from different professions who are not familiar with each other’s worlds, particularly lawyers with little experience of computing and developers with little experience of law. As such it can sometimes end up using over-simplifications to interest people, or just using expressions (particularly “interpretation” and “decision-making”) that mean different things to the different audiences. It is also easy to confuse Rules as Code with the wider “computational law” movement, and particularly with a false idea that humans bring only bias and irrationality whereas computers bring only impartiality and consistency, or with an idea that “Artificial Intelligence” will offer solutions to every human problem¹⁵. Lawyers coming to Rules as Code for the first time can assume that it is based on a failure to understand how law works and particularly how it is interpreted and the way in which that is different from the approach of a computer interpreting its programs. Lawyers also rightly worry about “automated decision-making” amounting to unlawful sub-delegation or to an abdication of a decision-making duty.

The relatively small number of legislative drafters involved in Rules as Code need to take on the concerns of other lawyers and try to clear up misconceptions. This note was prompted by a recent article headed “‘Rules as Code’ will let computers apply laws and regulations. But over-rigid interpretations would undermine our freedoms” (Governatori et al. 2020) as well as by a new

book titled *Is law computable?* but with the subtitle *Critical perspectives on law and artificial intelligence* (Deakin and Markou 2020). Both of those are rightly concerned with the dangers of digitisation in law, but the article also warns that Rules as Code risks undermining “the rules themselves, human freedoms, and democracy” if it involves a regulatory computer system that implements a single, government-determined interpretation of legal rules.

But what if, hype aside, Rules as Code is not really intended to be magically transformative, and is not really about automating legal decisions or about programs that implement law themselves? In reality Rules as Code is more about taking further steps in existing work on “fixing the plumbing” and making gradual improvements often associated with the slogan “bring back boring”. It is about applying human intelligence, rather than AI, and about the less glamorous ways in which computers are already handling law and could do better in aiding humans. It can involve merely highlighting the logical structures that the drafter is trying to create in the legislation, so that any use of that logic should always be traceable, explainable and open to correction or appeal in the same way as it is when a human follows the logic from the text. That could mean that legislative drafters and policy officers understand each other better during the drafting, that consultees can more easily grasp what it proposed and demonstrate how it could be changed, that inconsistencies in drafts can be spotted before they become problems, and that those who need to read the legislation can be helped to navigate complex sets of cross-references, conditions and exceptions to other exceptions. Those would represent significant benefits in themselves, without going anywhere near automating the implementation or enforcement of legislation.

3.2 INTERPRETATION AND IMPORTED MATERIAL

In the early days there was discussion in Rules as Code about the need to reassure people that the point is not to remove human discretion in legal decision-making functions. That led to talk of Rules as Code being more suited to something conceived as “prescriptive” legislation, rather than supposedly “discretionary” or “outcome-based” legislation, and that was sometimes framed in terms of the computer being able to “interpret” the one but not the other. But those ideas were never properly thought through, and amount to inadvertently selling Rules as Code

¹⁵ An excessive faith in technology can lead to the well-rehearsed problems of unaccountable “black box” AI. Equally problems such as the “robo-debt” issue in Australia can lead to excessive scepticism of all increased involvement of technology in the law.

short. The effort to reassure people is now moving away from the more straightforward concept of discretion¹⁶ to the more difficult concept of interpretation.

I mentioned above the idea that the “coding” (or mark-up) of the legislation should be traceable to what is already in the legislation, and should avoid adding material to reflect assumptions about procedures or implementation. There is a tendency among some critics to assume that the published code must itself be executable in a way that implements the legislation, and that the idea is therefore to hard-code in additional material (such as assumptions about steps being taken in a particular order, or information being supplied electronically) to turn the legislation into a set of procedures that can be coded. As a legislative drafter, I am on the wing of the Rules as Code movement that aims, as far as practicable, not even to include material derived from other legislation (as opposed to coding that links to the coding of another statute, where there is a cross-reference in the text or where an Interpretation Act applies) or from common law or other sources. Of course, no item of legislation can be understood fully without reference to material outside that item of legislation, but that does not stop us finding it worthwhile to publish the text of legislation as it stands¹⁷. Modern legislative drafters (at least the full-time professional drafters found in Commonwealth countries) use a variety of conventions, as well as Interpretation Acts, to achieve the desired result while avoiding cumbersome expression that would lose or confuse human readers. For example, there is the convention that, unless there is some other indicator, the same word will mean the same thing inside one piece of legislation, while a different word will have been used deliberately to carry a different meaning. Jurisdictions often have rules such as that expressions used in primary legislation have the same meaning in secondary legislation made under it, and conventions that although singular and plural imply each other the singular will be used (to give a more precise focus). We find that many

lawyers are unaware of these rules, having been taught statutory interpretation as if it was always a question of applying various rules to cases of ambiguity, rather than of understanding how definitions work and what are the conventions and rules applicable in their jurisdiction.

We also rely on being able to say, in a particular piece of legislation, merely “A person who does X commits an offence and is liable to a fine of £Y and to imprisonment for Z years”, without rehearsing or explaining the general criminal law (whether statutory or case law). Similarly, we can say a public body “must” do something, and leave it at that, relying on our jurisdiction’s body of administrative law to deal with what happens if the public body does not do what we said it must – we do not have to try to pin that down with a statutory formula that will inevitably be left behind by case-law and come adrift from later statutory versions of the formula. But it is still worthwhile for us to publish the text of the legislation without publishing a compendium of criminal and administrative case-law cross-referenced to each published item of legislation. Equally, Rules as Code does not need to encode all of this external material for it to be worthwhile creating and publishing a coded/marked-up version of an individual piece of legislation. So if the offence provision is “If person A does X or Y, but not Z, then A commits an offence [and is liable to ...]”, then it is worthwhile to code this to capture only the “if”, “or”, “but not” and “then”, plus the fact that A has to be the same person in both appearances (and, if the statute defines Z, perhaps a reference to that definition), without trying to capture anything about any common law rules on the necessary *mens rea* or the presumption of innocence, or any relevant criminal procedure rules and so on.

Developers have a natural tendency to want to show-case work where the computer produces results, so much of the initial work has been on the readily computable areas of social security benefits and tax. But work has now moved on to fields such as gambling regulation,

¹⁶ As far as discretion is concerned, the idea is to use the computer to help guide the human reader to the correct point at which a discretionary decision has to be made by a human, not to have the computer make or suggest the “correct” decision. But see below as to the human interpretation needed for each step on the way to any of those decisions.

¹⁷ This applies even in an extreme case, as follows. Most jurisdictions publish the text of their legislation on their websites, separately from any publication of case-law (some try to annotate legislation with references to case-law, but that is very difficult to do, particularly if the jurisdiction publishes its legislation as amended). A court might declare a statutory provision *ultra vires* or say words should be read in or ignored, but the legislation website will not take down or alter the text of the legislation just because that has happened. If subsequent amending legislation is passed to react to the court decision then those amendments will be published on the legislation website in due course, but the legislation website will not jump the gun by rewriting the published legislation to reflect the court decision. So jurisdictions already operate on the basis that there is utility in publishing the text of legislation as it stands, even when that legislation fails to reflect the true state of the law, as long as it is clear that no more is being achieved. Similarly, there would be utility in publishing coding that reflects that legislation, and being clear that this is all that is being achieved. It would then be left to those who want to make programs from it to decide whether they offer something based just on the legislation (answering the question “what does the X Act say?” rather than “what are my rights?”) or whether and how they try to go further by feeding in corrections from court cases or other external material.

reporting obligations of banks, and charity registration. It will become increasingly obvious that the benefits of Rules as Code are more about improving the way the rules are constructed (to avoid inconsistencies and loopholes, rather than to be machine-executable) and about how they can be explained and altered, as well as about assisting human readers to follow the often long and twisting logical strings that the drafter had to embed in the text.

3.3. A SPECTRUM OF INTERPRETATION

The role of interpretation in law is obviously a vast topic in academic law, and then there is an odd corner labelled “statutory interpretation” in which it seems all too often introductory courses encourage students to believe in magical “golden rules” that have been debunked long ago by specialist texts such as Bennion (Bailey and Norbury 2020). How judges make new law has also been a long-standing fascination for academic law in the common law world. The socio-legal studies movement has led to a focus on social practices, not just in courts, but in administrative offices and in the way citizens experience what the legal system produces. But relatively little attention is given in legal education, even in socio-legal courses, to how legislation is created and to how that makes it different from case-law and general principles.

On the other hand, people with backgrounds in technology rather than law tend to assume that there are “government regulations” and that the government has the authority to dictate how those regulations are interpreted. One of the features of Rules as Code is the way that legislative drafters have been able to explain to their technical colleagues that once legislation is enacted it is for the courts, not the government or Parliament, to interpret it.

But this leaves us with an over-simplified idea of the interpretation involved in the course of applying legislation to facts. A computer does not know what “dog” means¹⁸, because it does not know what a dog is. Even the most ardent AI evangelists have to recognise that AI can only

take us so far with that. In reality a computer does not know anything—it is a question of whether the inputs come from other programs or from humans. So if there was a computerised register of dogs, and the legislation was to say that for all legal purposes something only counts as a dog if it is on that register (and anything on that register counts as a dog) and that a dog was to be presumed/assumed to live at the address given for it on the register, then a program encoding an Act that mentions dogs could rely on computerised input from that register to establish whether a dog lives at a particular address. If the Act required dog-owners to display a sign on their house to be visible from the road, then perhaps AI could be trained to use something like *Google StreetView* to check whether there was a sign on a house that appeared on the register, and the computer could send an automated email imposing a civil penalty on a registered owner who did not have a sign. But of course this is nonsense—no sensible legislature is going to say that the computerized dog register is legally definitive of what a dog is¹⁹, and any lawyer would expect the registered owner to be given a right of appeal against any penalty imposed based on AI reading images of streets. Outside of the more fevered imaginations of some AI proponents, and the hype of some people with no qualms about selling fake miracles, there is no sense in which anyone is really proposing that the dog registration and sign display rules should be fully automated.

But what a computer can usefully do is take a human through a set of questions, where the human inputs an answer based on their interpretation of the words in the question, and the computer uses that answer to take the human to the correct next question. In typical legislation on dog registration, say, there would be a set of conditions for dog registration to be mandatory, and a set of exceptions, some of which will have their own exceptions and some of which will cross-refer to other rules and complex definitions. There will then be exceptions and cross-references to the rule about displaying a sign, which will not correlate completely with the registration rule, with the

¹⁸ But what the legislative drafter (or accompanying coder) can do is to mark up the text, or have an element in the code, to tell the computer that every occurrence of those three letters (and the four letters for “dogs”, or any other grammatical variants) refers to the same concept (and that “the/that dog” refers back to an immediately preceding instance of “a dog”, and so on). See above on how these conventions are followed in modern legislative drafting in Commonwealth countries. This also reflects the difference between Rules as Code, which has humans apply the mark-up/coding during the drafting, and other approaches that use AI to try (after enactment) to fathom what “dog” was doing in the text. With the mark-up/coding published by government, as envisaged in Rules as Code, the point would be that a human still has to answer the question “is this a dog”. Meanwhile the mark-up/coding would enable others (whether in government or not) to create software that takes a human through the logic to the correct part of the statute to end up at that question, based on the human’s answers to other questions about the other English expressions used in that statute.

¹⁹ Equally, the published mark-up/coding would just be “official” in the sense that any government explanatory text is official. The coding is not intended to say, when the legislation does not, that something should not be recognised as a dog unless an electronic register confirms that it is a dog. That would be going beyond the legislation, and any “official” status could not be enough to give it any credence or usefulness, never mind give it any legal effect.

result that many readers of the legislation will become lost when tracking back and forth to work out their position, even though they may have no trouble (on the particular facts) in identifying whether the animal in question is a dog, whether the sign is visible from the road, and so on. That is where the computer can be used to capture the “if-then-and-or-not” logic²⁰ of the legislation, the use of defined terms (and the use of “means” or “includes” in definitions), and so on, rather than the actual meaning of “dog” or “visible from the road”. If it is an offence to fail to display the sign, then the court will have to answer yes or no to whether there was a dog registered at that address and whether the sign was visible from the road.

Ultimately a court, using a natural language, will “interpret” all the relevant words of a statute when applying it to a set of facts – interpretation is not just about ambiguous words. But there is a spectrum of elements of a statute, where the elements need a greater or lesser degree of respect for the court’s function of interpretation where there is no claim of ambiguity. So for example if there is a Dogs Amendment Act, and it says that paragraph (1) of section 1 of the Dogs Act (as it stands when the Amendment Act comes into force) is amended to substitute “poodle” for “spaniel”, then, as long as there is a s1(1) which has the word “spaniel”, we are not really saying that identifying this provision and switching the words is at a level of interpretation at which we want the court to be able to make its own mind up about the meaning (the court might strike down the amendment as ultra vires, or hold that the provision as amended should be read to cover both poodles and spaniels on constitutional grounds, but that is a different story). If the coding captures the substitution provision, nobody is going to object to that as usurping the judicial or human function of interpretation. In fact several jurisdictions already do produce and publish consolidated versions of their legislation, using XML coding in the original legislation and the amending legislation to do the job (by identifying a paragraph as being s1(3)(d) of the X Act as amended up to a certain date, for example), without causing an outcry about having added in and hard-coded their own interpretation.²¹

Similarly the Dogs Act, before the amendment, might say that if the dog is a spaniel then it is not covered by an exception for dogs owned by farmers from an exceptional

rule that farmers do not need to display the sign as long as they register their dogs, and it might define farmers as people who work in any agricultural business. This is what is going to take most people time and effort to wade through, and where most mistakes will be made by those implementing the legislation or checking their rights, but we do not see this as sacred ground on which human input is essential. The human input is needed for whether the dog counts as a poodle, whether the owner counts as a person who works in an agricultural business, and whether what the farmer did counts as displaying the sign or registering the dog. Even those basic terms require human input, let alone a provision saying the farmer commits an offence by failing to take “reasonable” steps to ensure the sign is not removed by anyone else, where it is obvious that the court will have to apply its own human approach to working out what is reasonable. The human input is essential for those points, rather than for unravelling the nested exceptions and their relationship to the main rule. The legislative drafter will try to make that logic as simple and transparent, and free of inconsistencies, as possible in the time available and under whatever other constraints might be involved (such as a need to mirror some other provision). If the coding in Rules as Code merely picks out the if-then-and-or-not logic of these exceptions and definitions (and points up a scenario where the first draft would produce an unintended result), then it will have done an extremely valuable job. But it will not have tried to usurp the human interpretative role by trying to capture the entire meaning of the Act’s provisions or automate the entire handling of the Act. This is much less exciting for those who want to see Rules as Code as just another attempt to have computers take over the world of law, but it is still exciting enough for legislative drafters.

4. REFERENCES

1. Bailey, D., Norbury, L. 2020. *Bennion, Bailey and Norbury on Statutory Interpretation*. 8th Edition, LexisNexis Butterworths.
2. Deakin, S., Markou, C. (eds.). 2020. *Is Law Computable? Critical Perspectives on Law and Artificial Intelligence*, Oxford: Hart Publishing.

²⁰ The logic that captures the relations in “if X then Y”, “X and Y”, “X or Y” and “not X”. For instance “or” can be inclusive or exclusive (not both X and Y). Usually the intended meaning is obvious enough to a human, so we do not overburden the reader by spelling out in the English text. But when more is needed a drafter might write “X or Y, but not both”, “either or both of X and Y”, or “any one or more of the following”, and so on.

²¹ See for example New South Wales <https://beta.legislation.nsw.gov.au/help/xml-pdf-files> and the United Kingdom <https://www.legislation.gov.uk/developer/formats/xml>

3. GNZ. Government of New Zealand. 2018. *Better Rules for Government Discovery Report*. March, <https://www.digital.govt.nz/dmsdocument/95-better-rules-for-government-discovery-report/html>
4. GNZ. Government of New Zealand. 2019. *What is Better Rules?* By H. Fraser, 20 December, <https://www.digital.govt.nz/blog/what-is-better-rules/>
5. GNSW. Government of New South Wales. 2019. *Rules as Code*. <https://www.digital.nsw.gov.au/digital-transformation/policy-lab/rules-code-0>
6. Governatori, G., Barnes, J., Zeleznikow, J., de Koker, L., Poblet, M., Hashmi, M., Casanovas, P. 2020. "‘Rules as Code’ will let computers apply laws and regulations. But over-rigid interpretations would undermine our freedoms", *The Conversation*, November 26th. <https://theconversation.com/rules-as-code-will-let-computers-apply-laws-and-regulations-but-over-rigid-interpretations-would-undermine-our-freedoms-149992>
7. McNaughton, S. 2019. "Week 64 — The State of Rules as Code in the Government of Canada". <https://medium.com/@mcnaughton.sa/week-64-the-state-of-rules-as-code-in-the-government-of-canada-8f3cb327448d>
8. Mohun, J. and Roberts, A. 2020. "Cracking the code: Rulemaking for humans and machines", *OECD Working Papers on Public Governance*, No. 42, OECD Publishing, Paris, <https://doi.org/10.1787/3afe6ba5-en>.
9. Palmirani, M., Cervone, L., Bujor, O. and Chiappetta, M., 2013a. "RAWE: a web editor for rule markup in Legal-RuleML". *Proceedings of the 7th International Rule Challenge, the HLT and the DC at RuleML2013*, the 8th International Symposium on Rules. In *CEUR workshop proceedings*, Vol. 1004. <http://ceur-ws.org/Vol-1004/>
10. Palmirani, M., Cervone, L., Bujor, O., Chiappetta, M. 2013b. "RAWE: An Editor for Rule Markup of Legal Texts", OA-SIS, <https://2013.ruleml.org/presentations/RuleML-palmirani2013v2.pdf>
11. Sartor, G., 2006. "Fundamental legal concepts: A formal and teleological characterisation." *Artificial Intelligence and Law*, 14 (1-2): 101-142.
12. Sergot, M.J., Sadri, F., Kowalski, R.A., Kriwaczek, F., Hammond, P. and Cory, H.T. 1986a. "The British Nationality Act as a logic program", *Communications of the ACM*, 29(5): 370-386.
13. Sergot, M., Cory, T., Hammond, P., Kowalski, R., Kriwaczek, F. and Sadri, F., 1986b. "Formalisation of the British nationality act", *International Review of Law, Computers & Technology*, 2 (1): 40-52.
14. Waddington, M. 2020. "Non-financial legislation: human input, discretion & logic." Rules as Code Knowledge-Sharing Event, 28 February, Bank of England. <https://www.slideshare.net/MatthewWaddington3/non-financial-legislation-human-input-discretion-and-logic-28220>
15. Wyner, A. 2019. "Annotating and Querying Content within Machine-readable Legal Instruments", Office of the Parliamentary Counsel, London May 21. https://azwyner.info/wp-content/uploads/2019/05/WynerUKGovt_v3.pdf
16. Wyner, A., Gough, F., Levy, F., Lynch, M. and Nazarenko, A. 2017. "On Annotation of the Textual Contents of Scottish Legal Instruments". A. Wyner and G. Casini (eds.) *Legal Knowledge and Information Systems*, JURIX-2017, Amsterdam: IOS Press, 101-106.