# Digitising legislation: progress and prospects

**Preprint** · March 2024

DOI: 10.31235/osf.io/fpczw

# Digitising legislation: progress and prospects

**Matthew Waddington** (ORCiD 0000-0002-2396-5417), Legislative Drafter, Lead for Computer-Readable Legislation Project at Legislative Drafting Office, Jersey (own views, not employer's)

**Pre-print** as at 1 March 2024, adapted version of draft chapter 10 & appendix submitted for proposed book "Legislating for risk and precaution - Bridging the divide between science and law for biosecurity", editors Black, Moloney & Graffham, proposed publisher CRC Press (Taylor & Francis), expected 2024 or 2025

## Contents

This chapter looks at the development of the technology that has been applied to producing legislation, and considers where that may lead – what might be achieved as we move further in digitising legislation? The chapter looks at the effects of the move from paper to word processing and web publication, and at where XML systems might lead in capturing the structure of the layout and numbering of legislation. It then turns to the question of capturing the logical structure of legislative provisions, and the risks of trying to capture the meanings of words used particularly when that is intended to lead to automation of the execution of the processes set out in the legislation. It describes the "Rules as Code" movement and progress in "computational law". It considers the difficulties inherent in digitising the logic of existing current statute books, and then turns to how technology might help with drafting in future. It looks at what help might be given by generative Artificial Intelligence (and "Large Language Models") in various tasks related to the drafting and understanding of legislation.

## *The story so far*

### *1. From analogue to digital*

10.1. Users of legislation, and those involved in its production, now expect the text of legislative instruments to be published online, preferably including versions that are consolidated and updated for all current amendments. It is not so long ago that everyone expected legislation to by in type on paper, with literal "cut and paste" methods of updating paper copies. But we have not yet fully implemented the already apparent potential benefits of digitisation, and we are still working out what new benefits might be available in the near future. The progress so far has been unevenly distributed – some larger jurisdictions have modern purpose-built drafting tools and websites, while some smaller jurisdictions without resources are still drafting with word-processors (without templates or other technical standardisation) and have only recently been able to publish their enacted legislation as static pdfs without any hyperlinking or updating. This chapter looks at the direction of travel in the increasing digitisation of legislation, along with the benefits to be gained and the pitfalls to be avoided. The progress in digitisation has also, as a side product, led to new appreciation of the logical structure of legislation, fostering increased rigour in legislative drafting.

10.2. We have almost all moved already from typewriters and paper to word-processors and publishing on websites (in at least pdfs and possibly html). Modern word-processors allow use of styles, heading hierarchies and templates to add computer-readable structure to legislative drafts, enabling, for instance, automatic production of tables of contents, and possibly automating work-flows. The word-processors and web publication also enable us to add hyperlinks to the text of legislation, which should help with internal and external cross-referencing. But even the most advanced offices have been slower than the public might expect in getting the fullest benefit out of the potential for hyperlinking. Some have hyperlinked express cross-references, but hardly any have made progress with the obvious idea that defined terms should be hyperlinked back to their definitions (preferably including definitions that are in the Interpretation Act, or in an Act under which the present legislation was made, for instance).[1]

10.3. Many of the better resourced legislative drafting offices no longer use a word processor to draft, followed by conversion to pdf and html for publication. Instead they use systems based on “XML”, which allows richer metadata to be included in files. They draft their legislation in an XML editor and then publish it in XML with automatic conversion to html & pdf for a website. These XML systems automatically tag each of the structural parts of the legislative instrument with its identity and that of the instrument – such as “section 3(4)(a)(ii) of draft 5 of Amendment 6 to the Dogs Law” (changing automatically through the life cycle of the draft). The metadata that identifies these elements can be read by a computer and is very useful, such as for enabling a website user to call up just a particular section, or a whole Part of an Act, and for accurate hyperlinking of cross-references inside the Act. Many offices are using a form of XML called Akoma Ntoso,[2] which is designed to help track how parliamentary documents relate to legislation (so a bill and its accompanying documents can be traced into the much amended piece of enacted legislation that it subsequently became) and to enable court judgments to use similar metadata (which would help automatically annotate legislation with links to judgments that considered it). Even the most advanced offices still have room for improvement, and for Valentine’s Day in 2023 Hamish Fraser wrote a “Love letter to the Parliamentary Counsel of the world”,[3] starting a discussion of whether legislative drafters and the official publishers of legislation can do better in giving tags, that are easily readable by both humans and computers, to uniquely identify every single paragraph of a piece of legislation.

10.4. There is a strong sense in which the published digital versions of each jurisdiction’s legislation form a data set that has not yet been fully exploited. Until recently it has been assumed that advances on this front would need greater standardisation of format (and possibly expression) to turn statute books into “structured” data, using something like Akoma Ntoso. But since the launch of ChatGPT in November 2022, there has been speculation that the published legislation as it stands will be able to be read and understood by “generative Artificial Intelligence” using “Large Language Models” (see below).

## 2. *Potential benefits and pitfalls of further progress*

10.5. The next move on, after getting a computer to know where a provision is in a piece of legislation, would be to get the computer to be able to read some of the meaning of that provision. Much legislation is already effectively digitised in this way in that a version of it has been turned into a computer program to run a process under the legislation. We do not expect government departments administering tax and social security to have humans working out each case and doing the calculations manually. But the published versions of the legislation do not have any links to the relevant elements of those programs (and the programs are not made public). The idea behind “computational law” is to encode legislation in a way that can be tied to the text of the legislation and changed when the legislation is amended. Among the main current centres of work on computational law are MIT,[4] Stanford Law School’s Codex Centre for Legal Informatics[5] and the Centre for Computational Law at Singapore Management University.[6]

[1] The history, and potential uses, of hyperlinks in legislation are set out in Leon Qiu (2023) “Facilitating Access to Justice through Hypertextualising the Statute Book”, LLB dissertation at Edinburgh Law School (unpublished).

[2] http://www.akomantoso.org/

[3] https://hamish.dev/the-love-letter-contextualised

[4] https://law.mit.edu/

[5] https://law.stanford.edu/codex-the-stanford-center-for-legal-informatics/complaw-corner/

[6] https://cclaw.smu.edu.sg/

10.6. It might be that some legislation can be coded in a way that appropriately enables procedures under that legislation to be automated, but that is both risky and difficult, and it is not the only available benefit. One of the key benefits of capturing elements of meaning is in being able to explain the legislation to lawyers or lay people, and enabling them to check how the legislation would apply to their particular factual scenarios. The best current demonstration of this is DataLex, which is a free, web-based program that asks questions taken from legislation and gives results in which it explains every step. Its makers, AustLII, have put up examples of four Australian Acts for which anyone can use DataLex to see how the Act applies given the user's answers to the program's questions.

10.7. The key group researching the risks of digitising law (and particularly of AI in law) is led by Mireille Hildebrandt at COHUBICOL. They warn that text-driven legal rules cannot be converted into computer programs without changing their character, particularly if meanings are fixed, contestability is lost and the courts' role in interpretation is ignored.[7] A particular concern is how the increasing digitisation of law will impact on the rule of law, on which a balanced perspective is given by Lisa Burton Crawford.[8] Australia's "robo-debt" scandal is one of the most worrying concrete examples of what can go wrong when the execution of legislation is delegated to a computer.[9]

10.8. But these objections have much less force against programs like DataLex that merely have the computer "read" the logical structure, to be able to guide a human to the right questions, and do not get the computer to assign meanings to the substantive undefined expressions in the text (steering well clear of automating decisions, or determine what counts as "reasonable", and so on). There is a temptation to take an absolutist position that every word in an item of legislation is open to interpretation, but this ignores the reality that some elements are more open than others. For instance, a provision might say "In section 5(3) of the Dogs Act, for "spaniel" there is substituted "poodle".". As long as there is a Dogs Act, and the word "spaniel" does appear in section 5(3), there is no real issue of interpretation of this provision (though "spaniel" and "poodle" obviously require interpretation in the original Act and its amended version), and nobody would be alarmed by the fact that publishers of the legislation already routinely use automated systems to implement this sort of legislation by making the necessary textual changes.

## *Possible future avenues for digitisation of legislation*

### *3. Making the meaning or logical structure of new legislation computer-readable*

10.9. Many legislative provisions are expressly in the form "if this, that". But, even for those provisions that do not use "if" expressly, they can almost always still effectively be recast as "if these conditions are met, then there is this legal effect". Computers of course are famous for being able to handle if-then concepts, so this is a starting point towards having the computer understand the logical structure without understanding the meaning of the substantive terms. For example, in a provision that says "if a person drives a vehicle, the person must wear a seat belt", the computer does not need to understand the meaning of any part of "a person drives a vehicle" or "the person must wear a seat belt", but if it understands the "if" structure it can ask a human "is it the case that a person drives a vehicle". It can be programmed to require the human to pick "yes", "no" or "unknown", and then to return the answer "the person must wear a seat belt" for "yes", "this provision does not apply" for "no", and "it is unknown whether this provision applies" for "unknown". In fact this is so easily computable that no

[7] For COHUBICOL see https://www.cohubicol.com/, and their Journal of Cross-disciplinary Research in Computational Law at https://journalcrcl.org/crcl. In particular they have published two reports on their website – Laurence Diver et al (2023) "Research Study on Text-Driven Law" and Pauline McBride et al (2024) "Research Study on Computational Law" – and they have published a "Typology of Legal Technologies" at https://publications.cohubicol.com/typology/ . See also the work of the Allens Hub for Technology, Law and Innovation at the University of New South Wales https://www.allenshub.unsw.edu.au/ .

[8] Crawford, Lisa Burton (2023) "Rules as code and the rule of law" [2023] Public Law 3:402-423 https://search.informit.org/doi/10.3316/agispt.20230721091881

[9] See Anna Huggins (2020) "Executive Power in the Digital Age: Automation, Statutory Interpretation and Administrative Law" in Janina Boughey and Lisa Burton Crawford (eds), "*Interpreting Executive Power*" (Federation Press, 2020) 111-128.

traditional programming is needed - an Excel spreadsheet can be set up to do it by anyone who can use Excel's if-then function. Again the point is that Excel understands the if-then element, but not the meaning of the questions or answers it is giving.

10.10. Legislative drafters also now ensure they are careful with their use of "and", "or" and "not", such as by using different levels of sub-paragraph with different numbering (and indenting) to ensure "and" and "or" apply to the correct elements in a list. So, for instance, "a pig and a cow or a goat" is ambiguous (between "a goat alone, or a pig and cow together" or "a pig plus either a cow or a goat"), but we draft it as either "(a) a pig; and (b) either - (i) a cow, or (ii) a goat" or, if the other meaning is needed, "(a) both - (i) a pig, and (ii) a cow; or (b) a goat" (with the roman numeral elements indented further than the letter elements). Again computers are very happy handling this sort of conjunction and disjunction, as well as the concept of negation. We are therefore able to capture at least the "if", "and", "or" and "not" elements in our drafts in a way that computers can read and use to help humans address the correct questions, and that is tied very closely to the wording of the legislation. The example in Appendix [X] illustrates this idea of capturing only the logical structure, and not the meaning of the substantive elements, by using "pseudo-code" and Excel (as mentioned above) to identify two conditions, and a resulting legal effect, in an imaginary provision of a Biosecurity Act. The example demonstrates that a program, in this case as generic and common as Excel, can produce correct results from the user's answers to automatically generated questions, with the questions and results sticking faithfully to the wording of the original legislation (and with any changes to the conditions or effect elements of the legislative text being automatically reproduced in the corresponding questions and results).

10.11. The rigour of disciplined use of language is shared by both legislative drafters and computer programmers. One stream of work in this field is with "controlled natural languages" which are versions of English that are disciplined in such a way that a computer can read elements of them. Robert Kowalski is working on a controlled natural language called "Logical English" which he says was "inspired in part by the language of law"[10] and which makes "if", "and", "or" and "not" computer-readable in human-readable sentences. An alternative approach, adding metadata to the text of the legislation, was taken by LegalRuleML,[11] which could capture these elements and more.

10.12. This brings us to "Rules as Code" (or "RaC"),[12] which is the idea that it should be possible to work out how to encode (or mark up) key elements of draft legislation (and the policy that the legislation is based on), before it is enacted. RaC started in 2018 in New Zealand with "Better Rules"[13] (attempting to improve rule-making by government) and it spread to the government of New South Wales[14] and

[10] In Robert Kowalski (2020) "Logical English", position paper for Logic and Practice of Programming (LPOP) 2020, available at https://www.doc.ic.ac.uk/~rak/papers/LPOP.pdf. See also Robert Kowalski and Jacinto Dávila (2023) "Logical English Demonstration", Proceedings 39th International Conference on Logic Programming, available at https://www.doc.ic.ac.uk/~rak/papers/Logical_English_Demonstration_for_ICLP.pdf .

[11] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=legalruleml and http://sinatra.cirsfid.unibo.it/rawe-legregsw/

[12] See "Rules as Code Handbook wiki" available online at https://github.com/Rules-as-Code-League/RaC-Handbook/wiki/, and Mohun, J. & Roberts, A. (2020). Cracking the code: Rulemaking for humans and machines. OECD Working Papers on Public Governance, No 42, https://doi.org/10.1787/3afe6ba5-en

[13] https://www.digital.govt.nz/blog/what-is-better-rules/ and https://www.digital.govt.nz/dmsdocument/95-better-rules-for-government-discovery-report/html .

[14] https://www.digital.nsw.gov.au/article/rules-code-nsw-joins-worldwide-movement-make-better-rules

the national government of Australia[15], but also to France[16], Canada[17], Singapore[18], the UK[19], Jersey[20] and elsewhere. The point is to help avoid drafting inconsistencies and errors, and to contribute to ensuring that the drafter, the policy official (and politician responsible) and any consultees are not misunderstanding each other. The result would be published, not as a replacement for the English (or other natural language) text as enacted, but as a non-authoritative supplement like official explanatory notes. It would be made available and kept updated (as the text of legislation is currently) by publishing it in a form that can be called up by computer programs designed to use it, possibly through an API (Application Programming Interface). That would mean everyone shared a single version of the legislation with government endorsement (but not enacted status, and not claiming to displace the interpretative function of the courts). That would improve explanation, and possibly implementation, but it would also help legislative drafting to continue its history of gradually increasing rigour and discipline.

## 4. *Drafting differently or more rigorously?*

10.13. In 1978, Layman E. Allen's "normalized legal drafting"[21] started a long trend (generally ignored by drafters) of making suggestions that drafters should radically change how they draft, to use lessons from modern logic and computing. In 2000 Thomas Blackwell[22] suggested drafters should learn from computing's concept of "object-oriented" analysis. More recently the team at AustLII[23] created "DataLex"[24] and are trying to persuade legislative drafters to phrase their drafts in ways that can be read by AustLII's "yLegis" preprocessor[25] to convert it into their "yScript" language[26] which can be used to run DataLex consultations about how the legislation applies to facts given by a human user (through answering questions posed by the computer, drawn from the legislation).[27]

10.14. It seems unlikely that drafters will be persuaded to draft in ways that they believe are less easy for humans to understand, in order to make the legislation easier for computers to understand, but that

[15] https://www.govcms.gov.au/news-events/news/govcms-announces-enterprise-adoption-rules-code

[16] https://openfisca.org/en/

[17] https://www.csps-efpc.gc.ca/video/rules-as-code-1-eng.aspx

[18] https://ink.library.smu.edu.sg/sol_research/3093/

[19] https://www.publictechnology.net/2022/10/24/economics-and-finance/dwp-looks-embed-machine-readable-laws-digital-universal-credit-navigator/ and https://www.bankofengland.co.uk/-/media/boe/files/paper/2020/transforming-data-collection-from-the-uk-financial-sector.pdf

[20] https://substack.com/@digitallegislation and https://www.gov.je/Government/NonexecLegal/StatesGreffe/SiteAssets/pages/legislativedraftingoffice/Introduction%20to%20the%20Computer-Readable%20Legislation%20Project.pdf

[21] See Allen, L. E. & Engholm, C. R. (1978) "Normalized Legal Drafting and the Query Method", *Journal of Legal Education*, Volume 29, pp. 380-412, and later Allen, L. E. & Saxon, C. S. (1986) "Analysis of the Logical Structure of Legal Rules by a Modernized and Formalized Version of Hohfeld Fundamental Legal Conceptions" in: A. A. Martino & F. S. Natali, eds "*Automated Analysis of Legal Texts*" Amsterdam: North-Holland, p. 385–451.

[22] Blackwell, Thomas (2000) "Finally Adding Method to the Madness: Applying Principles of Object-Oriented Analysis and Design to Legislative Drafting", *New York University Journal of Legislation and Public Policy* Vol. 3, No. 2, Spring 2000, p. 227 - 293 https://nyujlpp.org/wp-content/uploads/2012/10/Thomas-Blackwell-Finally-Adding-Method-to-the-Madness.pdf

[23] https://www.austlii.edu.au/

[24] https://datalex.org/ and https://austlii.community/wiki/DataLex/

[25] https://austlii.community/foswiki/pub/DataLex/WebHome/ylegis-intro.pdf

[26] https://austlii.community/foswiki/pub/DataLex/WebHome/ys-manual.pdf

[27] See Mowbray, Andrew and Chung, Philip and Greenleaf, Graham (2023) "Explainable AI (XAI) in Rules as Code (RaC): The DataLex approach" Computer Law and Security Review, 2023, 48, p105771 https://www.sciencedirect.com/science/article/pii/S0267364922001145 & https://ssrn.com/abstract=4093026

remains to be seen. Meanwhile, drafters have been looking at systematising their use of precedents (which are generally frowned on) and office directions on how to draft particular types of provisions for consistency. Inspired by the way that computer programmers had taken up “pattern languages”, drafters in the UK drafting offices worked together to produce “Common legislative solutions”[28], which set out questions that policy officers should think through when asking for typical sorts of provisions as part of their drafting instructions for a Bill, such as the setting up of a statutory body or licensing scheme, the creation of offences or search powers. The guidance includes pointing policy officers to examples of those provisions that are already on the statute book. From there it is not a huge step to look at improving the drafting of standard provisions for these topics, and parsing those standard provisions to make them computer-readable.[29]

10.15. A separate question is whether in the near future we might find there are improved drafting tools available that could help legislative drafters to create, flag and check the logical structures that we build in draft legislation. Oracle Intelligent Advisor[30] is an existing commercially available program that does that for writing business rules, but was originally developed for legislation and could serve as a model. Grimmelmann has suggested that legal drafters need the equivalent of the computer programmer’s “integrated development environment” (“IDE”) and “Jupyter Notebooks”, to help us to see and test the logical structures we are building in our drafts.[31] Meanwhile a team working on Israel’s legislation has fleshed out what an “Integrated Legislation Drafting Environment” might look like.[32]

### *3. Bolstering the logical rigour of drafts*

10.16. A different issue is one of the most promising to have come out of work by legislative drafters in this field, regardless of whether any new technology becomes available, namely that the increased focus on logical concepts underlying legislation will add further rigour to modern legislative drafting.

10.17. The styles of legislative drafting are very different in common law and civil law countries (and even between France and Germany), but the common law countries have seen a gradual improvement in the rigour of their approach since George Coode[33] urged a systematic way of looking at legislative sentences in the nineteenth century. Most recently the plain English movement persuaded most Anglophone legislative drafting offices to drop “shall” in favour of “must”, but the side-effect of that was to reveal to us that we had been sprinkling “shall” liberally where it did not equate to “must”, as in “shall be entitled to”, “shall mean”, “shall be established” and so on. That helped us to become more focused on whether we are creating an obligation (which can be contravened, and needs a link to some consequence for contravention) or a provision that has effect by operation of law. In turn, work on

[28] Office of Parliamentary Counsel (2022) "Common Legislative Solutions: A Guide to Tackling Recurring Policy Issues in Legislation" at https://www.gov.uk/government/publications/common-legislative-solutions-a-guide-to-tackling-recurring-policy-issues-in-legislation. See also Norbury, Luke (2018) "Common Legislative Solutions" *The Loophole*, Issue No. 3 of 2018, p2 https://calc.ngo/publications/loopholes

[29] For more on common legislative solutions and how pattern approaches could help drafting, see Leon Qiu, Yiwei Lu, Burkhard Schafer (2024) "Formalisation Memories: Towards a Pattern Approach to Legal Design" in proceedings of 27th Internationales Rechtsinformatik symposion 2024 (IRIS2024) https://iris-conferences.eu/ and https://easychair.org/smart-program/IRIS24/2024-02-16.html#talk:244823

[30]
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Overview/Overview.htm

[31] James Grimmelmann (2022) “Programming Languages and Law: A Research Agenda”, *Proceedings of the 2022 ACM Symposium on Computer Science and Law*, p155–165 https://dl.acm.org/doi/abs/10.1145/3511265.3550447 . See particularly paragraphs 3.7 and 3.8

[32] Schwartz, Elhanan and Bar-Siman-Tov, Ittai and Gelbard, Roy (2023) “Design Principles for Integrated Legislation Drafting Environment” https://ssrn.com/abstract=4556959 or http://dx.doi.org/10.2139/ssrn.4556959

[33] Coode, George (1852) “On Legislative Expression: or, the Language of the written law”, Turpin & Ridgway. Also reproduced in Driedger, Elmer (1976) “The composition of legislation; legislative forms and precedents” Canadian Department of Justice

computational law has helped us become clearer about how we use our basic building blocks of "must", "must not" and "may" (along with "means" and "includes" in definitions).

10.18. Since Coode a theme of the training of legislative drafters has been that the main point of legislation is to impose obligations and prohibitions (the common law approach is that permissions do not generally need to be granted by the state, in that natural persons may do or not do anything unless the law prohibits it or obliges them to do it). The structure of the classic "normative" legislative sentence is then that a person (natural or legal) must or must not do (or refrain from doing) a particular act, if certain conditions are met. Other provisions, such as definitions, are "constitutive" in that they build up to the normative provisions, usually by expanding on the meaning of the conditions or by adding exceptions to the conditions (and one type of exception is to restore the person to the position that they may do or not do the act, as they choose). Unfortunately drafters often use "must" and "may" in constitutive provisions, even though they are more naturally normative. Often the constitutive nature is hinted at by the fact that the drafter departs from the usual preference for active voice over passive voice, or by the subject not being a person. An example would be "An enforcement notice must contain information as to the right to appeal, and may contain additional steps that must be taken", where the "must" and "may" are really just constitutive conditions for the validity of the notice, without which it simply fails to have the legal effect it would otherwise attract (see discussion of "powers" below).

10.19. Meanwhile formal logic, and computing, have moved on since Coode. Formal logic[34] helps us recognise the need to distinguish between cases where we are using "if" to mean "if and only if" and those where we mean "if but not necessarily only if". If we do not spell out the meaning, then we need to be satisfied that our readers are going to see the context as clear enough to imply the correct one. Computing puts this as "if-then-else" – reminding us of the need to check we are clear about what is meant to be the position where the conditions we have set out are not met (the "else" part). This is particularly important given that almost all legislative provisions amount to "if these conditions are met, then this is the legal effect", whether they are expressly couched as using "if" or not. Even with "must" itself, there is an implied follow-on provision – "if a person must do an act, and the person does not do the act, then the person counts as in breach of this provision" – again we need to be clear enough about "what if not" (for a real "must" there needs to be a consequence of breach).

10.20. A branch of formal logic, called "deontic" logic, deals with obligation, prohibition and permission, and helps expose for us some of the oddities of how English deals with negation in "must" and "may" constructions. When we say "if a person drives, the person must wear a seatbelt", and we check the effect where the person does not drive (so the condition is not met), the answer is not that the person "must not" wear a seatbelt. Instead it is that it is not the case (at least as a result of this provision) that the person must wear a seatbelt. But we cannot express that simply in English with "must" and "not" (we cannot say "the person *not must* wear a seatbelt"), so we have to resort to expressing it in terms like "the person is not obliged to wear a seatbelt", which sits badly with our need to keep consistency of terms (we would now always say "must wear" instead of "is obliged to wear").

## *4. Ensuring "may" works properly*

10.21. There is a similar problem with "**may**", which we use to mean "may do or not do" (so the person is neither obliged nor prohibited) although we generally leave the "or not do" implied. If "may" worked like "must" the permission to refrain from doing something would be "the person *may not* do the act". But in English we tend to avoid "may not" because it does not naturally mean "is permitted not to" and instead is read as "must not" (which we would say if we meant it).

10.22. In 1913 Hohfeld[35] was concerned that some of the concepts we render with "may" were being confused in judgments and commentary. He wanted to distinguish a "right" from a "power" from a "privilege", and he did so in terms of relationships between two people. For him, if I have a duty to you to do X,

[34] For more detail see Waddington, M. (2021) "The basics of symbolic formal logic as a useful tool for legislative counsel" *The Loophole*, *Issue 2 of 2021*, 56-78 https://calc.ngo/publications/loopholes

[35] Hohfeld, W. N. (1913) "Some Fundamental Legal Conceptions as Applied in Judicial Reasoning" *Yale Law Journal,* 23(1), pp. 16-59. But see also his confusingly named second article – Hohfeld, W. N. (1917) "Fundamental Legal Conceptions as Applied in Judicial Reasoning" *Yale Law Journal,* 26(8), pp. 710-770.

then you have corresponding "right" against me for me to do X. If I have no duty to you to do X, then I have a "privilege" (of not doing X) as against you. A "power" is more complicated – for him, I have a power as against you when my choice to do X will change your legal position (often by imposing a duty on you), and correspondingly you have a "liability" in respect of me doing X (in that you cannot control whether your legal position changes).

10.23. Sartor[36] is the leading modern analyst of these senses of "may" for law, logic and computing. He clarifies some of the differences between constitutive and normative provisions, but on Hohfeld and "may" he complicates things by adding more of the parties who can be affected by each other in different ways. As modern drafters we would not risk building up such complex structures by relying on just one word to capture the roles of multiple parties, especially a word as fraught as "right" or "power". So instead we keep things on a surer footing by spelling out who must or may do what – so for a power that means we are just saying what one person "may" do, and then saying what someone else "must" or "must not" do if the first person does what they may do.

10.24. That means that for our purposes there are two main normative uses of "may" in legislation, though both can apply at once to the same instance of "may". The drafter's "what if/else" question needs to be "will my reader be clear enough as to what the effect is if the person *does* do what they may do?".

- One use is as **permission**, which effectively operates as an **exception** to a prohibition (or obligation) imposed on the permission-holder by some other provision (not necessarily statutory). There the drafter needs to be satisfied that the reader can clearly enough see what that linked prohibition is. An example would be –

  "(1) A driver must not exceed the speed limit.

  …

  (6) *Despite paragraph (1)*, a police officer may exceed the speed limit if driving … ."

- The other is as a **power**, which effectively operates as a **trigger** to impose a prohibition (or obligation) on someone else (or possibly on the power-holder) under some other provision (again not necessarily statutory). Again the drafter needs to be satisfied that the reader can clearly enough see what that other linked prohibition or obligation is. An example would be –

  "(1) An applicant may request a review by the Minister of a decision under section 7.

  …

  (6) *If an applicant makes a request under paragraph (1)*, the Minister must consider the request and … ."

10.25. None of this is to say that the linked provisions have to be expressly set out in the same legislation, or in any other.

- In a "must" provision the link is to the consequences of breaching that provision. Sometimes that will be that the person commits a criminal offence or suffers some other sanction set out in the legislation. But often the subject of the "must" is a public body and the consequences are left unstated because the drafter is confident that it is clear enough that general administrative law will do whatever is appropriate about any breach (typically the public body will be liable to Judicial Review) and the drafter does not want to interfere with that (especially in a country where administrative law has not been codified into statute, like England).

[36] Sartor, G. (2006) "Fundamental legal concepts: A formal and teleological characterisation." *Artificial Intelligence and Law*, 14 (1-2): 101-142. See also Guido Governatori, Antonino Rotolo, and Giovanni Sartor (2021) "Logic and the law: philosophical foundations, deontics, and defeasible reasoning" in Dov M. Gabbay, John Horty, Xavier Parent, Ron van der Meyden, and Leon van der Torre, editors "*Handbook of Deontic Logic and Normative Reasoning, volume 2*", chapter 9, p655–760 (available at https://www.researchgate.net/publication/353013213_Logic_and_the_Law_Philosophical_Foundations_Deontics_and_Defeasible_Reasoning)

- In a “may” provision the link could equally be to something in common law that the drafter does not want to interfere with. For instance the power example could have been “An applicant may appeal to the court against a decision …”, and in that case there might not be an express provision saying the court must entertain a correctly made appeal, because that is adequately implied. The point is just that the drafter, if leaving the link implied, needs to be confident that readers of the legislation will be sufficiently clear in seeing the same link as the drafter.

10.26. I discuss how our Computer-Readable Legislation Project has dealt with parsing and marking up some of these logical points in an article for the Loophole.[37] There is obvious scope for a computer to help guide the drafter through these logical paths, in that computers find logic much easier to follow reliably than we do.

## *5. Computer-readability for existing statute books*

10.27. Although legislative drafters naturally focus on what can be done with the new legislation that they draft, there is still the question of what to do with the vast majority of any jurisdiction’s statute book, which was drafted long ago and inherited in a form that is not machine-friendly. As mentioned below, “machine learning” can be applied to a statute book by treating it as a data set. That works better if the data is structured, but existing statute books are unlikely to be structured in any way that relates to the meaning of the legislation. Unstructured data sets can still yield useful results from a machine learning approach, by finding patterns in the statute book that provide useful pointers. But there are risks if that is not done carefully, particularly if unreliable assumptions are made about what certain kinds of data would show.[38] For example it should not be assumed –

- that older legislation that is still in force is out of date (its longevity might be due to its continued usefulness, and it might have been amended many times);
- that reference to older technology (such as telegrams or faxes) always mean the legislation is out of date (the legislation may have been amended to leave the old reference in place but add something like “or any other transmission of the information by electronic means or otherwise”, ensuring likely future technology is covered);
- that counting instances of “must” or “must not”, or numbers of offences, is a reliable guide to weighing the regulatory burden imposed by a piece of legislation (complex offences can be drafted as one offence or several, and similarly “must” can be used once or multiple times in a complex obligation provision);
- that the number of cross-references is a guaranteed measure of the complexity of the legislation or of duplication (cross-references generally reduce duplication, some make the legislation simpler, and many of the most important cross-references are merely implied, especially given there will be an Interpretation Act which applies without being expressly referenced).

10.28. One of the things that machine learning is good at is finding patterns in existing legislation, and doing so across different countries. In biosecurity there is a need for different countries to implement the same regimes to enable trade on common terms. This is similar to the longstanding issue of EU member states needing to implement Directives, and the EU Commission needing to ensure they do so properly, where the EU has been turning to technology to help with the process through the “THEMIS” system and increased interoperability.[39] In the field of financial services there is a demand from firms for information about how different countries regulate similar financial services. The

[37] Waddington, M. (2023). Jersey’s project on parsing drafts for if-then structures for “Rules as Code”. *The Loophole*, *Issue 2 of 2023*, 2-41. https://calc.ngo/publications/loopholes

[38] For more detail see Waddington, M (2022) "*The statute data swamp: NLP, data cleansing, data interpretation and legislative drafting*", conference paper at “Computational Legal Studies: Past, Present, and Future”, Singapore Management University Centre for Computational Law, 2-4 March 2022, online at https://www.researchgate.net/publication/377762067_The_statute_data_swamp_NLP_data_cleansing_data_interpretation_and_legislative_drafting

[39] See https://joinup.ec.europa.eu/interoperable-europe/themis

“Regulatory Genome Project”[40] is an initiative by Cambridge University using machine learning to create an open standard framework for classifying regulatory content from different countries.

10.29. Nevertheless, a problem remains when it comes to capturing the logical structure of existing legislation in the way discussed above for draft legislation. That is that, whenever you try this exercise you will spot a logical error in a piece of legislation. In draft legislation the point is that you are spotting the problem before the legislation is enacted, so the drafter can fix the problem before it is too late. But once legislation is enacted, the problem cannot be fixed without enacting new legislation to amend the existing legislation (and politicians are not generally[41] excited about steering mere tidying provisions through the legislative process). Otherwise you just have to wait for a court to rule on which interpretation it will apply (or even whether it will decide the provision has no effect at all), and you cannot fix the problem by coding it in the way the government would prefer. LegalRuleML can be used to encode alternative interpretations of the provision,[42] but that does not really solve the problem.

## *6. Using Artificial Intelligence to help with drafting and understanding legislation*

10.30. Artificial Intelligence has captured people’s imagination and attention since ChatGPT was launched in late 2022, and there are likely to be continuing developments in finding uses in law for chatbots, generative AI and large language models (“LLMs”). The main issue is that, like predictive text, generative AI will guess an answer (or “hallucinate” as it has come to be known), whereas in the legal context we normally need clarity about degrees of certainty or doubt (and generative AI’s apparent confidence, and habit of making up sources, can give a misleading aura of truth). There is still a great deal of hype about AI, and a temptation to assume all legal technology should include some AI as the latest magic ingredient, but it also seems extremely likely that generative AI will become more reliable and will be applied in ways that are useful and safe. Meanwhile those predicting doom clash with those hailing AI as a cure-all in law.[43] But problems with over-reliance on technology in law pre-date generative AI (see above on “robo-debt”). Meanwhile the EU has turned its attention to the various ways in which modern Artificial Intelligence might be able to help legislative drafters and those who need to understand legislation.[44]

10.31. It is beyond the scope of this chapter to go into detail about how AI works and what the different types are, but it is worth mentioning some distinctions at this stage. The term “AI” goes back decades, as does interest in applying AI to law,[45] but “AI” is now mainly being used for the latest “generative” AI that employs LLMs to run chatbots. Generative AI is a growth out of “machine learning”, which has

[40] https://www.regulatorygenome.org

[41] One way to generate enthusiasm, in some countries, might be that, if new legislation in a field has a digital version, then it could be attractive to re-enact related existing legislation in that field. The idea would be to fix any logical problems but also to be able to create a digital version of that legislation too, gradually making the digital versions more useful by rendering their coverage more comprehensive in a given field.

[42] Athan, T. et al. (2014) “Legal Interpretations in LegalRuleML”, in *SW4LAW+DC@JURIX2014* http://ceur-ws.org/Vol-1296/paper2.pdf

[43] An example of the pro-camp is Aidid, Abdi and Alarie, Benjamin (2023) "*The Legal Singularity: How Artificial Intelligence Can Make Law Radically Better*", University of Toronto Press. An example of the anti-camp is Deakin, S., Markou, C. (eds.) (2020) "*Is Law Computable? Critical Perspectives on Law and Artificial Intelligence*", Hart Publishing. See also the notes on COHUBICOL in section 2 above.

[44] Palmirani, M., Vitali, F., Van Puymbroeck, W., and Nubla Durango, F. (2022) “*Legal Drafting in the Era of Artificial Intelligence and Digitisation*”, European Commission https://joinup.ec.europa.eu/sites/default/files/document/2022-06/Drafting%20legislation%20in%20the%20era%20of%20AI%20and%20digitisation%20%E2%80%93%20study.pdf https://joinup.ec.europa.eu/collection/justice-law-and-security/solution/leos-open-source-software-editing-legislation/document/drafting-legislation-era-ai-and-digitisation

[45] See the special issue of the journal "*Artificial Intelligence and Law*", Volume 30, Issue 4, December 2022 on "Thirty Years of Artificial Intelligence and Law" https://link.springer.com/journal/10506/volumes-and-issues/30-4. See also the annual records of ICAIL (the International Conference on AI and Law) at https://dl.acm.org/conference/icail, Jurix (the International Conference on Legal Knowledge and Information Systems) at https://dblp.org/db/conf/jurix/index.html and other groups.

been used for several years already[46] by researchers such as Daniel Katz to analyse patterns in large data-sets, including court judgments and statute books, to measure complexity[47] or detect potential problems[48] and generally to treat collections of legal texts as data.

10.32. But even before that, the label "AI" was being used for what were known as "expert systems",[49] using reliable rule-based inferencing software (able to work out answers, but also questions, with "backward and forward chaining"). Those systems are often now not treated as being AI at all, but are also known fondly as "good old-fashioned AI" (or "GOFAI"), or as "classical", "symbolic" or "rules-based" AI, especially when attempts are made to use them in combination with modern generative AI to try to get the best of both worlds (particularly the reliability of GOFAI, with the user-friendliness of generative AI). The GOFAI approach also combines well with knowledge bases, knowledge graphs and ontologies, which are among the other ways in which attempts are being made to render generative AI more reliable (alongside other ways of handling the generative element, such as "RAG" or "retrieval augmented generation").

10.33. Approaches like RAG are attractive because, as well as hallucinating, generative AI does currently have a problem with reasoning. It is predicting the most likely next text, rather than computing an answer in the way we traditionally think of computers working, so it is not surprising that reasoning would be a weak point. This is a severe limitation on its ability to produce accurate answers about applying legislation to facts, because that task typically involves needing to jump between different provisions, read definitions into substantive provisions, and work out what happens if conditions in the provision are not met. For example, with a provision that says "a person driving on a public road must wear a seatbelt", the system needs to find and apply a definition of "public road" from another provision, and be able to give a safe answer if the facts do not meet that definition (even if that answer is just "this provision does not determine either way whether the person on that road must wear a seatbelt"). Much of the current research[50] in this area is into how generative AI might be combined with rules-based systems (to produce "neurosymbolic AI"[51]) in a way that tackles generative AI's problems of hallucinating and being unable to reason. This does look promising, particularly when combined with work to improve the reasoning abilities of LLMs, such as in Google's Gemini Ultra.[52] But even if the reasoning element still needs to be done by a rules-based logical system, there is still plenty of scope for generative AI to help. In particular the rules-based systems work best when the user's questions happen to match what the legislation is about. A generative AI chatbot could take a user's natural language description of their issue and help the user find the right legislation to which

[46] See for example, from as far back as 2009, Michael J. Bommarito, Daniel Katz, and Jon Zelner (2009) "Law as a seamless web? comparison of various network representations of the United States Supreme Court corpus (1791-2005)" in *Proceedings of the 12th International Conference on Artificial Intelligence and Law (ICAIL '09)*, Association for Computing Machinery, 234–235 https://dl.acm.org/doi/abs/10.1145/1568234.1568270

[47] Katz, D.M., Bommarito, M.J. (2014) "Measuring the complexity of the law: the United States Code" *Artificial Intelligence and Law* 22, 337–374 https://link.springer.com/article/10.1007/s10506-014-9160-8

[48] Corinna Coupette, Dirk Hartung, Janis Beckedorf, Maximilian Bother & Daniel Martin Katz (2022) "Law Smells: Defining and Detecting Problematic Patterns in Legal Drafting" *Journal of Artificial Intelligence & Law* 31, 335–368 https://link.springer.com/article/10.1007/s10506-022-09315-w

[49] For the history of a current system that has its roots in the expert systems of the 1980s, see Greenleaf, Graham and Mowbray, Andrew and Chung, Philip (2018) "*The Datalex Project: History and Bibliography*" UNSW Law Research Paper No. 18-4, https://ssrn.com/abstract=3095897

[50] See for instance Morris, Jason (2023) "GPT4 and Rules as Code", available online at https://gauntlet173.github.io/post/2023_04_12_llm_and_rac/

[51] See the Alan Turing Institute's introduction https://www.turing.ac.uk/research/interest-groups/neuro-symbolic-ai

[52] Google claim that Gemini marks a significant improvement in generative AI's performance at reasoning, including "mathematical reasoning" and using "its reasoning capabilities to think more carefully before answering difficult questions" - https://blog.google/technology/ai/google-gemini-ai/#performance. The author works, at the Legislative Drafting Office of the States of Jersey, on the Computer-Readable Legislation Project which (at the time of writing) is hoping to use Google Cloud's AI on Jersey's statute book, to see what can be done to deal with the issue of hallucination.

it would then apply the rules-based system. The chatbot might then be safe to explain the rules-based system's results to the user and suggest follow-up questions (again passing them on to the rules-based system). But also an LLM system could help humans to parse the existing body of legislation, and help drafters to parse what they were drafting as they went along (including suggesting different structures for the same logical effect), so that the parsed legislation could be fed to the rules-based system (or possibly direct to a chatbot to improve its performance).

## *Conclusion*

10.34. The rapid pace of change in technology, and in its application to law and legislation, means that it is impossible to be certain about what will have happened in this field for even just a few years into the future. What we can be more confident of, though, is that the field will develop quickly and that those involved in drafting new legislation (and settling the policy behind it) will need to adapt how they work to meet the challenges that these developments will bring.

10.35. It is already the case that most lay people will use a search engine (typically Google at the time of writing) if they want to find a piece of legislation, rather than the specialised search facilities offered by the official sites for that country's legislation. In the future people are increasing likely to use generative AI chatbots to get answers to questions about their statutory rights or liabilities. Legislative drafters will need to consider that as part of their thinking on how they make their drafts understandable to those who are going to use the resulting legislation (similar to the way that "plain English" was adopted as a means of helping humans who read text versions of legislation). That might mean ensuring we can produce reliable coded versions of our drafts, or that we draft in ways that do not unnecessarily confuse AI. We will increasingly need to think of computers, and not just humans, as part of the intended readership of what we draft – as Pia Andrews puts it "machines are users too".[53]

## Further reading

Mohun, J. & Roberts, A. (2020). Cracking the code: Rulemaking for humans and machines. *OECD Working Papers on Public Governance*, No 42, https://doi.org/10.1787/3afe6ba5-en

Morris, J., 2020. *Spreadsheets for Legal Reasoning: The Continued Promise of Declarative Logic Programming in Law*. [Online] Available at: https://ssrn.com/abstract=3577239

"*Rules as Code Handbook wiki*". [Online] Available at: https://github.com/Rules-as-Code-League/RaC-Handbook/wiki/

Waddington, M. (2020). Research Note - Rules as Code. *Law in Context*, 37(1):179-86. Available from: https://journals.latrobe.edu.au/index.php/law-in-context/article/view/134

Waddington, M. (2021). The basics of symbolic formal logic as a useful tool for legislative counsel. *The Loophole*, *Issue 2 of 2021*, 56-78. https://calc.ngo/publications/loopholes

Waddington, M. (2023). Jersey's project on parsing drafts for if-then structures for "Rules as Code". *The Loophole*, *Issue 2 of 2023*, 2-41. https://calc.ngo/publications/loopholes

[53] See for instance, her presentation at https://drive.google.com/file/d/1HdQjTjhHGAirnOMrdFzawMl0zQp8NIOz/view, referenced in Kennan, Ariel and Soka, Sara (2022) "Benefit Eligibility Rules as Code: Reducing the Gap Between Policy and Service Delivery for the Safety Net", Beeck Center for Social Impact and Innovation, Georgetown University, available at https://beeckcenter.georgetown.edu/report/benefit-eligibility-rules-as-code/

# *Appendix*

## *Example – for chapter 10 "Digitising legislation: progress and prospects"*

This example is to show readers how to construct a spreadsheet in Excel (or any equivalent with an "if" function) that gives answers about a legal provision based solely on the text of the provision and the logical structure embedded in it, as mentioned in paragraph 10.10 of Chapter [10].

Imagine a provision of biosecurity legislation that is slightly more complicated than the seat-belt example in paragraph 10.9 – in section 7(1) of an imaginary Biosecurity Act –

"(1) Where the biosecurity risk has changed for a permit and the Director believes the change justifies revocation, the Director may revoke the permit."

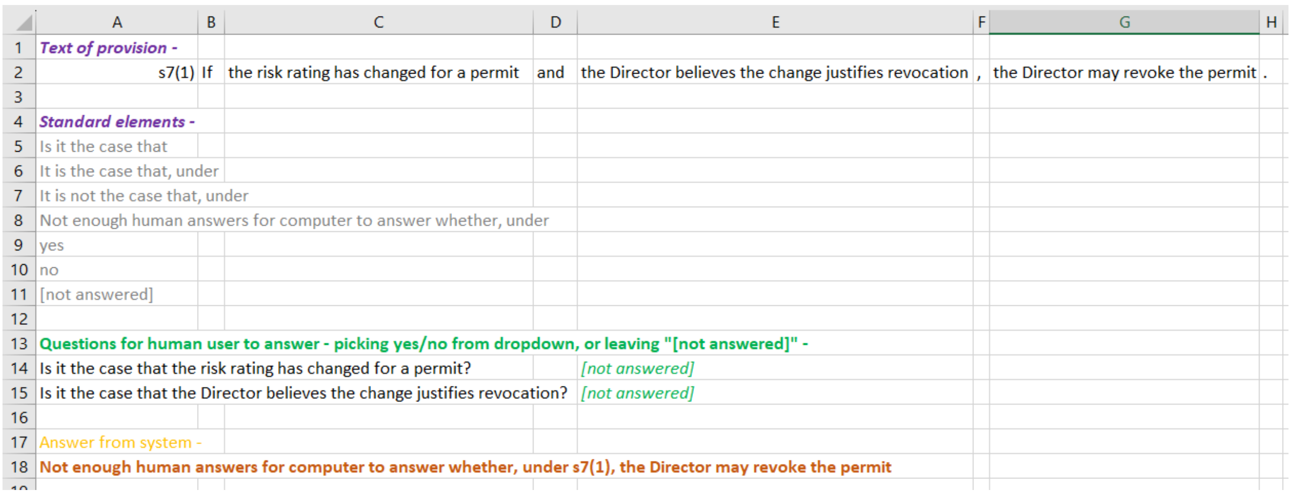
The "where" is equivalent to "if", and there is an "and" joining two conditions, and the effect is stated after the comma (without using "then", which drafters typically avoid because it could be misinterpreted as relating to time). So the logical structure of this provision can be represented as –

"If A and B, then C".

"Pseudo-code" means something that is not actual code (it cannot be run on a computer), but is an arbitrary representation of a bridge between the natural English and a coded version. If we follow the logical structure above, then we could represent the provision in an invented pseudo-code as –

"IF ((the biosecurity risk has changed for a permit) AND (the Director believes the change justifies revocation)) THEN (the Director may revoke the permit)"

We can then represent this with Excel. The Computer-Readable Legislation Project at Jersey's Legislative Drafting Office has set up Excel spreadsheets for legislation that it has parsed for if-then structures. All of the following details are drawn from their spreadsheet "IfThenWithTwoConditions.xlsx" available at https://osf.io/5j63s (which is published by their project under the Creative Commons Attribution 4.0 International Public License). The spreadsheet illustrates that, if we carve up the text of the provision into blocks of text in adjacent cells of a spreadsheet, we can use the names of those cells to refer to the blocks of text. If we then add some standard logical elements (such as "Is it the case that" and "[not answered]"), each in one cell, we can call up those elements too by referring to their cells. We can then use the cell references for the blocks of text to produce questions for a human user, and use the cell references for the logical elements to offer that user a choice of "yes" or "no" in a dropdown list in a cell corresponding to each question. We can then feed the results in those answer cells into an Excel "if" formula, so that the spreadsheet is completed with the correct answer (about whether the Director may revoke the permit under this provision) automatically produced from the answers picked by the human user. Here is an image of what the spreadsheet will look like if we enter the text of this example (but with "risk rating" instead of "biosecurity risk" to illustrate that any text can be used in the main cells, and with "If" instead of "Where" to illustrate the different ways of expressing the same logical elements) –



| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | ***Text of provision -*** | | | | | | | |
| 2 | s7(1) | If | the risk rating has changed for a permit | and | the Director believes the change justifies revocation | , | the Director may revoke the permit | . |
| 3 | | | | | | | | |
| 4 | ***Standard elements -*** | | | | | | | |
| 5 | Is it the case that | | | | | | | |
| 6 | It is the case that, under | | | | | | | |
| 7 | It is not the case that, under | | | | | | | |
| 8 | Not enough human answers for computer to answer whether, under | | | | | | | |
| 9 | yes | | | | | | | |
| 10 | no | | | | | | | |
| 11 | [not answered] | | | | | | | |
| 12 | | | | | | | | |
| 13 | **Questions for human user to answer - picking yes/no from dropdown, or leaving "[not answered]" -** | | | | | | | |
| 14 | Is it the case that the risk rating has changed for a permit? | | | | *[not answered]* | | | |
| 15 | Is it the case that the Director believes the change justifies revocation? | | | | *[not answered]* | | | |
| 16 | | | | | | | | |
| 17 | Answer from system - | | | | | | | |
| 18 | **Not enough human answers for computer to answer whether, under s7(1), the Director may revoke the permit** | | | | | | | |

In this version cells E14 and E15 have been set up as dropdown lists using the options in cells A9 to A11. In this image, the dropdowns are at their starting point of "[not answered]".

The Excel formula that generates the first question for the human user, in cell A14 is –

```
=CONCAT(A5,C2,"?")
```

For the second question, in cell A15, the formula is –

```
=CONCAT(A5,E2,"?")
```

The most important part is the formula in cell A18, which captures the “if” and “and” logic to generate the correct answer depending on whether the user has picked yes or no in cells E14 and E15. There are different ways to couch that formula, but the one that is most faithful to the text of the legislation (subject to the point about “not answered” below) would be this one, taken from the Computer-Readable Legislation Project’s example mentioned above –

```
=IF(AND(E14=A9,E15=A9), CONCAT(A6,A2,", ",G2), IF(OR(E14=A11,E15=A11), CONCAT(A8,A2,", ",G2), CONCAT(A7,A2,", ",G2)))
```

That is, in a real sense, an encoding of the “if-then-else” (and “and-or”) logical structure of our imaginary legislative provision. Although (of course) it is not elegant and it is not powerful enough to work on a greater scale, it demonstrates the point that the conditions and effects can be replaced by any other statement that has a “yes” or “no” value, and that therefore all that the encoding is doing is to capture the logical structure and not the meaning of the conditions or effects (whose wording can be preserved precisely as it is in the original legislation).

Readers will notice that in fact this example is slightly more sophisticated than a straightforward “yes” or “no”, in that the logic has been extended to cater for the position before the user has answered a question. It does so by adding a third option as “[not answered]” (in cell A11) as a starting position in the dropdown list, alongside the options for “yes” and “no”. That allows the spreadsheet to start off without assuming any of the answers (also, if the conditions had instead been joined by “or”, this would allow the user to answer just one question with “yes” to obtain a definite result, even though the other question remained unanswered). The logic is more complicated with this option, because there are more permutations of possible answers from the user and there is an additional possible result from the computer (shown in the image above at cell A18, using the standard logical element from cell A8 – “Not enough information to answer whether, under”). The pseudocode with the “not answered” element (as in the “if” formula above) could be something like –

```
IF ((ANSWERED-TRUE(the biosecurity risk has changed for a permit) AND
ANSWERED-TRUE (the Director believes the change justifies revocation))
THEN RETURN-TRUE(the Director may revoke the permit)
    ELSE IF ((UNANSWERED(the biosecurity risk has changed for a permit)
    OR UNANSWERED (the Director believes the change justifies revocation))
    THEN RETURN-UNANSWERABLE(the Director may revoke the permit)
        ELSE RETURN-FALSE(the Director may revoke the permit)
```

The point of including the reference to the provision – as in “It is not the case that*, under s7(1),*” – is to avoid the mistake of saying baldly “the Director may *not* revoke the permit”. That would be a mistake because we do not know whether there is another revocation power elsewhere (and “may not” is ambiguous between “must not” and “is permitted not to”). That means we avoid needing to worry about the issue of the distinction between “if-but-not-necessarily-only-if” and “if-and-only-if”, because the answer does not depend on that, as it is just about what s7(1) does. Again it is not ruling out whether there could be some other obligation/ prohibition/ exception about revoking permits in some other provision of this legislation (or in a contract, or common law, and so on).

As this is a spreadsheet, just reproducing the content of particular cells when required, you can also renumber the provision or enter any other appropriate text in cells C2, E2 and G2, for the two conditions and the result, and the spreadsheet will still work to give the correct answers. That reflects the fact that it is based on formal logic – it is the logical structure that is being captured, not the meaning of the contents of the conditions or the effect.