The basics of symbolic formal logic as a useful tool for legislative counsel

Matthew Waddington[1]

Abstract

This article offers an introduction to symbolic formal logic and its application to legislative drafting. A familiarity with the basics of symbolic formal logic can reinforce our common sense logic checking of drafts, and could in future help us to use computerised checking. The article introduces the logical connectors "not", "and", "or" and "if", which offer safeguards against ambiguities and mistakes in the use of those terms in drafting. The article then shows how the notation and concepts of symbolic formal logic cast further light on our use of "must" and "is" (after the banishment of "shall"), along with "must not", the weak and strong versions of "may", and the treacherous "may not".

Table of Contents

[1] Senior Legislative Drafter, States of Jersey. This article expands on part of the paper I gave at CALC's conference in Zambia in 2019. The original inspiration for this article was the annual "Law and Logic" summer school run by the European University Institute.

Introduction – what is symbolic formal logic and how is it useful in drafting?

Legislative counsel use many skills, not just our specialist legal knowledge.

- We need a grasp of grammar and how to parse a sentence. Some of us back that up with knowledge of other languages or of linguistics.
- We also need common sense logic. Equally some of us back that up with knowledge of philosophical logic, mathematical logic, computer logic or several varieties of logic.

It is not essential for legislative counsel to know other languages or to understand formal logic, but both of them can help in our work. Some legislative counsel (including me) did not hate algebra at school, but have not embarked on learning computer programming.[2] For us, the basic[3] notation of formal symbolic logic can be a useful tool in its own right to help sustain and reinforce our rigorous approach by supporting our common sense logic. That is not to suggest that all of us should take up formal logic, or that the full power of formal logic can be applied to legislative drafting, only that it may be a useful tool for some people for some aspects of drafting.

Traditional logic relies on statements being true or false, with nothing in between, and on reducing arguments to their core propositions (organised as premises leading to conclusions). Common law practitioners often contrast this with our human, non-binary approach and the flexibility of the common law (or casuistry, as civil lawyers may see it).[4] Legislative counsel in the Commonwealth typically work with the common law, but our aim is to produce internally coherent statutory rules which can be applied broadly. That involves a binary element in ensuring there is someone who has to make a decision on whether or not a person is guilty, an applicant is fit or an action is reasonable, however human and vague those questions are.

What is logic and what is it for?

We all know that logic is something that, as legislative counsel, we already use to check the structure and consistency of our drafts. It is no good a draft saying somebody has to do something, if somewhere else it says they have to refrain from doing it (unless the drafter

[2] Legislative counsel with coding skills are already diving into producing programs that computerise legislation. Some might see formal logic as a distraction and argue we should learn to code instead. But even the simplest formal logic can help us understand how programmers are working with legislation, particularly for "Rules as Code" (see Loophole June 2019 and https://oecd-opsi.org/projects/rulesascode/).

[3] Formal logic is a very rich topic. This article sticks firmly to the shallow end. For more see Hage (2016) "Elementary Logic for Lawyers" http://www.jaaphage.nl/pdf/ElementaryLogicForLawyers.pdf and Sartor (2006) "Fundamental legal concepts: A formal and teleological characterisation" (2008), 14(1-2) *Artificial Intelligence and Law* and at DEON 2008.

[4] O W Holmes Jr famously said "The life of the law has not been logic; it has been experience" in *The Common Law* (Little, Brown & Co: Boston, 1881). But see also Lovevinger, Lee) "An Introduction to Legal Logic" (1952, 27(4) *Indiana Law Journal* and S. Haack, "On Logic in the Law: 'Something, But Not All'" (2007), 20(1) *Ratio Juris*.

has reconciled the two by making one an exception to the other, setting out different circumstances in which each applies, or in some other way).

But what is there beyond this common sense logic? The starting point is classical propositional logic, which deals in propositions that can only be true or false and looks at the formal structure of the relationships between propositions that produce valid arguments by leading from premises to conclusions. The classic example is –

> If Socrates is a man, then Socrates is mortal;
>
> Socrates is a man;
>
> therefore Socrates is mortal.

A legal equivalent is –

> If the force used by the defendant was reasonable, then the defendant is not guilty;
>
> the force used by the defendant was reasonable;
>
> therefore the defendant is not guilty.

Symbolic formal logic uses notation to bring out the abstract form of these relationships by stripping out the content of each particular proposition and substituting a place-holder that can stand for any proposition (as long as the substituted proposition is still one that has to be true or false). So the structure of the Socrates argument works for any pair of propositions, including reasonable force and guilt. If that pair is represented by P and Q (in each case couched as “it is true that …”), then the argument can be represented as –

> If P, then Q;
>
> P;
>
> therefore Q.

That form of an argument is evidently valid. By contrast, one invalid argument would be –

> If P then Q;
>
> Q;
>
> therefore P.

That is because Q could be true without P, in that Socrates could be mortal without being a man, and some other defence could render the defendant not guilty.

These propositions can be formalised further using symbols for the relationships, so that “If P then Q” is rendered symbolically as “$P \rightarrow Q$”. But there is a different relationship where P always stands or falls together with Q, for which we need “if and only if” (abbreviated as “iff”), which is rendered symbolically as “$P \leftrightarrow Q$”.

But formal logic's symbolic notation, and its concepts of the relationships between propositions, can be used for purposes other than examining arguments for validity.[5] Legislative counsel could use it to check that we are being logically consistent in our drafts – not giving a circular definition, not simultaneously imposing an obligation and a prohibition on the same person in relation to the same activity, and so on. That can be done by drafters using the symbolic logic notation ourselves[6] or by developing a software program to run the logic checks and produce questions.[7]

Languages and logics

Languages can be divided into natural and non-natural and both can be parsed.

- **Natural languages** – Legislation is drafted in a "natural" language. In much of the Commonwealth, and in the USA, that language is English. In some jurisdictions legislation is drafted in two natural languages with equal status – English and French in Canada; English and Welsh in Wales. In the EU legislation can be drafted in multiple natural languages (and is made in 24 languages, in theory all with equal status).
- **Non-natural languages** – As well as natural languages, there are also non-natural languages. Computer languages are one example, which breaks down into sub-groups including machine languages, mark-up languages (such as HTML for websites and XML for structured documents) and programming languages (such as Python, C, Java). Like natural languages, computer languages each have their own grammar, with semantics (what the elements represent) and syntax (how the elements relate to each other). Logic is like a language or set of (families of) languages. Each "logical language" has its own syntax and semantics, with types and families of logics sharing elements of their syntax or semantics or both.[8]
- **Parsing** – Statements can be parsed, both in natural languages and in logics. Legislative counsel parse English sentences to work out what is going on and check that they are properly constructed. Parsing in a natural language involves (at some level) identifying not only which words are verbs, nouns, adjectives, adverbs, pronouns and so on, but also (more importantly for us) how the words (and phrases and clauses) relate to each other. Parsing is a useful tool in logic too (and in computer coding).

[5] Variants of symbolic logic notation are used in computer logic, to give instructions to a computer in an unambiguous way to ensure the right output for a given input.

[6] An early version of this suggestion was the quirky "systematic pulverization" system in EA Layman, "Symbolic Logic: A Razor-Edged Tool for Drafting and Interpreting Legal Documents" (University of Michigan Law School Scholarship Repository, 1957).

[7] Work on Rules as Code may produce such a tool. Ideally it would not be a "computer says no" nuisance, but a polite helper that says "Just checking – that looks like a logical inconsistency, I am probably missing something, but did you mean it. If so, is it worth seeing if you can recast it to remove the false impression of inconsistency, or is it all fine?".

[8] This article looks at basic notation, and sacrifices consistency to explanation. For a good example of how a more complex notation system can be built up, see Sartor, above n. 3.

Connectors – “not”, “and”, “or”, “if”

Connector symbols

After the variables (in this case the true/false propositions) have been abstracted into symbols, the next element to formalise is the connectors (often called “operators” in computer logic).

The argument from the previous example can be made slightly more complex by adding “and” –

If Socrates is a Greek, ***and*** Socrates is a philosopher, *then* Socrates is wise;

Socrates is a Greek ***and*** Socrates is a philosopher;

therefore Socrates is wise.

Again, that argument works for any trio of propositions, including legal propositions. Representing that trio by P, Q and R, the argument can be abstracted and formalised as

If P and Q, then R; P and Q; therefore R.

The other special connectors that can be used in these structures are “not” and “or” –

If Socrates is a philosopher, ***or*** Socrates is ***not*** a Persian, *then* Socrates is wise;

Socrates is ***not*** a Persian;

therefore Socrates is wise.

Using P, Q & R again, the structure of that argument is

If P or not Q, then R; not Q; therefore R.

Unfortunately, the natural language “or” and “if” turn out to be ambiguous in ways that haunt our drafting. The logical connectors have fixed meanings to help avoid that ambiguity.

~	not	Negation of a proposition that has to be either true or false. So there will be a negation, the negation of the negation will be the original, with no in between.
$\wedge$	and	Both have to be true. (“but” can be “and” with an overlay of “surprisingly”.)
$\vee$	or	One or other *or both*. (Separate from “exclusive or” as “one or other, *but not both*”.)
$\rightarrow$	if, then	“If X, then Y” – if X is true, Y has to be true. But all other combinations are allowed, including X false with Y true or false.

$\leftrightarrow$	if & only if	"If & only if X, then Y", "Iff X, Y" – if X is true then Y has to be true, but also if Y is true then X has to be true. So if either is false, they both have to be false.

Truth tables

Each of the logical connectors produces a new combined proposition that is itself true or false. That means the connectors can be mapped into tables showing how they preserve or alter the truth or falsity of the propositions fed into them. So, the combined proposition "P *and* Q" is true only when both P and Q are true, and not if either or both are false. But if instead what links P to Q is the "exclusive or", the combined proposition is true if either one of P or Q is true, but not if both or neither are true. These results can be set out in tables as a way of fully specifying the meanings of each of the connectives without ambiguity using "T" for true and "F" for false (though "true" and "false" are not defined).

Not (takes just one proposition)

P	~ P
T	F
F	T

And

P	Q	$P \wedge Q$
T	T	T
T	F	F
F	T	F
F	F	F

Inclusive or

One or the other or both

P	Q	$P \vee Q$
T	T	T
T	F	T
F	T	T
F	F	F

Exclusive or

One or the other, but not both/neither

P	Q	$P \oplus Q$
T	T	F
T	F	T
F	T	T
F	F	F

If (but not **only if)**

No P without Q

P	Q	$P \rightarrow Q$
T	T	T
T	F	F
F	T	**T**
F	F	**T**

If & only if – "iff"

Both or neither, but not either one without other

P	Q	$P \leftrightarrow Q$
T	T	T
T	F	F
F	T	F
F	F	T

Truth tables can be made for connectors producing every other permutation of the results in the third column (for connectors that link only two propositions). But it is not useful to do so, as most of these connectors can be reduced to each other, and the other versions are not commonly needed.[9]

Using connectors

Using the connectors, we can formalise whole arguments. Going back to

If P or Q, then R;

Q;

therefore R,

that can be represented as –

$(P \vee Q) \rightarrow R$;

Q;

therefore R.

If we look further back to

If P then Q;

P;

therefore Q,

we can symbolise that as

$P \rightarrow Q$;

P;

therefore Q.

[9] Some of the other permutations can appear in computing, where electrical engineers use "logic gates" for connectors, including "NOR", which is only true when both P and Q are false, and "NAND", which is only false when both P and Q are true.

We can apply this to one model of the law as, for example –

> If (the defendant worked without a licence) then (the defendant committed an offence);
>
> (the defendant worked without a licence);
>
> therefore (the defendant committed an offence).

In this model the first part is the rule drafted as legislation (and explained by the judge to the jury). The second is a finding of fact by the jury. The third is the conclusion about the legal position and is drawn from the application of the rule to the fact. In the rule, "P' is a statement of fact (which is true or false in a given case), "Q" is a statement of the legal status arising from that fact (again true or false in a given case) and "$P \rightarrow Q$" is a legal rule.

This is an over-simplification in that the rule needs to be expressed generally and the finding of fact needs to be particular (and is usually in the past) – "If *any* person work*s* without a licence, then *that* person commit*s* an offence; *this* person work*ed* without a licence; therefore, *this* person committ*ed* an offence". There are logical ways of representing that too, but they are beyond the scope of this introduction.[10]

"And", inclusive "or", exclusive "or" – semantic or syntactic ambiguity

As Commonwealth legislative counsel we are also careful with how we combine or separate items in lists. We use "and" and "or", and we use numbered paragraphs with different numbering styles and indenting at different levels, with only one connector (after the penultimate item) per level. That is because we recognise that "if a person walks and talks or claps" is ambiguous. But it is ambiguous in two different ways, syntactically and semantically.

Syntactic ambiguity – First there is a problem over the grouping and range.

- Does it mean the person needs to be clapping, or else both walking and talking, or does it mean the person needs to be walking, but also either talking or clapping?
- To distinguish in a complex draft, particularly where there are subordinate clauses in the elements, we use paragraphing, numbering and indenting conventions rather than parentheses (which many jurisdictions save for explanatory material).
- But in a simple case like this the legislative counsel is likely to use the natural language rewritten with commas and "both" or "either" ("if a person walks, and either talks or claps"), rather than go for the paragraphed version (unless there is going to be some need elsewhere in the draft to refer back to an element which is easier to identify by its number in the paragraphed version).
- Either way, the logic can be a neat way to sum up the relations and to check them.

[10] This aspect of legislative drafting conventions has been noted in a new English-based computer language "Logical English", being developed by Robert Kowalski – see the items gathered on his Research Gate page at https://www.researchgate.net/project/Logical-English.

- In mathematics there is a sequence in which operations should be taken to be intended when no further guidance is given.[11] But in algebra it is generally easier just to use parentheses to isolate elements, to determine the order in which you apply the addition, subtraction, multiplication and division. So (2 x 3) + 4 = 10, whereas 2 x (3 + 4) = 14. Brackets can be artificially added to natural language to mark the ranges of the "and" and "or".
- Logic uses brackets similarly, with "$\vee$" for "or" and "$\wedge$" for "and" –

  "if a person (***w***alks and ***t***alks) or ***c***laps, then …" = $((W \wedge T) \vee C) \rightarrow \ldots$

  "if a person ***w***alks and (***t***alks or ***c***laps), then …" = $(W \wedge (T \vee C)) \rightarrow \ldots$

Semantic ambiguity – There is also a problem with two meanings of "or".

- Is this example meant also to cover the case in which the person does all three activities, or not? How do we indicate which?
- The natural language "or" can be **inclusive** or **exclusive** – the inclusive is "X or Y, or both", while the exclusive is "X or Y, but not both". Legislative counsel tend to use "or" in the inclusive sense, but sometimes exclusive. But do we always check whether it is clear which one we are using, and whether we need to add "or both" or "but not both"?[12]
- Logicians use "$\vee$" for inclusive "or". For exclusive "or" logicians sometimes use "$\oplus$" (borrowed from computing), but they tend to prefer instead to spell out the exclusive form as "X or Y, but not both", so that it becomes "$(X \vee Y) \wedge \sim(X \wedge Y)$".
- There is a third possible type of "or", though not a normal English meaning, which can be rendered "neither or either, but not both". It is known as "*NAND*" (again borrowed from computing) and is "$\sim(X \wedge Y)$" – they cannot both be true, but they can both be false, or either of them can be true on its own.

[11] It is known in Britain as "BODMAS" for Brackets, Orders, Divide, Multiply, Add, then Subtract (with other names in other countries).

[12] A recent decision of the British Columbia Court of Appeal deals with an example where this was not checked: *R. v. Ghadban*, 2021 BCCA 69 (CanLII), https://canlii.ca/t/jd6nq

Here are renderings of “person walks and talks or claps” – firstly where clapping is an alternative to walking while talking, but with only the logic versions fully specifying that the “or” is inclusive.

Natural language rewritten with commas and “both”	… person claps, or both walks and talks
Modern Commonwealth legislative drafting style – using disciplined paragraphing, numbering and indenting, with one connector per level	(1) … person – (a) both – (i) walks, and (ii) talks; or (b) claps.
Natural language with brackets added	… person ((walks and talks) or claps)
Same with logical connector symbols	… person ((walks $\wedge$ talks) $\vee$ claps)
Fully formalised with W for “walks”, T for “talks”, C for “claps”	… $(W \wedge T) \vee C$

Here are renderings of “person walks and talks or claps” – where walking is required, but can be accompanied by either talking or clapping. Again only the logic versions fully specify that the “or” is inclusive.

Natural language rewritten with commas and “either”	… person walks, and either talks or claps
Modern Commonwealth legislative drafting style	(1) … person – (a) walks; and (b) either – (i) talks, or (ii) claps.
Natural language with brackets added	… person (walks and (talks or claps))
Same with logical connector symbols	… person (walks $\wedge$ (talks $\vee$ claps))
Fully formalised	… $W \wedge (T \vee C)$

Lastly, here are renderings that expressly tackle both the scope of "and"/"or" and the inclusive/exclusive "or", with different instructions resulting in four different drafts.

Instructions	***Legislative draft***	***Logic***
Clapping is an alternative to walking while talking, or all three can be done at once	(1) … person does either or both of – (a) walking and talking; (b) clapping.	$(W \wedge T) \vee C$
Clapping is an alternative to walking while talking, but all three cannot be done at once	(1) … person does either, but not both, of – (a) walking and talking; (b) clapping.	$(W \wedge T) \oplus C$
Walking is required, and needs to be accompanied by either talking or clapping or both	(1) … person – (a) walks; and (b) talks or claps, or does both.	$W \wedge (T \vee C)$
Walking is required, and needs to be accompanied by either talking or clapping (but there cannot be both of those two)	(1) … person – (a) walks; and (b) talks or claps, but does not do both.	$W \wedge (T \oplus C)$

At this point I hope many legislative counsel will appreciate the way the logical formulations avoid ambiguity while remaining concise and even elegant. I hope they will also extrapolate from these simplified examples to the more complex structures that we are often asked to draft.

"Not" – binary decisions and a drafting puzzle

As legislative counsel, we know we have to be careful with negations. "Not" is rendered in symbolic logic notation as "~" and placed before the negated element.[13] Formal logic works in a binary way (like a computer), so that for every scenario we can think of, either X is true of that scenario, or X is not true of it. The logical idea of negation is linked to the logical idea of a proposition, which is just a statement that has to be either true or false (unlike a question or command).[14] So "X" stands for "it is the case that X [is true]", whereas "~X" stands for "it is not the case that X [is true]". Imposing legal obligations works like that too

[13] Other "logics" may symbolise negation as "-", "!" or "¬". There are alternatives to the other symbols used here as well. Consistency inside one logical language matters, rather than the actual symbol.

[14] At first sight legislation might seem like commands, but it is better to see "A person who does X must do Y" as a statement of the law. Those statements are true within the correct jurisdiction, during the period the legislation is in force, and to the extent that some other law does not override them. At another level it is also true that a given person in a given situation either is or is not legally obliged by this rule to do Y.

in a sense. In the end somebody has to decide either that you complied or did not comply, with no half-way house. We try to draft to enable users to be clear about what does and does not count as compliance, but ultimately somebody has to make their mind up.

This article does not attempt to do more than scratch the surface of how logic works beyond simply formalising propositions. But an aspect worth mentioning in connection with negation is one of De Morgan's Laws[15] which can be formalised as –

not (X **or** Y) = not-X **and** not-Y (rather than "not-X ***or*** not-Y")

$\sim(X \vee Y) = \sim X \wedge \sim Y$

A drafter can be caught out by the need to flip from "or" to "and" when the range of the negation is changed. To see that in practice, consider section 8(10) of the *Human Transplantation (Wales) Act 2013* which says, with some paraphrasing –

(10) A person is **not** eligible to act under an appointment if the person –

(a) is **not** an adult, **or**

(b) **is** of a prescribed description.[16]

The idea was to prescribe descriptions of persons who do not have capacity. Another jurisdiction wanted to adapt the Welsh model. The initial double negative was considered unhelpful, particularly with the hidden negative in the lack of capacity. It was feasible to switch to a positive expression because this was the only eligibility rule. But an attempt to manipulate the concepts without help from De Morgan's laws led to overlooking the need to change "or" to "and", which resulted in:

(10) A person **is** eligible to act under an appointment if the person –

(a) **is** an adult, ***or***

(b) is **not** of a prescribed description.

When it came to drafting the subsidiary legislation to prescribe descriptions of persons who lacked capacity, the problem became apparent. If the prescribed description involves lack of capacity, then the result is that incapacitated adults are eligible, as are children with capacity, not just adults with capacity. Switching "or" to "and" gives the same result as the Welsh legislation, in that only adults with capacity are eligible. This could have been flagged by using symbolic logic (a computer can apply De Morgan's laws even if a drafter struggles with them).

[15] Named after Augustus De Morgan, a 19th-century counterpart of George Boole. Boole's work on logic is credited as significant to the development of computing (including for his use of 0 and 1 to stand for false and true, as forerunners of digital bits). Many non-coders know of "Boolean searches" (using logical connectors).

[16] 2013 (anaw 5)

I will come back to negation again below in relation to "must" to show how the notation can also help clarify what negation is doing in relation to different parts of a sentence dealing with obligations, prohibitions or permissions.

"If", "iff" – implicit use, and drafting clearly about "if not"

"If" is special because of its prominence across formal logic, computer logic and legislative drafting.

- In symbolic logic notation "If X then Y" can be rendered as "X→Y". But logical "if" is odd, in that it does not imply either causation or passage of time and when X is false, "X→Y" is true regardless of whether Y is true or false.17 But it can be useful for drafters to be reminded to check whether they are assuming either of those factors. Also logic distinguishes between "→" for "if, but not necessarily only if" and "↔" for "if and only if" (abbreviated as "iff"). Sometimes in a draft the difference will not carry any weight, but legislative counsel could benefit from routinely checking which version they mean and whether that is beyond doubt in the draft.
- The classic use of "if" in **computer logic** is for "if, else" instructions – "if this is true, do that; otherwise do this other action". In one sense legislation tells people what to do, but it is not set out as a sequence of instructions for performing a task legally, or as a description of a procedure.[18] Instead it sets out what the law is, leaving the users to apply it to their circumstances. Nevertheless, it may be useful for drafters to consider what is meant to happen if X is false, and whether that is clear enough in the draft. Often we do not spell out the "else" because the draft is not attempting to dictate the legal position in that case, and is leaving it to whatever was already the law.

In common law legislative drafting we want rules that can be applied to cases, so we tend to avoid statements of general principles as obligations and rely on "if" instead. Perhaps the fundamental model[19] is "If facts X and Y are the case, then Z is the legal consequence", where Z is a statement of the law rather than of facts – "If you killed someone, and you meant to, then that amounts to the offence of murder for which you are liable to be punished".

[17] See truth tables above p.63.

[18] "Rules as Code" has highlighted computing's distinction between "declarative" programs (such as Prolog) and "imperative" programs (such as Python). Imperative programs are much more common, and are embedded in the thinking of most programmers, because they tell a computer how to execute a program to perform a task. Declarative programs are not instructions to perform a task, and instead they tell the computer what are the conditions for something to be true, much like formal logic but also like legislation.

[19] Dating back to 1877 in Lord Thring, *Practical Legislation: The Composition and Language of Acts of Parliament and Business Documents* (George E. Eyre and William: Spottiswoode, 1877).

In symbolic logic “(X does P) $\leftrightarrow$ (X must do Q)” means there are no situations other than X doing P in which X must do Q. But “(X does P) $\rightarrow$ (X must do Q)” implies there might be situations other than X doing P in which X must nevertheless do Q.

In computer logic “if A, then B, else C” means “if A then B, and if not A then C”. This renders “iff X does P then X must do Q” as “if X does P, then X must do Q, else not(X must do Q)”. But “if X does P then X must do Q” might come out as “if X does P, then X must do Q, else X must do R” where having to do R might be compatible with (but not fully interchangeable with) not having to do Q.

In the common sense logic of drafting, legislative counsel are used to the idea that we need to ensure there is a clear express or implied sanction for non-compliance with a "must", without which it would not really be a “must”.

When we draft a provision with the form “if X does P, then X must do Q”, we know it must be clear what the result is if X does not do Q. Often that involves an express provision that X commits an offence. But if X is a public body we will commonly leave it to the principles of administrative law. In other cases there will be other sanctions or consequences expressed or implied.

Are we always as clear when it comes to the result if X does not do P in the first place? For example, if we provide that “if an inspector serves a pig hygiene notice, the farmer must clean the pig pen” are we always clear about whether the farmer need not clean the pig pen in the absence of a notice? If we have an additional more general provision that “a farmer must keep animal pens clean”, we will need to clarify the relationship between the two. Perhaps failing to clean the pig pen without a notice is a less serious offence than failing to clean it after a notice, or perhaps we really mean that pens for all other animals must always be kept clean, but pig pens only need to be cleaned when there is a notice. In other cases the consequence of the “if” condition not being met might be limited to just meaning that this provision does not impose the obligation, but there may be other law that does impose it.

For example, “if the employee works over 7 hours, the employer must give a meal break” would not normally be intended to imply that there is no legal obligation at all to provide a meal break to a 6-hour employee, although a particular employee could be contractually entitled to the break. Using formal logic notation would help legislative counsel to ensure we have thought through all these permutations and to check whether we ought to make express provision for “if not P” (instead of just for “if P”) or whether an implied provision about “if not P” is clear enough to readers.

These “if” structures are also often present implicitly in our drafts, such as in “a person who does X must Y” and “a person doing X must Y”. Those are both equivalent to “if a person does X, that person must Y”. Our definitions, with their strict use of “means” and “includes”, can also be rendered as “if” and “iff” structures.

> “Ship” means a large boat = If (and **only** if) a boat is large, it counts as a ship = $S \leftrightarrow LB$
>
> “Ship” includes kayak = If (but **not only** if) this is a kayak, it counts as a ship = $K \rightarrow S$
>
> “An X must do Y” + “X **means** a Z that is a T” = “**If** a Z is a T, the Z must do Y”

This last one is not "iff", despite the "means", because "an X must do Y" translates as "if, but not only if, something is an X, then that thing must do Y". As mentioned above, when we say "if an inspector serves a pig hygiene notice, the farmer must clean the pig pen" we are not necessarily implying that there are no other circumstances in which the farmer must clean the pig pen (and it may or may not be clear enough in a particular case for the draft to rely on the reader using cannons of construction such as *expressio unius*).

It is worth mentioning the dangers in using negation with "if", which relate to the caution drafters exercise over "unless". I will not go into that further here, but the reader is invited to try using "unless" to solve the human transplantation rewrite problem set out above (in relation to "not").

"Must", "may" and "is" – the logic of our core elements

Deontic logic – "must", "must not" and *"may"*

A key part of what legislative counsel do is to impose legal obligations and prohibitions (that would not have been there otherwise) on a person in given factual circumstances and to carve out permissions from those obligations and prohibitions. We also take care to ensure there is a legal consequence if the person contravenes the obligation or prohibition. These features are what make our provisions a legal rule, as opposed to a mere statement of fact. In modern Commonwealth drafting we have three basic building blocks we use to do this –

"must",

"must not",

"may".[20]

The branch of logic that attempts to systematise these concepts is deontic logic, which deals in obligations, prohibitions and permissions. Deontic logic can produce sophisticated renderings of rules.[21] In modern drafting, as in deontic logic, these building blocks can be treated as binary because legislation will rely on an adjudication system to decide whether a given person is or is not subject to the obligation or prohibition and has or has not complied.

[20] Dating back to 1845 in Coode's "legal action" as one of his "essential elements" of a "legislative sentence" – see "On Legislative Expression", reprinted in EA Driedger, *The Composition of Legislation*, 2nd ed. (Department of Justice: Ottawa, 1976) at 317. In those days "shall" was used rather than "must" (and it still is used in some Commonwealth offices). It causes problems not only because some readers think it is future, but also because it blurs the difference between obligations and other elements, such as in "shall mean" or where the provision means something happens by operation of law (as in establishing a legal person).

[21] More sophisticated when combined with "predicate logic". This article sticks to "propositional" logics that only operate on a whole proposition (a complete statement that can be true or false). Predicate logic breaks down propositions, rendering "a person walks" as "pW", instead of just "P" ("a person" alone is not true or false, nor is "walks"), to allow quantification (all, some, none) and preserve references to the same person between two propositions.

But there are several problems with deontic logic. For instance, one of the most troublesome is the “gentle murder” or “contrary to duty” paradox[22] where simple forms of deontic logic struggle with taking “you must not murder” and adding “if you murder, you must murder gently” because these forms of logic produce the contradiction “you must murder”. But legislative counsel are familiar with the idea of aggravating factors and overlapping offences of increasing seriousness. We might draft an offence, carrying a low penalty, of contravening a provision that “a person must not assault another person” and another offence, carrying a higher penalty, of contravening another provision that “a person must not assault another person with a knife”, without expecting a reader to think there is a contradiction between the two prohibitions. We would not express the second prohibition as “if a person assaults another person, s/he must do so without using a knife” . Much academic work is devoted to how best to capture legal rules in deontic logic without falling foul of any of its paradoxes, but it is beyond what is needed for the purposes of this article. In particular when I suggest formal logic is useful to legislative counsel, that does not depend on whether deontic logic will eventually succeed in capturing legal rules fully, nor on whether computer logic is better suited to doing so.

“Is (to be [treated as])”, “means” / “includes” – the role of “if / iff”

Before looking further at “must”, “must not” and “may”, we should touch on the other elements in our drafts where we are not using those concepts. One of the results of the shift in the modern style from “shall” to “must” has been a sharper focus on the distinction between imposing obligations and these other elements. Obligations can be contravened, and we need to say what happens if they are. But in other cases where we used to use “shall” we now find “must” does not fit.[23]

In some provisions we are actually creating a legal effect just by operation of law. “There is established an Office of the Digitisation of Statutes” and “The XYZ Act is repealed” each mean some legal change has actually happened; it is not a question of somebody being able to contravene the provision. “The contract is void if X” similarly works by operation of law, but only when an “if” condition is met. For these we now use –

> “is”,
>
> “is to be [treated as]”,
>
> “[other appropriate present tense verbs]”.

Many of these cases might be unpacked as really being about their consequences for a “must/may” provision elsewhere in the draft, where the “is” provision serves to feed into

[22] See H Prakken and M Sergot, "Dyadic Deontic Logic and Contrary-to-Duty Obligations" in E Nute (ed) *Defeasible Deontic Logic* (Springer: 1997), available at https://www.doc.ic.ac.uk/~mjs/publications/DyadicDeontic.pdf .

[23] See “constitutive rules” and “counting as” in Sartor, above n.3.

the “must/may” provision. For instance, a provision saying “the contract *is* voidable if X” could be unpacked as

> “if X, then a party to the contract *may* Y, and if Y then all parties *may* act as if the contract had not been made”.

Those cases seem to be different from definition provisions, which are merely flagging the use of particular words in the draft (often merely as a convenient shorthand), rather than changing the law or making anything happen. For definitions we now use –

> “means”,
>
> “includes” (and “does not include”).

As explained above on “if” and “iff”, there is also a sense in which definitions can be seen as another set of “if” provisions (for “includes”) and “iff” provisions (for “means”).

Then there are cases like “the application/notice is invalid if it does not contain X” (which is sometimes rendered as “the application/notice *must* contain X”, even though it is not couched in terms of imposing an obligation on a person). In one sense that amounts to no more than saying that the application or notice will lead to certain legal consequences (the regulator must grant or consider the application, or the person on whom the notice is served must cease trading) as long as it is made/served in compliance with a set of “if” conditions. That is using “invalid” as shorthand to cover the class of applications/notices that do not meet the “if” conditions and therefore do not lead to the legal consequences.

“Must”, “must not”, “need not” – how can deontic logic symbols and negation help?

Now we come back to “must” and “must not”, before moving on to “may”.

“Must” – In symbolic notation for deontic logic “must” can be rendered as “O”, which can also be translated as “it is obligatory that”. So “O*P*” means “it is obligatory that *P*”, where “*P*” is a proposition that can be true or false (but is not made true just by being obligatory). But as legislative counsel we are not dealing with how the world should be in general – we need someone obliged to make it that way (and a consequence if they do not). That can be rendered as “O*xP*”. That equates to “it is obligatory on *x* that *P* happens”, where “*x*” stands for a type of a person and “*P*” stands for a state of affairs or action. That equates to “*x* **must** do *P*”, or “*x* must bring it about that *P* is the case” (where *P* is not necessarily *x*’s action), or “*x* must see to it that that *P* is the case” (abbreviated to STIT). So if “*x*” stands for “a person who sets up a business” and “*P*” stands for “the person on whom the obligation is imposed applies for a licence”, then “O*xP*” can be used to stand for “a person who sets up a business must apply for a licence”.

"Must not" and "need not" – The magic of this "O" is that you can apply a negation to it in two different ways. The negation symbol (here "~") always comes immediately before whatever it negates. That gives you –

- "O*x*~*P*" as "it is obligatory on *x* that not-*P*" – "*x* does have to not-apply-for-a-licence", which means "*x* is prohibited from applying for a licence", which we express as "*x* **must not** apply for a licence".
- "~O*xP*" as "it is not obligatory on *x* that *P*" – "*x* need not apply for a licence" – which sounds like "may" (discussed next).

The prohibition works on the footing that either *P* is the case or it is not, so anything other than *P* is not *P*. Not *P* ("~*P*") captures everything *x* could be doing, being or whatever, apart from *P*.

Weak "must" – There are uses of "must" that do not fit the classic Coodean idea of a person being obliged to do something or face a sanction.[24] For example, sometimes we say "an application/notice *must* contain" or "an appeal *must* be lodged within 28 days". These do not fit the model of imposing an obligation on a person who faces a sanction for breach. Instead they are better seen as concerning validity, in the sense of whether my action will have its intended legal effect in imposing some duty on someone else. There is not space here to go further into that or into other examples of "weak must",[25] but I touch on similar issues with "awkward may" below.

"May" – what can logic reveal about what it is doing in drafts?

"May" is slightly fiddly compared to "must".

- Returning to *P* as an action being taken by the person on whom the obligation is imposed, so far we have seen "O*P*" (you must do the action), "~O*P*" (it is not the case that you must do the action), and "O~*P*" (you must not-do the action). That leaves "~O~*P*" (it is not the case that you must not-do the action).
- There is a sense in which, if you *must* do something, then you may do it. It would be a drafting error to require someone to do something if elsewhere you had said they were not permitted to do it. In the same sense if you *must* not-do something, then you *may* not-do it. But that is a very weak sense of permission and is not normally how we use "may" in drafting.
- Legislative counsel normally use "may" in a stronger sense, to mean you are permitted to do the action or not at your discretion. That can be rephrased as "it is not obligatory that you do the action, and it is not obligatory that you don't do the action".

[24] See n. 20 above

[25] Kowalski, above n. 10 favours avoiding deontic logic because of the problems in its sophisticated operation. One approach might be to treat all "must" provisions, whether "weak" or Coodean, merely as if-then provisions leading to whatever consequences are set out (or implied) as the result of the "must" requirement or condition not being met.

- Going back to licences – in our legislation "you may apply for a licence" normally implies "also you may not-apply-for-a-licence". That is different from "you may-not apply-for-a-licence", which really means "you must-not apply-for-a-licence" – that ambiguity is why we need to avoid "may not" or use it with care.
- Switching the "may" out for "not-obligatory" gives "it is not-obligatory that you apply for a licence, and it is not-obligatory that you don't apply for a licence".
- In the symbolic rendering that is "**~O*P* ∧ ~O~*P***".

Interchangeability – So far we have seen how to use the logical ideas of negation and "and" on the trio of "must", "must not" and may" to reduce them all to "must" –

"must X"

"must not X" = "must ~X"

"may X" = "(~must X) & (~must ~X)".

These can be rendered in an exaggerated version of a logician's natural language as –

It is the case that (it is obligatory to see to it that (it is the case that (X is true))).

It is the case that (it is obligatory to see to it that (it is *not* the case that (X is true))).

It is *not* the case that (it is obligatory to see to it that (it is the case that (X is true))) and it is *not* the case that (it is obligatory to see to it that (it is *not* the case that (X is true))).

If we use the symbol O, and drop the X, then we get –

Must	=	O
Must not	=	O~
May	=	~O ∧ ~O~

That is the whole of our main tool-set reduced to one symbol and two connectors.

Alternative symbols – Sometimes "must not" is represented as F (forbidden), and "may" as P (permitted). Sometimes, confusingly, it is the other way round – P for Prohibited and F for Facultative. It does not matter which way, as long as you are consistent (in the particular logical language). Equally the terms could all instead be reduced to prohibition (P), instead of obligation –

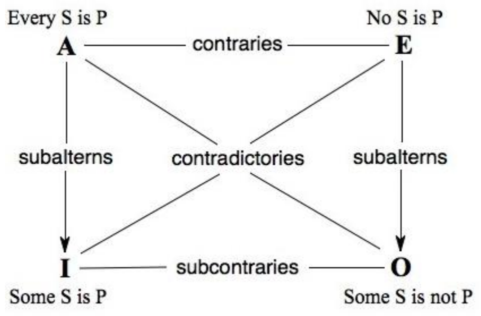
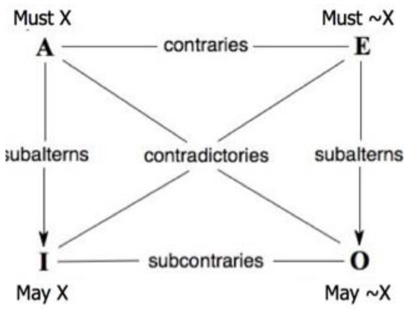
must = P~ ; must not = P ; may = ~P ∧ ~P~

Again, it does not matter which way around it is done, as long as you are consistent. I will stick to "O" here.

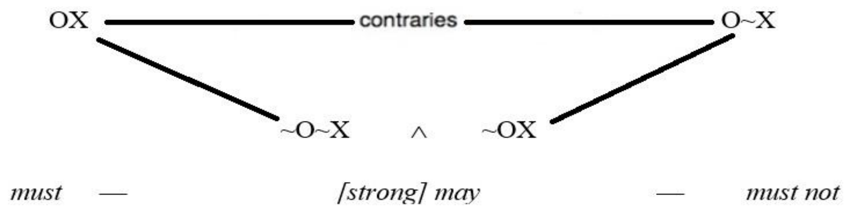
Ambiguous "may" – Turning back to the ambiguity of "may" –

- the strong "may" is "~O ∧ ~O~", where you can choose either way;
- the weak "may" is just "~O", which can be compatible with a prohibition;
- the weak "may-not" is just "~O~", which can be compatible with an obligation.

These relationships can be represented by a "square of opposition" in which the contraries are the opposite state of affairs, whereas the contradictories are merely inconsistent. The original square was produced to illustrate Aristotle's logic of "all", "none" and "some", but has since been applied to deontic logic.

All/Some/None	Must/May/Must-not
 Every S is P No S is P A contraries E subalterns contradictories subalterns I subcontraries O Some S is P Some S is not P	 Must X Must ~X A contraries E subalterns contradictories subalterns I subcontraries O May X May ~X

The ambiguity of "some" ("at least part of, and maybe all of" or "at least part of, but not all of") reflects the ambiguities of "or" and "may". Normal English conventions would be that if you say "Some S is P", you are implying that not all S is P. Drafting conventions take that further – if we meant "All S is P" we would have said so and our choice of "Some S is P" must be read as implying "not all". Equally with "must" and "may", drafters would not say "may" and leave it at that if we meant "must" – a weak "may" does not tell you the actual position, unlike the strong "may" which does. So, if we mean "must" or "must not" then we will say them, and finish the job, without bothering with "may". That squeezes the bottom of the square into a trapezium or triangle, with the "may"s at the bottom linked by "and" in the drafter's strong "may" –



But are we always clear about the distinctions, or are we sometimes lulled into assuming "may" is unambiguous (as we can be with "or")? We are often saved from embarrassment by the common law background to our drafting, which adds three more factors.

- A "**natural person**" (a human "individual") is entitled to do (or not do) anything that is not prohibited and entitled not to do (or to do) anything that is not obligatory. So the default for a human person is that they "may" do anything (in the strong sense of "may") unless the law says otherwise.
- By contrast a "**legal person**" is commonly a "creature of statute". It is created by the statute itself, like a Minister in some jurisdictions, or created by somebody taking some action under the statute, like a company or foundation. It will only have power to do what the statute enables it to do (of course the Crown is

different, not being a creature of statute). So we do need to use “may” somewhere for legal persons.

- We normally only legislate to **change** what would be the underlying position anyway.
- So, when a human is the subject (of our “must”, “must not” or “may”), we generally do not need “may” because it is implied anyway, and it is the “must” or “must not” that we need to carve out of the underlying freedom. We then need “may” only where the draft has already imposed a “must” or “must not”, from which we now need to carve out an exception reverting to the underlying freedom. But with a legal person, are we really carving out what it “may” do from an underlying prohibition, or is it sometimes a case of powers, to which I turn next?

Awkward “may”, treacherous “may not”, discretion/power, Hohfeld’s power/liability

Carving out exceptions with “may” is the clearest case, but it also helps in analysing the less clear cases, such as “A car-owner may apply for registration”. Of course none of us is breaking the law if we apply pointlessly to people for things that they cannot give us. So this is not a case of the draft containing (somewhere else) a prohibition (“must not”) on making applications, from which the “may” is now being carved out as an exception. Instead, as mentioned above, there is a sense in which “W may apply to X for Y” implies that, if W does so, then X is going to be obliged to give Y to W, if conditions Z (assumed to appear later in the draft) are met. If we treat this as an awkward case of “may” (not fitting the model of being neither obliged nor prohibited) then it can be unpacked as short-hand for –

> “If W applies to X for Y (and of course nobody can stop W doing so, or force W to do so), and conditions Z are met, then X must give Y to W”

In symbolic terms that can be rendered as –

> “$(\mathrm{A}wxy \wedge \mathrm{Z}) \rightarrow \mathrm{OG}xyw$”.

But this case might reflect an important distinction.

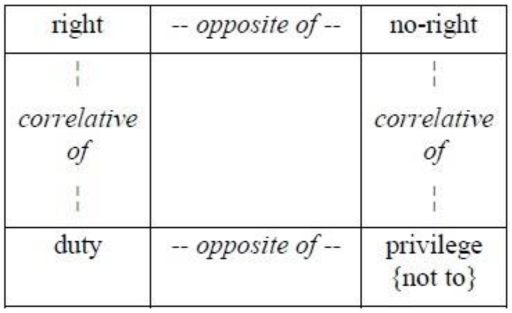
- At one level it is the legislator who, by enacting legislation, imposes legal obligations or prohibitions, or grants permissions, affecting the legal status of citizens or other persons (natural or legal).
- But at another level we all impose legal obligations and prohibitions on, and grant permissions to, each other every day – by entering contracts, granting or refusing permission to enter our land, and so on.

In that light, the effect of “may apply” is that a private citizen can impose a legal obligation on an authority to consider granting (and possibly to grant) something. The citizen does so by applying to that authority for that thing in the way that we have provided for with our “may apply”.

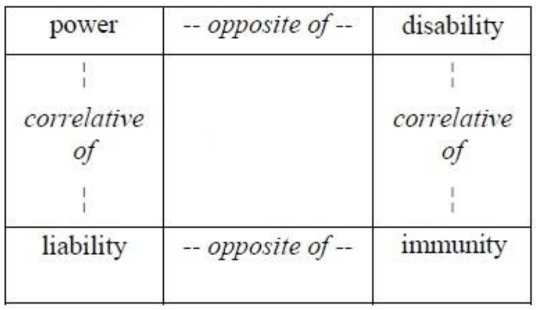
Another common case of “may” is “the inspector *may* serve an enforcement notice on the business-owner”. Inspectors generally don’t breach a prohibition by giving people ineffective paper (without intending to deceive, at least), so this is not carving out from a prohibition. Instead, it can be seen as reflecting a difference between “may” as a “power” and “may” as an ordinary freedom or discretion.

But then there is “may not”, which I have already mentioned as ambiguous and to be avoided unless the particular use is unambiguous and convenient. Usually it can be replaced by “must not”. But sometimes it is used as an exception to a power, as in “the inspector *may not* serve an enforcement notice if…”. If we used “must not” there, it might imply the notice would be valid and there would be some other sanction on the inspector, whereas “may not” seems to imply the notice would be invalid (with no further sanction). Those distinctions seem better dealt with explicitly, rather than hanging too much weight on “must/may not”.

A key early writer on rights and powers was Wesley Hohfeld.[26] He analysed these and other concepts, and fitted them into a logical grid of their relationships to each other, represented here as two squares of opposition.



right	-- *opposite of* --	no-right
correlative of		*correlative of*
duty	-- *opposite of* --	privilege {not to}



power	-- *opposite of* --	disability
correlative of		*correlative of*
liability	-- *opposite of* --	immunity

Hohfeld takes the common idea that my right is reflected in your duty (adding that the absence is my “no-right” and your “privilege”) and works up to the idea of my “power” to do something that imposes on you a duty you did not already have. So, you have a corresponding “liability” before you have the duty. Then if I do not have the power then I have a “disability” and you are correspondingly immune.

But do we really need to analyse it in that way? As legislative counsel we happily build on “must”, “must not” and “may” for nearly all our needs, rarely having to trouble ourselves with these additional concepts. This results

- partly because we favour a simpler term over a more complex one;
- and in turn partly because more complex common law terms can set hares running that we do not need;
- and perhaps partly also because we are not often tinkering directly with the richer common law concepts, given the acute awareness among Commonwealth legislative counsel of the folly of trying to pin live butterflies.

[26] In his 1913 and 1917 Yale Law Journal papers, both called “Some fundamental legal conceptions as applied in judicial reasoning” – https://digitalcommons.law.yale.edu/ylj/vol23/iss1/4.

We could break down Hohfeld's concepts into "must", "must not" and "may", but that is beyond the scope of the present article. Meanwhile Sartor offers a symbolic logic analysis of Hohfeld.[27]

Conclusion

I hope this outline shows that the basic notation system of formal symbolic logic can be useful to legislative counsel and help add further rigour to the way we check our drafts using our common sense logic. Some of that help is fairly routine, such as reminding us to check for inclusive or exclusive "or", and some is less routine, such as checking for "if" or "iff". But some of it should also make us clarify what we are doing with our "is/must/may" tools. The switch from "shall" to "must" or "is" has already shone some light on those tools, and the application of formal logic should help take that further.

[27] Sartor, above n. 3.

{extracted from}

Commonwealth Association of Legislative Counsel

THE LOOPHOLE

October 2021 (Issue No. 2 of 2021)