The Legislative Recipe: Syntax for Machine-Readable Legislation

The prospect of machine-readable legislation is both terrifying and thrilling. This renewed popularity is owed to the Rules as Code initiative. The fervor around Rules as Code was accelerated by the recent OECD Observatory of Public Sector Innovation Report titled, "Cracking the Code."

This report articulates how machine-consumable, defined as machines understanding and actioning rules consistently, "reduces the need for individual interpretation and translation" and "helps ensure the implementation better matches the original intent." This methodology enables the government to produce logic expressed as a conceptual model – in effect, a blueprint of the legislation.

So, what is the attraction and what are its limits?

I frequently turn to this example. Layman E. Allen lamented about ambiguity in legal drafting owed to syntactic uncertainties. In a fascinating study, he deconstructs an American patent statute and notices immediately the complexity with the word 'unless.' He asks whether the inclusion of 'unless' asserts a unidirectional or a bidirectional condition. That is, does the clause mean (a) if not x then y; or (b) if not x then y and if x then not y?

Though nuanced, Allen exposes an ambiguity that muddies the legal force of the statute. An interpretation of 'unless' as a bidirectional condition raises the question of what "not y" would mean. In this particular case, this could affect whether exceptions are possible in determining patent eligibility. In short, for Allen, legislative language must have a clear structure.

These ideas are not new. The ancestry dates back to twelfth century logicians reflecting on the use mathematically precise forms of writing. In the mid-1930s, German philosopher, Rudolf Carnap, reflected on a logical syntax for language. His argument is that logic may be revealed through the syntactic structure of sentences. He suggests that the imperfections of natural language point instead to an artificially constructed symbolic language to enable increased precision. Simply put, it is treating language as a calculus.

More recently, Stephen Wolfram made a similar argument. Simplification, he states, could occur through the formulation of a symbolic discourse language. That is, if the "poetry" of natural language could be "crushed" out, one could arrive at legal language that is entirely precise.

Machine-readability appears then to bridge the desire for precision with the inherent logic and ruleness of specific aspects of the law. In other words, a potential recipe to resolve the complexity of legalese. However, if a new symbolic language, like code, effectively enforces a controlled grammar, what are its implications as it moves across the legal ecosystem (in particular, its interactions between various legal texts)?

Machine-readable legislation may, therefore, be regarded as a product that evolved out of the relationship between syntax, structure, and interpretation. But at the core, it boils down to one question: what should be the *role* of machine-readable legislation? Is it simply a 'coded' version

of legislation? (one possible interpretation); or is it a parallel draft of the legislation (one that is legally authoritative)? Or, is it a domain model of regulation from which third parties derive versions (open-source code)?

These three scenarios have their own sets of implications. And only in answering this question, would a fruitful assessment of how logic syntax and symbolic language, found in machine-readable legislation, are capable of representing legal knowledge.