

THE RESEARCH UNIVERSITY PRESS

International
Computational Law Journal

VOLUME 2 / ISSUE 1 / CURATED SOURCE EDITION

Law and Computation: A Conversation

Meng Weng Wong · Marc Lauritsen

MIT Computational Law Report · Originally published 2024-09-10

Independent reader edition under CC BY 4.0. This attribution cover was added; original source PDF pages follow unchanged. Original authorship and source identity are retained. MIT, Stanford University, Stanford Law School and CodeX have not sponsored or endorsed this republication.

Source, license and version evidence accompany the online article.

MIT Computational Law Report

Law and Computation: A Conversation

Meng Weng Wong Marc Lauritsen

Published on: Sep 10, 2024

URL: https://law.mit.edu/pub/lawandcomputationconversation

License: Creative Commons Attribution 4.0 International License (CC-BY 4.0)

ABSTRACT

This conversation between a computer scientist and a lawyer/technologist – about evolving collaborations across their several disciplines – was triggered by interactions at the SubTech'conference in Singapore in July 2022. Together with Alexis Chun they recently published Using Domain-Specific Languages in Legal Applications in *The Journal of Robotics, Artificial Intelligence & Law*.

Meng Weng Wong, principal investigator at Singapore Management University's Center for Computational Law, is a computer scientist (CS Penn '97) and co-founder of Legalese, a deep-tech startup that applies computer science to law. Meng previously designed Internet email infrastructure (RFC4408) and co-founded two high-tech startups and a startup accelerator, JFDI.Asia. He's been appointed to research fellowships at Harvard's Berkman–Klein Center for Internet & Society, Stanford University's CodeX for Legal Informatics, and Ca'Foscari University.

Marc Lauritsen, president of Capstone Practice Systems, is a Massachusetts lawyer and educator with an extensive background in practice, teaching, management, and research. He helps people work more effectively through knowledge systems. He has taught at five law schools, done pathbreaking work on document drafting and decision support systems, and run several software companies. Marc is a fellow of the College of Law Practice Management, past co-chair of the American Bar Association's eLawyering Task Force, and the author of *The Lawyer's Guide to Working Smarter with Knowledge Tools*.

`Marc` Tell us about your center and its work, Meng.

`Meng` At the Center for Computational Law, we're interested in improving access to justice in ways deeply informed by computer science. Our main project now is the imaginatively named Research Programme in Computational Law, where we're building L4, a domain-specific language and toolchain for expressing and communicating legal rules as code. The Programme and Center are unique in that while we sit in a law school, and my co-directors are lawyers and law professors, our main output is (open-sourced) software. So our team includes engineers with backgrounds in computer science, logic, and math. The Programme was launched in 2020 with a generous grant from Singapore's National Research Foundation. This means we've been working on this for a few years and are about ready to launch some early products for use by the scientific and legal communities, so stay posted for announcements.

`Marc` What are the biggest opportunities at present in your view?

`Meng` One premise of our project is that every profession has been transformed by information technology – except the legal profession. The most successful companies in every industry have leveraged Marc Andreessen's famous insight that "software is eating the world" – but most lawyers today still mostly use

computers for fairly mundane purposes, like word processing, research, and hearings via Zoom. Meanwhile, software developers have created entire stacks of computer-aided software engineering tools to manage things like decision logic, code review, debugging, verification and validation, you name it. The theory and practice of law differs from code in meaningful ways, but we believe that there's large, untapped potential for lawyers to adopt tools and practices from the world of software. Take version control – perhaps the lowest hanging fruit. We routinely hear from software engineers who find themselves dumbfounded when interfacing with legal – emailing Word documents back and forth with file names like "Jan 12 edited by Joe Feb 14 edited by Sam final revised final copy clean.docx" … because Git and Github have solved that entire class of problems. Making legal workflows more efficient is especially important today when, as I understand it, law firm clients are less willing to pay by the hour, and inhouse legal budgets are constantly being squeezed. All while the volume and complexity of legal regulation is growing. The world of software offers tools that promise to help non-lawyer citizens and consumers make sense of contracts and regulations, and to help legal drafters anticipate loopholes and better architect their "code".

`Marc` What do you see as the biggest challenges?

`Meng` The dream of computational law is frankly not new, and had been discussed even before Love and Genesereth's seminal 2005 paper christened the field as "computational law". Many giants before us have devised intricate and well-thought out approaches for making law amenable to computer processing, and our research thankfully gets to stand on their shoulders. But it is probably fair to say that many efforts have not made it out of the academy and into practice. Let me highlight one challenge in theory and one in practice.

The theoretical challenge is broadly around mapping legal concepts to code – to logic – to some standard format or protocol that formalises an understanding of legal rules. Specifically, what aspects of law can be, or are already fundamentally, computable, and how might we best represent them in some agreed structure? Taxes and regulations to do with entitlements and benefits, are the easy examples: they are already mostly mathematical algorithms phrased as word problems. Financial contracts – investments, insurance, trading – are also, crudely speaking, about how events affect the transfer of money. The challenge is that beyond these domains, law exists to address an unbounded set of societal problems, and some of those problems may not be computable, formally speaking. So, unlike what some critics of computational law seem to assume, we are not aiming to make all laws computable – only the bits which can be usefully made so. There has been significant scholarship on which bits matter, but no clear consensus. When in doubt, we "bottom out" our legal rules in human judgement, just as the legal system does, by calling out to witnesses – and judges – to turn messy shades of gray into black-and-white matters of fact.

The practical challenge is adoption. As I said above, many computational law systems have been proposed but not many are widely used in industry. It is not easy to convince typically overworked and well-compensated lawyers to incorporate new technologies into their workflows, especially if they are told that they first need to study formal logic, data structures, algorithms, and programming language theory to use them properly. There

are lawyers who are very tech-savvy and maybe even coders themselves, but we can't expect the average lawyer to pick up a DSL and start coding it in – at least not at the start. Developers are frequently told: you have to meet users where they are. And the clue there is "users". William Gibson's famous quote "the street finds its own uses for things" is key here. Lawyers and legal tech is a little bit like taxi drivers and Uber – there are tectonic shifts going on that bring end-users to the fore, and disintermediate a traditionally licensed industry. That's why we heed the difference between the legal industry and the legal profession – they used to be one and the same, but we believe that a growing segment of laypersons are opting to author their own contracts, plan their own wills, and try to make sense of legal rules for themselves, without relying on qualified, certified, billable-hours lawyers.

We've seen this play out over the course of our research programme. We are mandated to conduct use-inspired research, which means we develop L4 incrementally in response to use cases from industry partners, while keeping an overall eye on language design. Our use cases have come to us from a range of sources – from government agencies to blue-chip insurance firms, but one thing they all have in common is the theme of formalising legal rules in code, and turning that code into more code: user interfaces that help citizens and companies understand their rights and duties; authoring IDEs that tell legal drafters where bugs and loopholes and Catch-22s lurk in their "programs"; and runtime rule engines that can be queried at scale by operational enterprise systems.

`Marc` That sounds intriguing; so how, in your system, do rules turn into code?

`Meng` How do we translate natural-language laws into formal specifications? This is a question familiar to any working developer: how to convert high-level requirements and product-manager pseudo-code into actual software that a compiler accepts? As any developer will tell you, working through that process will uncover hidden traps and cans of worms – users and project clients will say they want a thing, but they will say it in vague and ambiguous terms – no shade being thrown, this is an unavoidable part of the process; then the thing developers build may match the letter but not the spirit of the requirements; and often it will transpire that different users want different things, so someone has to lay down the law, no pun intended.

Interestingly, in every project we've done, we have discovered some fundamental "bug" in the legal text we were handed: a law might require a party to both do something and not do the same thing; a contract may rely on an undefined term to the tune of millions of dollars in interpretive latitude. These bugs are familiar to software engineers, under names like "race conditions" and "concurrency mutability bugs". But they are universal to all designed systems. And we are designing our compiler to warn drafters accordingly: "syntax error in section 12; logic bug in section 42; model checking fails verification."

The next challenge is a problem of scale. The "knowledge acquisition bottleneck" has haunted A.I. for decades. To date we have been responsible for encoding legal rules as code. For the future, we are exploring large language models (LLMs), familiar to the public as ChatGPT and others, to help automate the conversion from

natural language to formal language. We presented a paper on that theme at the JURIX conference last year. (https://ink.library.smu.edu.sg/cgi/viewcontent.cgi?article=5999&context=sol_research) The pipeline we devised incorporated some elements of machine learning, and we are optimistically researching how large language models might further help us here.

Sorry to hog the airwaves. Actually, you've been in this 'space' for decades, Marc. Where do you see the opportunities?

`Marc` Thanks. I had my first taste of this overall field when I took an Artificial Intelligence course (taught by Patrick Winston) as an MIT freshman. It wasn't until I had graduated from law school, practiced for seven years, and began work as a clinical teacher that I began looking more closely at the opportunities in law. Harvard's Project Pericles (1984–88) supplied that occasion, and got me into both deeply reading the literature and building practical applications. My focus moved from teaching practitioners to teaching machines how to do useful work for lawyers, and for those without lawyers.

As a former legal aid lawyer I naturally gravitated to opportunities to address the scandalous lack of legal help most folks of limited means face. Document automation and expert systems offered promise, and I began creating them for use in the Harvard clinics. Client facing versions ensued, and once delivery over the Web became practical nationwide efforts began to blossom. LawHelp Interactive is one impressive outgrowth, serving nearly a million bespoke guidance sessions and document packages for free each year.

But much of my work has been in the for-profit sector, helping law firms and law departments more effectively serve clients by leveraging knowledge-based systems. Working Smarter with Knowledge Tools was an attempt to promote awareness across the profession.

The other domain in which I see endless opportunity is legal education. The rise of non-carbon-based intelligence will deeply disrupt the profession and open major new fields of study. Progressive law schools will embrace whole new ways of preparing students and responding to sharpened demands for lifelong learning.

`Meng` And the challenges?

`Marc` Like you, I've seen adoption as a major challenge. Not only in law firms, where hourly billing tends to act as a disincentive to efficiency, but in law departments, courts, and legal aid programs, where that factor is absent. People resist change, even when it's 'good' for them. Trained as professional skeptics, lawyers quickly see blemishes in proffered improvements. (Yet as herd creatures, they also often jump on bandwagons.)

Funding of course is a major issue for nonprofit and public sector initiatives. And there is the disciplinary divide you mention. Lawyers and computer scientists come from different 'planets.'

I'm a relatively unusual example of an interplanetary commuter, honed largely by regularly reading *The Communications of the ACM* and *Technology Review*, and participating in the international conferences on AI

and law. But even though my computational literacy exceeds that of the average legal professional, I'm often out of my comfort zone when it comes to dense symbolic expressions or extensive code.

We've of course got a long way to go in optimizing human/machine collaboration. Even with today's conversational modes, the user experience leaves a lot to be desired. I tackled some of the issues in Toward a Phenomenology of Machine-assisted Legal Work.

And then there's the "unauthorized practice of law," which is often alleged against applications that come too close to what lawyers do. Overzealous 'turf' management by bar authorities, misguided regulations, and entrepreneurs who are too easily chilled represent a serious challenge to progress. My own view is that interactive expressions of legal knowledge are works of authorship protected by rights of free speech. See Liberty, Justice, and Legal Automata.

`Meng` We've been cautious about unauthorized practice of law, too. From an innovation-theoretic perspective, it's been interesting to see how many recent innovations have originated in regulatory gray areas: Uber and AirBnB are the obvious examples. But it also speaks to pent-up demand for alternative business models. Clayton Christensen would call this a disruptive innovation – if we can have taxi services without taxi drivers, and we can have accommodation without hotels, maybe we can have legal services without lawyers! What form those services will take, and how consumers are protected in that brave new world, are of course open to lively societal debate, but the time is clearly nigh.

`Marc` You've read deeply in the literature of this field. Some suspect that your approach may be old hat, revisiting ideas about rule encoding and expert systems that were already popular in the 1980s. How do you respond to that concern? Where's the non-obvious innovation? In what ways is current reality more conducive to rules-as-code strategies?

`Meng` In 1998, Don Norman argued in *The Invisible Computer* that a successful technology's natural fate is to disappear into the background. "Distributed systems" used to be a specialised field of study in CS departments, and now have become part of the fabric of our lives (think WiFi, Zigbee, and 5G, not to mention Netflix and, heck, all of AWS). Similarly, "expert systems" used to be a specialised field of study in A.I. departments. Twenty, thirty years ago they started being commercialized by vendors, in the form of paid products like Oracle Policy Automation. Today every web form in any web app more complex than a login screen follows in the footsteps of expert systems. Now they are being reinvented from the open-source, open-standard angle in the form of DMN-based rule engines from Red Hat and new startups like Camunda. So, yes, it's true that expert systems have disappeared into the background, but that doesn't mean they're obsolete – rather, the opposite is true!

The innovation that rules-as-code brings to the table is this. If we can look past the ugly skin of a Word doc, if we can see through the unfortunate limitations of a PDF, and uncover the Platonic essence of the set of legal

rules that form the beating heart of a legal agreement, we can write those rules in a way that is still legible to non-technical, untrained humans, but can also be consumed by machine; and from that wellspring, we can generate expert systems and other software artifacts that faithfully embody the upstream source. When the sources change, everything downstream can change automatically. Automation saves labor and time; and we can seek assurance of fidelity from the esoteric fields of formal methods, theorem proving, and programming language theory.

`Marc` To what extent have recent developments in generative AI changed your thinking or the course of your research?

`Meng` Generative A.I. is certainly impressive, but when it comes to serious matters, legal matters, where money and life-altering consequences are on the line, relying on only an LLM to perform legal reasoning makes about as much sense as relying on only half your brain to think. The natural complement of the right brain is the left brain; the natural complement of an LLM is a reasoning engine, and that's exactly what we've been hard at work building over the past four years. It's only a matter of time before the pendulum swings the other way; indeed, ChatGPT is already building plugins to third-party rule engines. So we're building "algorithms that you can argue with" –where the result of your arguments feed into the system, transparently and publicly, for other users to understand and argue with. We're talking about open-source software. And in the same breath, we're talking about rule of law.

`Marc` Yes. Maybe as I noted in The End of a Legal Ice Age, "Neuro-symbolic artificial minds will preside over vast federations of public and private knowledge stores."

Well, I guess it's a wrap. Great talking with you!

This research/project is supported by the National Research Foundation, Singapore under its Industry Alignment Fund – Pre-positioning (IAF-PP) Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.